EE/CS 451 Midterm 2

Fall 2019

Instructor: Viktor Prasanna

Friday, 11/8/2019

3:30 – 5:30pm, GFS 118

Problem #	Topic	Points	Score
1	Definitions	20	
2	Scalability	15	
3	Communication Primitives	15	
4	Communication Primitives	15	
5	CUDA/GPU	10	
6	Task Dependency Graph	10	
7	Data Decomposition	15	
Total		100	

Student Name:

Student USC-ID:

Name Initials: _____

Problem 1 ($10 \times 2 = 20$ points)

Define/explain the following terms:

1. Amdahl's law

Amdahl's law is used to find the achievable speedup when a program is run on a parallel machine. Given a problem, the total execution time consists of the execution time for the serial portion (S) and parallelizable portion (P). The overall speedup $= \frac{T_{Serial}}{T_{parallel}} = \frac{S+P}{S+P/f} = \frac{1}{S+P/f} \leq \frac{1}{S}$ (S denotes the serial portion; P denotes the parallelizable portion; f denotes the speedup factor.)

2. Grid and Block in a GPU

Threads are arranged as a grid of thread blocks in GPU. Different kernels can have different grid/block configuration. Threads from the same block have access to a shared memory and their execution can be synchronized. The grid of blocks and the thread blocks can be 1, 2, or 3-dimensional.

3. Critical path length

In a task-dependency graph, let us refer to the nodes with no incoming edges by start nodes and the nodes with no outgoing edges by finish nodes. The critical path is a longest path from a start node to a finish node (number of edges). The sum of the weights of nodes along this path is the critical path length.

4. Owner-computes rule

Each partition performs all the computations involving the data that it owns.

5. Gather primitive

Gather primitive is the dual of one-to-all personalized communication, in which a single node collects a unique message from each node.

6. SIMT

Single Instruction Multiple Thread (SIMT) is a parallel execution model, used in GPGPU platforms. SIMT is a virtualized execution of single instruction multiple data (SIMD) based on multi-threading. For example, in CUDA, threads belonging to the same Warp always execute the same instruction on different data in each clock cycle. The number of threads can be greater than the number of hardware units.

7. Streaming Multi-Processor in a GPU

A Streaming Multiprocessor is a compute unit of GPU which consists of several CUDA cores which share a single control unit and a shared memory.

8. All-to-all personalized communication primitive

In all-to-all personalized communication, each node sends a distinct message to every other node.

9. Block cyclic distribution

Block cyclic distribution partitions data into many more (αp) blocks than the number of processors such that $1 \leq \alpha \leq \frac{n}{p}$, where n is the problem size and p is the number of processors. Then the blocks are distributed in a wraparound fashion so that block b_i is assigned to $P_{i\%p}$ $(0 \leq i \leq \alpha p)$.

10. Task interaction graph

Task-interaction graph describes the pattern of interaction among tasks. The nodes in a task-interaction graph represent tasks and the edges connect tasks that interact with each other. The nodes and edges of a task-interaction graph are assigned weights proportional to the amount of computation a task performs and the amount of interaction that occurs along an edge, if this information is known. The edges in a task-interaction graph are usually undirected, but directed edges can be used to indicate the direction of flow of data, if it is unidirectional.

Name Initials: _____

Problem 2 (15 points)

Scalability:

Assume n processors are connected using a hypercube in the network model.

1. Design a parallel algorithm to find the sum of *n* numbers in $O(\log n)$ time. Assume that initially each processing element is assigned one of the numbers to be added. What is the speed-up? What is the efficiency? Use order notation for speedup and efficiency. (3 + 2 + 2 = 7 points)

Assuming each processor j, $0 \leq j \leq n-1$, stores result in a local variable, $\text{sum}(j)$. Initially, $\text{sum}(j)$ is initialized to the j^{th} input (available in processor j)

Algorithm 1 Adding n numbers using n processors

In processor j, $0 \leq j \leq n-1$

```
1: for i = 0 to log2 n − 1 do
2:    if j mod (2^i + 1) = 0 then
3:       sum(j) = sum(j) + sum(j + 2^i)
4:    end if
5: end for
```

$$\text{Speedup} = \frac{O(n)}{O(\log n)} = O(\frac{n}{\log n})$$

$$\text{Efficiency} = \text{Speedup}/n = O(\frac{1}{\log n})$$

2. Design an algorithm using $p < n$ processors. Assume that initially each processing element is assigned n/p numbers to be added. Specify the range of *n* such that efficiency is constant. (4 + 4 = 8 points)

Algorithm 2 Adding n numbers using p processors ($p < n$)

In processor j, $0 \leq j \leq p-1$

```
1: sum(j) = sum of n/p numbers that are allocated to processor j
2: for i = 0 to log2 p − 1 do
3:    if j mod (2^i + 1) = 0 then
4:       sum(j) = sum(j) + sum(j + 2^i)
5:    end if
6: end for
```

$$\text{Speedup} = \frac{O(n)}{O(\log p + \frac{n}{p})} = O(\frac{np}{n + p\log p})$$

$$\text{Efficiency} = \text{Speedup}/p = O(\frac{n}{n + p\log p})$$

$$\text{Efficiency is a constant} \Rightarrow O(\frac{n}{n + p\log p}) = O(1) \Rightarrow n = \Omega(p\log p)$$

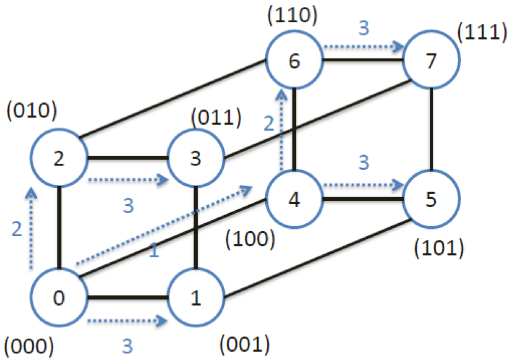
Name Initials: _____

Problem 3 (15 points)

Communication Primitives:

In the class, we discussed a hypercube One-to-all-broadcast algorithm in which the communication starts with the highest dimension (the dimension specified by the most significant bit of the binary representation of a node index) and proceeds along successively lower dimensions. The following figure illustrates it on an eight-node hypercube with node 0 as the source.

1. Write an iterative pseudo code for One-to-all broadcast on a p-node ($p = 2^k$) hypercube (network model) where the communications start with the lowest dimension. Assume the data to be broadcast is 1 unit. (6 points)




```
mask = 2^k-1
for i = 0 to logp-1
    mask = mask XOR 2^i
    if(my_id  AND mask) = 0 then
        if(my_id  AND 2^i) = 0 then
            send (msg,  my_id  XOR 2^i)
        else
            recv (msg, my_id  XOR 2^i)
    Endif
Endfor
```

2. What is the running time of the algorithm? You may use order notation. (4 points)

runtime is $O(\log p)$

3. Show how the algorithm can be simulated in a 1-D mesh of size p to run efficiently. What is the running time? (5 points)

Runtime is $1 + 2 + \cdots + \frac{p}{2} = O(p)$

Name Initials: _____

Problem 4 (15 points)

Communication Primitives:

Suppose we want to do Gather operation on a hypercube.

1. Write a recursive program to perform the Gather operation in a p-node ($p = 2^k$) hypercube with node 0 as the destination using the network model. Assume the data to be gathered is of size 1 unit in each node. (5 points)

```
gather (Node_destination, Node_last, node_no){
    if (node_no= 2) begin
        send (msg, Node_last, Node_0);
        combine_two_msgs;
    end else begin
        gather (0, Node_(node_no/2 - 1), node_no/2);        // gather the first half
        gather (Node_(node_no/2), Node_last, node_no/2);    // gather the second half
        send (msg, Node_(node_no/2), Node_0);
        combine_two_msgs;
    end
}
```

2. Derive an exact expression for the total number of steps (parallel recursive calls) and the total communication time. (5 points)

communication time = $1 + 2 + \cdots + \frac{p}{2} = p - 1$

3. If each node in the hypercube can use multiple ports to send and receive, what is the lower bound for the total communication time of Gather operation? What is the lower bound if each node uses single port communication? Explain. (5 points)

In a hypercube, a node can be connected through logp ports, hence the lower bound is $\frac{p-1}{\log p}$ when using multiple ports.The lower bound is $p-1$ when using single port.

Name Initials: _____

Problem 5 (10 points)

CUDA GPU:

Design a CUDA program to perform matrix multiplication $C = A \times B$. The size of each matrix is 1K × 1K. Each element is 1 byte. The matrices are initially stored in the global memory of GPU. The GPU has one streaming multiprocessor (SM) with 1K CUDA cores. Each CUDA core runs at 1GHz and can perform one floating point operation in each clock cycle. The peak bandwidth between the GPU and the global memory is 100 GB/s and the cache of the GPU has been disabled.

1. Design a simple CUDA program using straightforward vector inner product approach. Write the pseudo code for your CUDA program including both kernel function and host function. (3 points)
   Derive a lower bound on the execution time of your kernel function. Explain. (2 points)

```
Host_function(){

        //No need to copy A,B to gpu as they are already on the GPU

        Grid_dim=(1);
        Block_dim=(1024);

        Kernel_function<<< Grid_dim, Block_dim >>>( gpu_A, gpu_b, gpu_c);

        Mem_copy(C, gpu_c, DeviceToHost);
}

//Each thread is responsible for a single row of matrix C
Kernel_function(A, B, C) {
        my_id = threadIdx.x;

        For k = 0 to 1023
                Local_c = 0;
                For i = 0 to 1023
                        Local_c += A[my_id][i] X B[i][k]
                EndFor
                C[my_id][k] = Local_c;
        EndFor
}
```

Lower bound on the execution time is given by the data transfer time between global memory and the cuda cores for processing.

Input data A, B required by a single thread $= 1K \times 1K \times 2$. So, total input data transferred for all the threads $= 1K \times 1K \times 1K \times 2$.
Global memory bandwidth $= 100$ GB/s.
So, total input data transfer time $= \frac{1K \times 1K \times 1K \times 2}{100GB/s} = 0.02s$.

Output data C produced by a single thread $= 1K$ So, total output data produced by all the threads $= 1K \times 1K$
Total output data transfer time $= \frac{1K \times 1K}{100GB/s} \approx 1 \times 10^{-5}s$

So, total time $\approx 0.02001s$.

2. Assume the GPU has an on-chip buffer (shared memory) which can store 3 × 32 × 32 elements and has a peak access bandwidth of 1 TB/s.
   Write pseudo code for an optimized CUDA program using block matrix multiplication approach. Show the kernel function and the host function. (2 points)
   Derive lower bounds for 1) the computation time, 2) the local data access time in SM, and 3) the global data access time of your optimized kernel function. Explain. (3 points)

```
Host_function(){

//No need to copy A,B to gpu as they are already on the GPU

Grid_dim=(1);
Block_dim=(32 X 32);

Kernel_function<<< Grid_dim, Block_dim >>>( gpu_A, gpu_b, gpu_c);

Mem_copy(C, gpu_c, DeviceToHost);
}

//Each thread is responsible for a single element in each block of matrix C
Kernel_function(A, B, C) {
my_idx = threadIdx.x;
my_idy = threadIdx.y;

__shared__ A_s[32][32], B_s[32][32]

For iterx = 0 to 32 - 1
       For itery = 0 to 32 -1
              For z = 0 to 32 -1
                     A_s[my_idx][my_idy] = A[iterx*32 + my_idx][z*32 + my_idy]
                     B_s[my_idx][my_idy] = B[z*32 + my_idx][itery*32 + my_idy]
                     __syncthreads()
                     local_c = 0;
                     For j = 0 to 32 - 1
                            local_c += A_s[my_idx][j]*B_s[j][my_idy]
                     EndFor
                     C[iterx*32 + my_idx][itery*32 + my_idy] = local_c
              EndFor
       EndFor
EndFor

}
```

Total data transferred from Global Memory to shared memory: $32 \times 32 \times 32 \times 32 \times 32 \times 2 = 2^{26}$
Global memory bandwidth = 100 GB/s.
So, total input data transfer time = $\frac{2^{26}}{100GB/s} \approx 64 \times 10^{-5}s$
Total data transferred from Shared memory to cuda cores: $32 \times 32 \times 32 \times 32 \times 32 \times 2 = 2^{26}$
Shared memory bandwidth = 1 TB/s.
so, total data transfer time to cude cores: $= \approx 2 \times 10^{-8}s$
Total output data produced by all the threads = $32 \times 32 \times 32 \times 32 \times 32$
Total output data transfer time = $\frac{32 \times 32 \times 32 \times 32 \times 32}{100GB/s} \approx 32 \times 10^{-5}s$

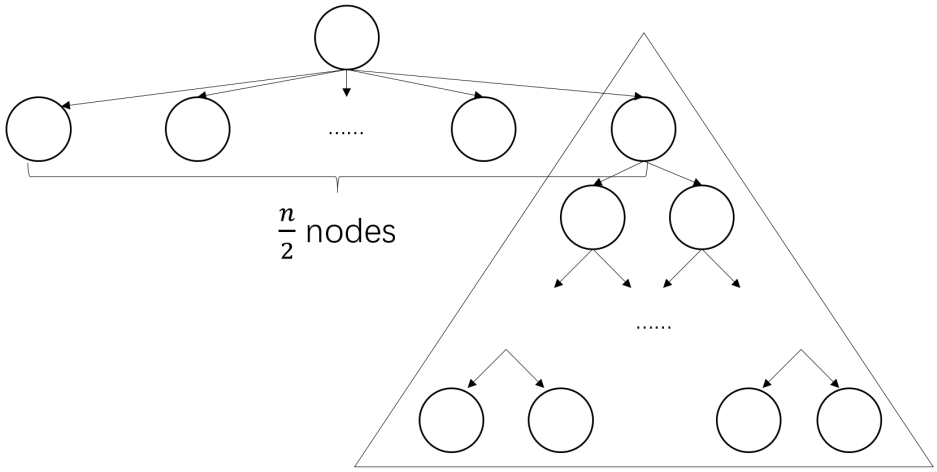
So, total time = 0.00096002

Name Initials: _____

Problem 6 (10 points)

Task Dependency Graph:

The task dependency graph shown below has n tasks ($n/2 = 2^k - 1$). Assume each task is of unit weight. Determine the following:



$\frac{n}{2}$ nodes in a full binary tree

1. Maximum degree of concurrency (4 points)

$$\frac{n}{2}$$

2. Critical path length (2 points)

$$\log\left(\frac{n}{2}+1\right)$$

3. The maximum achievable speedup over single processor when the number of processors $p = n$. (4 points)

$$\frac{n}{\log\left(\frac{n}{2}+1\right)}$$

Name Initials: _____

Problem 7 (15 points)

Data Decomposition:

Assume an $n \times n$ matrix is partitioned using block cyclic distribution with each block of size 1 and mapped to p processes (assume n and p are powers of 2, $p \leq n$). Assume row major order.

1. For n = 4, p = 2, draw a data partitioning diagram showing the assignment of the elements to processes. (3 points)

P_0	P_1	P_0	P_1
P_0	P_1	P_0	P_1
P_0	P_1	P_0	P_1
P_0	P_1	P_0	P_1

2. Define the mapping function showing how the elements are assigned to each process as a function of the index of the elements, *n* and *p*. (4 points)

   Mapping function:

   $\text{Matrix}\,(i,j) \rightarrow P_{(i \times n + j) \bmod p}$

   $0 \leq i < n,$

   $0 \leq j < n$

3. Consider $n \times n$ matrix multiplication $C = A \times B$. Assume $p = n$. Partition the input and output matrices among p processes using the approach above. Assume the owner of $C(i,j)$ computes $C(i,j)$. What is the total amount of data communication (among all the processes)? (8 points)

   Data layout:

P_0	P_1	...	...	P_{p-1}
P_0	P_1	...	...	P_{p-1}
P_0	P_1	...	...	P_{p-1}
P_0	P_1	...	...	P_{p-1}
P_0	P_1	...	...	P_{p-1}

   For P_0, it needs all the other elements of A except the first column, which means $(n-1)n$ elements.

   The same thing applies for the other processes. Hence, the total data of communication is

   $p(n-1)n$

Blank Sheet

Blank Sheet