ITCS 4145/5145 Parallel Programming
Final exam
Tuesday December 11th, 2012, 11:00 am - 1:30pm
With solutions

Name: .....

This is a closed book test. Do not refer to any materials except those supplied for the test.

Supplied: "*Summary of OpenMP 3.0 C/C++ Syntax.*"

Answer questions in space provided below questions. Use additional paper if necessary but make sure your name is on every sheet.

Total /60

Do not refer to any materials for this part

Qu. 1 Answer each of the following briefly:

(a) According to Amdahl's law, what is the maximum speed-up of a parallel computation given that 80% of the computation can be executed in parallel? Clearly explain. No points for simply putting down a numerical answer with an explanation. 2

S(p) = p / ((1+ (p-1)f), where f = 20%
As p approaches infinity, S(p) = 1/f = 1/0.2 = 5

(b) Which computer system was used at UNC-Charlotte for OpenMP programming assignments? Why? 2

coit-grid05.uncc.edu. Because it is a shared memory system with many cores (16 cores – 4 quad core processors)

(c) In the Seeds framework, the programmer uses the DataMap class, which extends the Java Hashmap class. How are data items identified within objects of this class? 2

Tuple stored - a key (a string) and associated data object. Obtain the data object by using the key.

(d) Which IDE (Integrated Development Environment) was used in Assignment 1? 2

Eclipse

(e) When does the MPI routine MPI_Send() return? 2

After the local operations have been completed, but not necessarily before the message is sent.

(f) What does the MPI routine MPI_Gather() do? 2

Brings together partial results from the various processors and accumulates them into the final result on the root processor.

(g) How does one specify that a section of code should be run in parallel on available processors using the Paraguin compiler? Give a specific example. 2

Using the begin and end parallel region pragma's. Ex.:

```
#pragma paraguin begin_parallel
…
#pragma paraguin end_parallel
```

(h) Write for Paraguin pragma to execute blocks of iterations of the following for loop in parallel: 2

```
for (i = 0; i< n; i++)
    …
```

```
#pragma paraguin forall C   p   i \
                      0x0   1  -1 \
                      0x0  -1   1

for (i = 0; i< n; i++)
    …
```

(i) Write the Paraguin pragma to get the input data array float a[n] to all the processors. 2

#pragma paraguin bcast a

(j) In the instruction to generate graphical output for Assignment 4, it is suggested that you will need a sleep() statement in the code. Why? What does this statement do? 2

To introduce a delay between each iteration and to get an appropriate speed for the motion to be viewed.

Waits x seconds before returning where x is the argument.

"The *sleep*() function shall cause the calling thread to be suspended from execution until either the number of realtime seconds specified by the argument *seconds* has elapsed … ."
http://pubs.opengroup.org/onlinepubs/009604599/functions/sleep.html

(k) If we have an algorithm whose sequential complexity is O(*N*) and we parallelize it with *P* processors, what is the maximum speedup we would expect (write this as a formula)? Is it possible to achieve greater speedup and, if so, under what circumstance might this happen? 2

The maximum speedup we would expect is linear speedup: S(P)=P.
It is possible to have superlinear speedup. One of the reasons this can happen is because of the increase is collective memory of many processors as compared to a smaller amount of memory for a single processor. The problem may not fit into the memory of a single processors (causing excessive paging), where it will fit into the memory of multiple processors once it has be divided up.
A second reason is that the sequential time is slower than the fastest possible time. We could be comparing the parallel time to the sequential time executed on a much slower computer, or we could be comparing it to a slower sequential algorithm.

(l) What arithmetic property is required for an operation applied as a reduction? 2

commutativity

(m) Under what circumstances would the message tag be imperative in MPI? 2

Where there are multiple messages sent between the same pair of processors. For example, if there are multiple processors sent between process 2 and 5. The message tag is the only way to distinguish those messages.

(n) Describe how sorting could be done using a pipeline pattern 2

Each processor keeps the lowest value it has sees so far and sends all other values to the next processors.

(o) If two threads execute the instruction x++ where x is a shared variable initialized to 0, what are the possible values that x could have after the execution of the threads? Clearly explain your answer. 2

If two threads operating at different times x = 2. Interleaved:

Thread 1	Thread 2
Read x	
	Read x
x++	
	x++
Write x	
	Write x

Get x = 1

(p) How many threads could be used for the computation below, each thread executing one or more of the instructions: 2

```
x++ ;
a = x + 2;
b = a + 3;
c++;
```

without changing the code. Explain clearly your answer.

```
1: x = x + 1;
2: a = x + 2;
3: b = a + 3;
4: c = c + 1;
```

As written, 1 must be executed before 2, and 2 must be executed before 3 because of true read after write dependencies. However 4 is completely independent so two concurrent threads could be used.

Challenge: How could the code sequence be changed to expose more parallelism but still achieve the same final result (including modifying the instructions)?

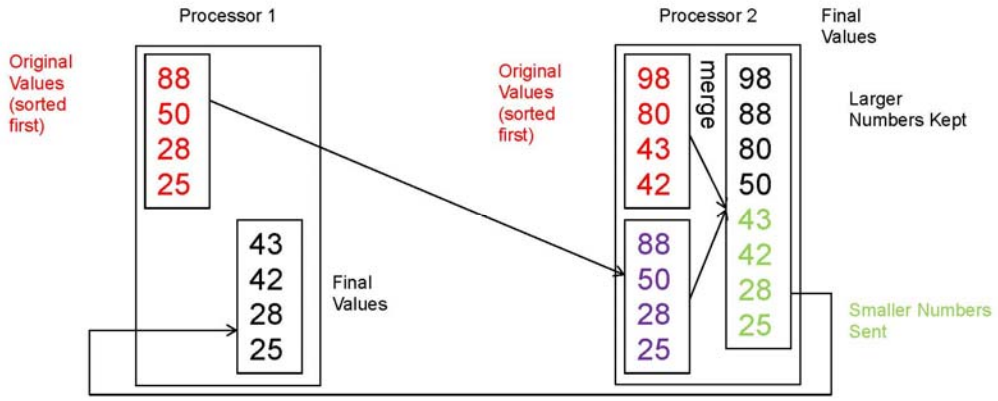
2

Thread 1: x++; thread 2: a = x + 3; thread 3: b = x + 6; thread 4 c++;
Assuming x used is original value

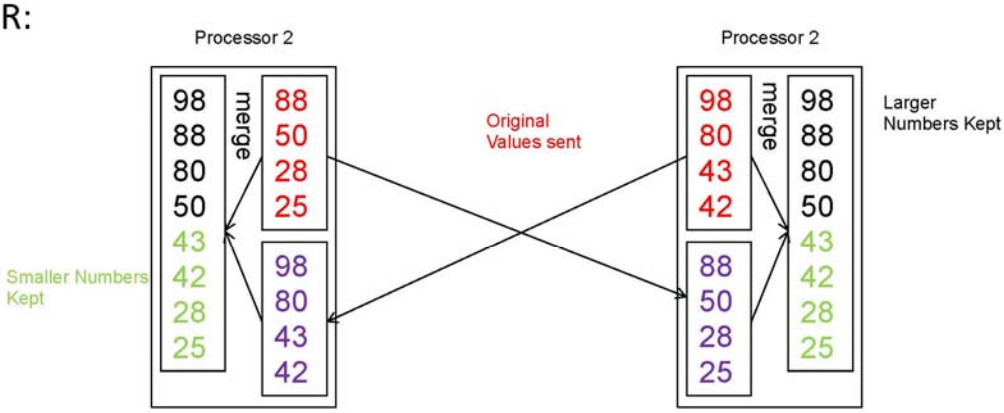
(q) If a routine is *sequentially consistent* when the same routine is executed on separate threads at the same time, why can we say the routine is *thread-safe*? 2

A thread safe routine can be called by multiple threads at the same time and still work.
Sequential consistency says program sequences executed on different threads/processes even if interleaved in time still create the correct answers. These program sequences could be the same routine, which would be thread safe if sequential consistency holds true.

(r) Many sorting algorithms use a compare and exchange operation. Suppose the compare and exchange operation operates on two groups of sorted numbers rather than individual numbers. Describe how the compare and exchange operation can be achieved on a message passing system. 2



OR:



(s) Which pattern is used in GPU programming with CUDA? 2

Data Parallel

Qu. 2 Write a *complete* OpenMP program that determines all the primes from 2 to N^2 using Sieve of Eratosthenes. Share the work across N OpenMP threads.

Provide very clear explanation of how the program works, and comments in your code. If I do not understand the code, I will assume it is incorrect.

10

```
#pragma omp parallel private(rank)
{

  p = omp_get_num_threads();
  rank = omp_get_thread_num();

  proc0_size = (n-1)/p;
  if (2 + proc0_size < (int) sqrt((double) n)) {
    if (rank == 0) printf ("Too many processors\n");
      return 1;
  }

  low_value = (rank*n-1)/p;
  high_value = ((rank+1)*n-1)/p;

  prime = 2;

  do {

    if (low_value % prime == 0) first = low_value;
    else first = low_value + prime – (low_value%prime);

    #pragma omp for
    for (i=first + prime; i<high_value; i += prime)
      marked[i] = false;

    if (rank == 0) {
      do {
        prime++;
      } while (marked[prime]);
    }

  } while (prime * prime < n);

}
```

No “nowait” here. Needs to synchronize after for loop

Qu 3 Write a *complete* CUDA program to add two *N* x *N* matrices, **A** and **B**, where *N* is a constant. Use #define statement for *N*. Use one thread for adding each pair of elements within square 2-dimensional blocks and a square 2-dimensional grid structure. The number of threads in a block and number of blocks in the grid (in each dimension) are to be given as #define constants, *T* and *B* respectively. The code is to work if *T* x *B* is greater than *N*. Initialize each matrix with numbers (any numbers) and print out the final results.

Some CUDA predefined variables:

The thread ID within a block in the x direction is given by threadIdx.x
The block ID in the x direction is given by blockIdx.x
The number of bocks in the x direction is given by blockDim.x

Provide very clear explanation of how the program works, and comments in your code. If I do not understand the code, I will assume it is incorrect.

10

```
#include <stdio.h>
#include <cuda.h>
#include <stdlib.h>
#define N 10     // size of array
#define T 10     // threads per block
#define B 1      // blocks per grid

__global__ void matrixadd(int *a,int *b, int *c) {
       int col = threadIdx.x + blockDim.x * blockIdx.x;
       int row = threadIdx.y + blockDim.y * blockIdx.y;
       int index = row * N + col;
       if(col < N && row < N)
       c[index] = a[index]+b[index];
}

int main(void){
       int a[N],b[N],c[N];
       int *dev_a, *dev_b, *dev_c;

       cudaMalloc((void**)&dev_a,N * sizeof(int));
       cudaMalloc((void**)&dev_b,N * sizeof(int));
       cudaMalloc((void**)&dev_c,N * sizeof(int));

       for(inti=0;i<N;i++) {
              a[i] = i;
              b[i] = i*1;
       }

       cudaMemcpy(dev_a, a , N*sizeof(int),cudaMemcpyHostToDevice);
       cudaMemcpy(dev_b, b , N*sizeof(int),cudaMemcpyHostToDevice);
       cudaMemcpy(dev_c, c , N*sizeof(int),cudaMemcpyHostToDevice);

       matrixadd<<<B,T>>>(dev_a,dev_b,dev_c);

       cudaMemcpy(c,dev_c,N*sizeof(int),cudaMemcpyDeviceToHost);

       for(inti=0;i<N;i++) {
              printf("%d+%d=%d\n",a[i],b[i],c[i]);
       }

       cudaFree(dev_a);
       cudaFree(dev_b);
       cudaFree(dev_c);
       return 0;
}
```