CS6868 Quiz 2

Dept of CSE, IIT Madras
Total marks = 24
Time = 60 min

06 Apr 2018

Read the instructions and questions carefully. You can make any reasonably assumptions that you think are necessary; but state them clearly. There are total three questions (eight marks each). You will need approximately 15 minutes for answering an eight marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets.

You will get an answer sheet with 8 pages (if you get a answer booklet with fewer pages then ask for a replacement). Leave the first page empty and start from Page#2. Start each question on a new page. Think about the question before you start writing and write briefly. **For any question, the answer (including the answers for all the sub-parts) should NOT cross more than two pages.** If the answer for any question is spanning more than two pages, we will strictly ignore the spill-over text. If you scratch/cross some part of the answer, you can use space from the next page. You mostly would NOT need any additional sheets.

1. [8] **OpenMP** (a) Give the syntax of OpenMP `schedule` clauses (for static, dynamic and guided policies) in the context of `omp for` pragma? Show with examples. [3]

   (b) What is the default value of the chunk-size in each of these policies? [1.5]

   (c) Assuming that there are four threads, chunk-size = 32, and the loop goes over 256 iterations, state which iterations will be executed by which thread, for each of the three scheduling policies. Assume each iteration takes roughly the same amount of time to execute. [2.5]

   (d) Which of the following is/are NOT conforming as per OpenMP4.0 [1].

```
#pragma omp parallel
{
#pragma omp barrier
}
          (a)
```

```
#pragma omp parallel for
for (int i=0;i<n;++i)
{
#pragma omp barrier
}
          (b)
```

```
#pragma omp parallel
{
#pragma omp for
for (int i=0;i<n;++i) ;
#pragma omp barrier
}
          (c)
```

```
#pragma omp parallel
{
#pragma omp for nowait
for (int i=0;i<n;++i) ;
#pragma omp barrier
}
          (d)
```

2. [8] **Memory consistency models**: a) Give an ordering among the following six memory consistency models: Eventual, weak, causal, PRAM, strict and sequential consistency. [2]

   b) Find below an execution trace. Each line describes a different process. Each process either writes to a variable, or reads from a variable or executes `sync` to synchronize with other processes. The integer values after a `W` or `R` operation represents the value as seen by that operation. The horizontal axis represents the global time line; for instance, as per the global clock, the operation `R(x)2` in process 2 executed after W(x)2 in process 1. There are six different options for the process 4. Using each of these options we get six different sets of traces (for example, {1-3, 4i} or {1-3, 4ii}, and so on). For each of these traces, indicate the strongest memory consistency model, among the above listed six memory models, that satisfies the trace. If none of them satisfy a trace, then mention 'none'. [6]

```
   1.   W(x)1 W(x)2 sync
   2.               R(x)2 W(x)5 sync
   3.               W(x)6 sync
  4i.                     R(x)2 R(x)1 sync
 4ii.               R(x)1 R(x)5 R(x)6
4iii.         R(x)1 R(x)2 R(x)6 R(x)5 sync
 4iv.                     R(x)5 R(x)1 sync
  4v.                     R(x)2 R(x)5 R(x)5 R(x)5 sync
 4vi.               sync  R(x)1 R(x)5
```

3. [8] **MPI**: (a) Assuming that MPI_COMM_WORLD consists of three processes 0, 1, and 2, what are the values of x, y and z on each process after the following code has been executed [6].

```
int x, y, z;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
switch (rank) {
        case 0: x = 0; y = 1; z = 2;
        MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
        MPI_Send(&y, 1, MPI_INT, 2, 43, MPI_COMM_WORLD);
        MPI_Bcast(&z, 1, MPI_INT, 1, MPI_COMM_WORLD);
        break;
        case 1: x = 3; y = 4; z = 5;
        MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
        MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
        break;
        case 2: x = 6; y = 7; z = 8;
        MPI_Bcast(&z, 1, MPI_INT, 0, MPI_COMM_WORLD);
        MPI_Recv(&x, 1, MPI_INT, 0, 43, MPI_COMM_WORLD, &status);
        MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
        break;
}
```

(b) Assuming that MPI_COMM_WORLD consists of two processes 0, and 1, what is the expected output?[2].

```
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0) {
        MPI_Recv(&x, 1, MPI_INT, 1, 42, MPI_COMM_WORLD, &status);
        MPI_Send(&y, 1, MPI_INT, 1, 42, MPI_COMM_WORLD);
} else {
        MPI_Recv(&y, 1, MPI_INT, 0, 42, MPI_COMM_WORLD, &status);
        MPI_Send(&x, 1, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD);
}
printf("%d %d\n", x, y);
```