

Review Notes for Final Exam

The final exam will be derived from these questions, with the approximate points shown. The midterm exam will be worth approximately 100 points. *Most* of the important material is summarized here. Both the points on each question below and the total number of points for the exam are approximate and may change.

Short Answer Questions: You should be able to answer the following short answer questions:

1. [8] Describe the four sources of performance degradation for HPC and give an example of each.
2. [5] Describe at least five of the types of parallelism found in clusters.
3. [6] Suppose that my application runs in 1 hour on one processor and that 10% of my application cannot be parallelized. According to Amdahl's Law, what is the maximum speedup that I can gain by running the application on a parallel computer, no matter how many processors I run it on?
4. [10] What is meant by the CSP programming model? Describe at least eight characteristics of this in a paragraph or so. Also describe at least three performance issues for CSP.
5. [2] True or false: You can write a program using MPI that will run across all of the cores of your multicore computer in parallel. Also, if this is possible, indicate if you think this is a good way to write the program. You must justify your answer to receive credit.
6. [2] In MPI, when does a blocking recv message return?
7. [2] In MPI, when does a non-blocking recv message return?
8. [4] What is an MPI communicator? What is MPI_COMM_WORLD?
9. [2] In MPI, what is a process rank?
10. [2] True or false: In MPI you set the number of processes when you write the source code.
11. [4] Give a short piece of pseudocode that illustrates the master/slave programming model.
12. [4] Explain if the following MPI code segment is correct or not, and why:
Process 0 executes:

```
MPI_Recv(&yourdata, 1, MPI_FLOAT, 1, tag, MPI_COMM_WORLD, &status);  
MPI_Send(&mydata, 1, MPI_FLOAT, 1, tag, MPI_COMM_WORLD);
```

Process 1 executes:

```
MPI_Recv(&yourdata, 1, MPI_FLOAT, 0, tag, MPI_COMM_WORLD, &status);
MPI_Send(&mydata, 1, MPI_FLOAT, 0, tag, MPI_COMM_WORLD);
```

13. [4] Suppose that process 0 has variable A, and process 1 also has a variable A. Write MPI-like pseudocode to exchange these values between the processes.

14. [4] Explain if the following MPI code segment is correct or not, and why:

```
if (my_rank == 0) {
    mydata = 2.5;
    MPI_Bcast(&mydata, 1, MPI_FLOAT, 0, MPI_COMM_WORLD); }
else MPI_Bcast(&mydata, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
```

If the code is correct, what is the value of mydata in process 1 prior to execution? After execution?

15. [4] Explain if the following MPI code segment is correct or not, and why:

Process 0 executes:

```
MPI_Bcast(&mydata, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
```

All other processes:

```
MPI_Recv(&mydata, 1, MPI_FLOAT, 0, tag, MPI_COMM_WORLD, &status);
```

16. [3 pts each] Explain the purpose of each of the library calls listed.

- MPI_Init
- MPI_Finalize
- MPI_Comm_rank
- MPI_Comm_size
- MPI_Send
- MPI_Recv
- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Reduce

17. [4] Suppose there are 6 processes, P0 up to P5, and that each process has declared array A[6]; Explain exactly the effect of a call to MPI_Scatter with process 0 as the root of the collective communication on the data in array A.

18. [4] Suppose there are 6 processes, P0 up to P5, and that each process has declared array A[6]; Explain exactly the effect of a call to MPI_Gather with process 0 as the root of the collective communication on the data in array A.
19. [4] Suppose there are 6 processes, P0 up to P5, and that each process has declared array A[6]; Explain exactly the effect of a call to MPI_Allgather. Is there a root process with this call?
20. [4] What is an MPI derived datatype and when would you use one?
21. [6] In class we talked about three components of OpenMP: directives, run-time library routines, and environment variables. Give an example of each one of these.
22. [2] What environment variable do you set to specify the number of threads for an OpenMP program?
23. [8] There are three main scoping clauses in OpenMP, shared, private, and reduction. Give an example of when each might be used. The access of which one(s) of these could create a race condition?
24. [4] Give the contents of array x after the execution of the following OpenMP code fragment if possible, or give the error if there is one. Assume that there are 5 threads all together:

```
int i, p, x[5] = {1,1,1,1,1}, y[5]={0,1,2,3,4};
#pragma omp parallel private (p)
{
    p=5;
    #pragma omp for
        for (i=0; i<5; i++)
            x[i]=y[i]+p;
} /* omp end parallel */
```

25. [4] Give the output of the following OpenMP code fragment if possible, or give the error if there is one. Assume that there are 4 threads all together:

```
int i; double sum = 0.0;
#pragma omp parallel for reduction(+:sum)
    for (i=1; i <= 4; i++)
        sum = sum + i;
printf("The sum is %lf\n", sum);
```

26. [4] Give the output of the following OpenMP code fragment if possible, or give the error if there is one. Assume that there are 4 threads all together:

```
int i; double sum
#pragma omp parallel private (sum)
```

```

{
  sum = 0.0;
  for (i=1; i <= 4; i++)
    sum = sum + i;
  printf("The sum is %lf\n", sum);
}

```

27. [4] Give the output of the following OpenMP code fragment if possible, or give the error if there is one. Assume that there are 4 threads all together:

```

int i; double sum = 0.0;
#pragma omp parallel shared (sum)
{
  for (i=1; i <= 4; i++)
    sum = sum + i;
}
printf("The sum is %lf\n", sum);

```

28. [4] Give the output of the following OpenMP code if possible, or give the error if there is one. Assume that there are 4 threads all together:

```

int i; double sum = 0.0;
#pragma omp parallel private (sum)
{
  for (i=1; i <= 4; i++)
    sum = sum + 1;
}
printf("The sum is %lf\n", sum);

```

29. [4] What OpenMP directive is used to provide mutual exclusion synchronization in sections of code? Give an example.
30. [4] Explain how load balancing affects the performance of OpenMP programs.
31. [4] True or false: You can write a program using OpenMP that will run across all of the nodes of your cluster using a large number of threads in parallel. Also, if this is possible, indicate if you think this is a good way to write the program. You must justify your answer to receive credit.
32. [6] What is embarrassingly parallel? Almost embarrassingly parallel? Give the steps of an almost embarrassingly parallel MPI program.
33. [4] Name and describe the two types of locality that a parallel program using array decomposition may take advantage of.
34. [4] Describe with a figure block and cyclic decomposition of a 2-dimensional array. Why might block decomposition be implemented differently in C versus Fortran?

35. [4] Explain why static mapping of data blocks to processors may be bad for the Mandelbrot program. Give one strategy that can provide good load balancing for this problem.
36. [4] Give the basic idea of a Monte Carlo simulation.
37. [4] Explain in a short paragraph what the N-Body problem is.
38. [4] Give the basic idea of the Barnes-Hut algorithm for solving the N-Body problem. That is, describe orthogonal recursive bisection.
39. [6] Give three goals of visualization and three reasons why visualization helps us understand the phenomenon underlying a data set.
40. [4] What is computation steering and how does visualization play a role in this?
41. [6] Describe three challenges and approaches to meeting these challenges for visualization.
42. [4] What is the average (approximate) latency time for a disk access today and what is the typical disk interface bandwidth (specify communication technology)?
43. [10] Expand RAID. Briefly describe RAID0, RAID1, RAID5, and RAID6.
44. [5] Give five reasons why NFS is bad for parallel I/O.
45. [6] Describe three key characteristics of parallel file systems.
46. [8] Describe the four major components of PVFS2 and the functionality of each of these.
47. [6] Give the steps of a file open and also for file data access in Lustre (slice 39 Lecture 19).
48. If the user enters 3 and 17 what is the output of the following code?

```
#include <stdio.h>
int main()
{
    int a = 0, b = 0;
    char buf[10];
    scanf ("%d%d", a, b);
    sprintf (buf, "%d %d");
    puts ("you entered: ");
    puts (buf);
}
```

49. [6] Name the six standard interface functions from the POSIX API. (slide 8 Lecture 20)
50. [10] Describe ten problems with POSIX I/O for parallel applications.
51. [6] Give six features of the MPI-IO library.
52. [8] Name and describe four types of MPI file types other than the elementary data type (slide 21 lecture 20).
53. [4] How is an independent access pattern different from a parallel access pattern (slide 37 lecture 20) ?
54. [2] What is netCDF?
55. [2] What is HDF5?
56. [8] Draw the process state diagram given for Linux. Include a description (reason) that might cause a process to move from one state to another. Can a process move directly from a waiting state to a running state? Why or why not?
57. [6] Describe three types of multithreading models (slide 27 lecture 21).
58. [2] What is the most common operating system used by computers in the Top 500 list?
59. [8] Describe in a short paragraph the two classes of I/O in Linux (slide 33 lecture 22)? What is the purpose of the Buffer Cache?
60. [10] Describe in a short paragraph the purpose and design of the Linux Virtual File System. Draw a simple diagram of its structure (slide 37 lecture 22).
61. [4] Expand BG/L. Who manufactures BG/L?
62. [5] Describe the five types of nodes in the BG/L organization.
63. [6] How is I/O supported in BG/L? Be specific about the capability of compute nodes vs. I/O nodes, and explain how function shipping works.
64. [6] Describe six design goals of the Sandia/UNM Lightweight Kernel.
65. [6] State why I/O is a major factor in system performance. How can performance be improved?
66. [4] Define and differentiate time-sharing and space-sharing concepts in scheduling.

67. [8] Describe at least four main activities of Workload Management Systems.
68. [6] Explain FCFS in the context of scheduling parallel jobs and what limitation it has.
69. [6] Give an example, with a figure, showing how backfill could work in the EASY scheduling algorithm. What is the primary disadvantage of the EASY algorithm?
70. [6] Describe three metrics used to measure the quality of a schedule.
71. [4] What is the purpose of MOM in the PBS system? Where in the cluster does it execute?
72. [4] What is going wrong with this code, which is supposed to print the elements in the array?

```
#include <stdio.h>
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int array[] = { 23, 34, 12, 17, 204, 99, 16 };
int main()
{
    int d;
    for (d = -1; d <= (TOTAL_ELEMENTS-2); ++d)
        printf ("%d\n", array[d+1]);
    return 0;
}
```

73. [6] Describe at least six differences between a static library and a dynamic library.
74. [5] Describe the features of the C++ Standard Template Library (slide 45 lecture 24).

Term and acronyms: You should be able to expand each of the following acronyms, and define each of these terms and acronyms in a line or two [2 points each]

- Supercomputer
- Top 500 list
- Flynn's Taxonomy (all four acronyms and an example of each)
- MPP
- SMP
- Parallel Vector Processor
- Commodity cluster
- Constellation
- Multicore

- Moore's Law
- benchmark
- Latency
- Peak performance
- Sustained performance
- Ops
- Ips
- Flops
- Mflops
- Gflops
- Tflops
- Pflops
- Strict scaling
- Weak scaling
- Capability computing
- Capacity computing
- Cooperative computing
- Memory wall
- Latency
- Cycle time
- Throughput
- Granularity
- CSP
- SPMD
- ILP
- Pipelining
- MIMD
- PGAS
- System Area Network
- UC-Berkeley NOW Project
- Beowulf cluster
- Gigabit Ethernet
- Myrinet
- InfiniBand
- Scheduler
- MPI
- MPICH
- OpenMPI
- NFS
- PVFS
- Condor
- Global barrier
- CSP
- Reduction

- TCP
- IP
- Hyper-transport-based SMP

Formulas and calculations: You should be able to provide the formulas and perform calculations using the following.

- Amdahl's Law, slide 56, Lecture 1
- Amdahl's Law with Overhead, slide 57, Lecture 1
- Speedup and efficiency, slide 5, Lecture 5
- Overhead
- Using Amdahl's Law to calculate speedup, slide 13, Lecture 9