Take-home Exam
(100 points)

If you have any questions regarding the exam, please e-mail Dylan Stark at dstark@cct.lsu.edu.

Return exams by e-mail to dstark@cct.lsu.edu.

(80 points)

1. Describe “cooperative computing” and differentiate it from both capacity and capability computing.

2. What are ClassAds? Briefly discuss the various steps involved in Condor MatchMaking.

3. Define “race conditions” and explain how “critical sections” work.

4. Briefly describe the 3 main types of data scoping constructs in OpenMP.

5. Briefly describe the 6 major calls in MPI?

6. Briefly describe :
    a. Temporal Locality

    b. Spatial Locality

7. Briefly describe
    a. Weak scaling :

    b. Strict scaling :

    c. Differentiate Capacity computing, capability computing, and cooperative computing based on the concepts of weak and strict scaling

8. List the four sources of performance degradation (four horsemen of the apocalypse) and describe how they affect performance.

9. Define what a benchmark is and identify the benchmark used to determine rankings on the Top500 list (http://top500.org).

10. Describe the characteristics of CSP model based computations.

11. Briefly describe:
    a. Memory Bandwidth and units it is measured in

    b. Memory Latency and units it is measured in

c. How does the cache hierarchy improve the average operational behavior in both cases

12. There are three main data scoping clauses in OpenMP, shared, private, and reduction. Give an example of when each might be used.

13. Describe in detail the concepts of OS:
    a. Multitasking :

    b. Multiprogramming:

14. Describe each one of the following :
    a. RAID 0

    b. RAID 1

    c. RAID 5

BONUS : (6 points) Name and briefly describe 3 advanced technologies that may be used in the future for high speed computing.

BONUS : (4 points) Give the number of seconds in a year as a power of 2.

Programming Question
(20 points)

1. Robert who hates parallel programming is attempting to write an OpenMP code to do matrix-vector multiplication; but his code doesn't work! The computed result for ∑c[i] should not be 0. What is the single error in the code? How should Robert fix the problem to get the correct value of Total Sum (stored in ***total***)? (Assume that there are 4 threads all together).

```
int SIZE = 4;
double total=0.0;
#pragma omp parallel shared(A,b,c) private(tid,i,total)
 {
    tid = omp_get_thread_num();
  /* Loop work-sharing construct - distribute rows of matrix */
  #pragma omp for private(j)∑
    for (i=0; i < SIZE; i++)
      {
      for (j=0; j < SIZE; j++)
        c[i] += (A[i][j] * b[i]);
      /* Update and display of running total must be serialized */
        total = total + c[i];
        printf("  thread %d did row %d\t c[%d]=%.2f\t",tid,i,i,c[i]);
        printf("Running total= %.2f\n",total);
      }   /* end of parallel i loop */
    } /* end of parallel construct */
  printf("\nMatrix-vector total - sum of all c[] = %.2f\n\n",total);
```

Sample Output containing the Error :

```
Starting values of matrix A and vector b:
  A[0]= 1.0 2.0 3.0 4.0   b[0]= 1.0
  A[1]= 1.0 2.0 3.0 4.0   b[1]= 2.0
  A[2]= 1.0 2.0 3.0 4.0   b[2]= 3.0
  A[3]= 1.0 2.0 3.0 4.0   b[3]= 4.0

Results by thread/row:
  thread 0 did row 0     c[0]=10.00     Running total= 10.00
  thread 1 did row 1     c[1]=20.00     Running total= 20.00
  thread 2 did row 2     c[2]=30.00     Running total= 30.00
  thread 3 did row 3     c[3]=40.00     Running total= 40.00

Matrix-vector total - sum of all c[] = 0.00
```

2. Robert who still hates parallel programming was trying to run the following code; however while executing the program…the program hangs!!
    a. Identify the error with the following code and
    b. describe how you would go about resolving it. (either MPI code or pseudo codes also acceptable)

```
…
  #define N 10000
  int size, rank, tag = 55;
  double sendbuf[N], recvbuf[N];
  MPI_Status status;
  …
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  if (size != 3) /* we want exactly 3 processes */
    MPI_Abort(MPI_COMM_WORLD, 1);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Send(sendbuf, N, MPI_DOUBLE, (rank < 2? rank+1: 0), tag, MPI_COMM_WORLD);
  MPI_Recv(recvbuf, N, MPI_DOUBLE, (rank > 0? rank-1: 2), tag, MPI_COMM_WORLD,
&status);
  printf("I'm process %d, received: %lf\n", rank, recvbuf[0]);
  …
…
```