SWE3021 Multicore Computing - Final Exam Fall 2019

Student ID	Name

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
For Instructor/TA only,									

Academic Honor Pledge

I affirm that I will not at any time be involved with cheating or plagiarism while enrolled as a student in Programming Language class at Sungkyunkwan University. I understand that violation of this code will result in penalties as severe as indefinite suspension from the university.

Your signature: _____

1. [Short Answers: 30pts] Briefly describe the following terms, i.e., what it is and what's the purpose of it. etc.

(a) NIS:
Authentication service in cluster

(b) NFS:
Cluster file system

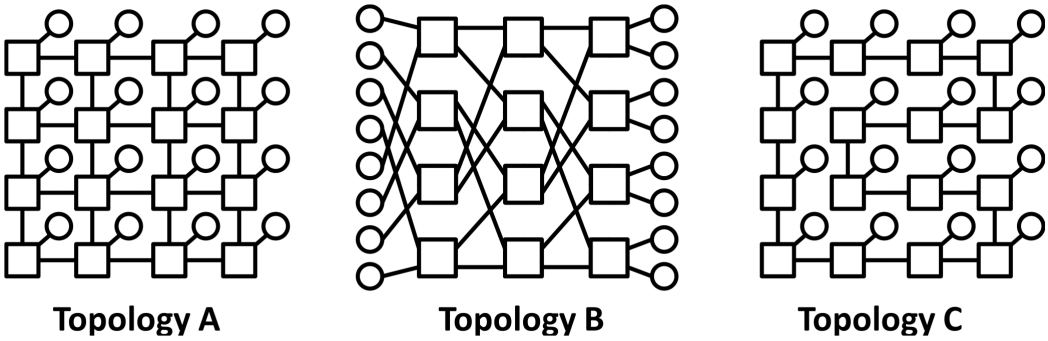
(c) QPI:
QuickPath Interconnect: interconnect for NUMA

(d) UVA:
Unfied Virtual Address: One address space for all CPU and GPU memory

(d) GPUDirect:
Direct path for data exchange between the GPU and a third-party peer device using standard features of PCI Express

(e) RDD:
Resilient Distributed Datasets: Cached data in Spark framework

2. [Interconnect: 15pts] The figure below shows four network topologies. Circles represent servers. Squares and lines represent routers and 1g Ethernet links respectively. What is the bisection bandwidth of each topology?



Topology A **Topology B** **Topology C**

Topology A: 4

Topology B: 8

Topology C: 2

3. [MPI: 10pts] Consider the following MPI program.

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char* argv[]) {
   int rank, sendbuf, recvbuf, numtasks;
   MPI_Status status;
   MPI_Request request;
   MPI_Init(&argc,&argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

   sendbuf = rank;

  if (rank) {
     MPI_Isend(&sendbuf, 1, MPI_INTEGER, 0, 32766, MPI_COMM_WORLD, &request);
     sendbuf = 4;
     MPI_Recv(&recvbuf, 1, MPI_INTEGER, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
  } else {
     MPI_Isend(&sendbuf, 1, MPI_INTEGER, 1, 32766, MPI_COMM_WORLD, &request);
     sendbuf = 8;
     MPI_Recv(&recvbuf, 1, MPI_INTEGER, 1, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
  }
  printf("me: %d, them: %d\n",rank, recvbuf);
  MPI_Finalize();
}
```

(a) Write all the possible outputs by process 0.
me: 0, them: 1
me: 0, them: 4

(b) Describe why multiple outputs are possible.
Non-blocking Isend without buffering may send the updated sendbuf 4 to the receiver.

4. [Graph: 10pts] Consider you are running parallel Floyd's algorithm using 2D block mapping on 16 MPI processors. How many one-to-one communications between processes will be be necessary? If a process broadcasts its data to K other processes, the number of communications is K. Justify your answer with illustrations of each step in the following figures.

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(a) Step 1

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(b) Step 2

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(c) Step 3

P00	P01	P02	P03
P10	P11	P12	P13
P20	P21	P22	P23
P30	P31	P32	P33

(d) Step 4

Correct answer: 96. (Please check out the lecture slides for the communication pattern.) Partial points were given for "72".

5. [CUDA: 15pts] Rewrite the following CUDA program to minimize warp divergence.

```
__global__ pairwise_sum(float *a) {
    int tid = threadIdx.x
    for(int stride = 1; stride<blockDim.x; stride*=2) {
        __syncthreads();
        if (tid%(2*stride)==0)
            a[tid] += a[tid+stride];
    }
}
```

Your answer:

```
__global__ pairwise_sum(float *a) {
    int t = threadIdx.x;

    for(int stride = blockDim.x; stride>1; stride>>=1)
    {
        __syncthreads();
        if (t<stride)
            partialSum[t] += partialSum[t+stride];
    }
}
```

6. [Concurrency: 15pts] Consider the following function that inserts a new object into a queue. If multiple threads call this function simultaneously, will a context switch at point (1), (2), or (3) corrupt the queue? Justify your answer.

```
void Enqueue(Object newobject)  {
     Node* newEntry = new Node(newobject);
     // (1)
     Node* oldtail = tail;
     // (2)
     tail = newEntry;
     // (3)
     oldtail->next = newEntry;
}
```

(1) [Yes / No] Why? No. Construction of a QueueEntry is a purely local operation (and does not touch shared state in any way).

(2) [Yes / No] Why? Yes. An intervening Enqueue() operation will move the shared variable "tail" (and enqueue another object). As a result, the subsequent "tail=newEntry" will overwrite the other entry.

(3) [Yes / No] Why? No. At this point in the execution, only the local thread will ever touch "oldtail.next" (since we have moved the tail). Thus, we can reconnect at will. People who worried that the linked list is "broken" until this operation can relax. The worse that will happen is that the list appears to be shorter than it actually is until execution of "oldtail.next=newEntry," at which point the new entry becomes available for subsequent dequeue.

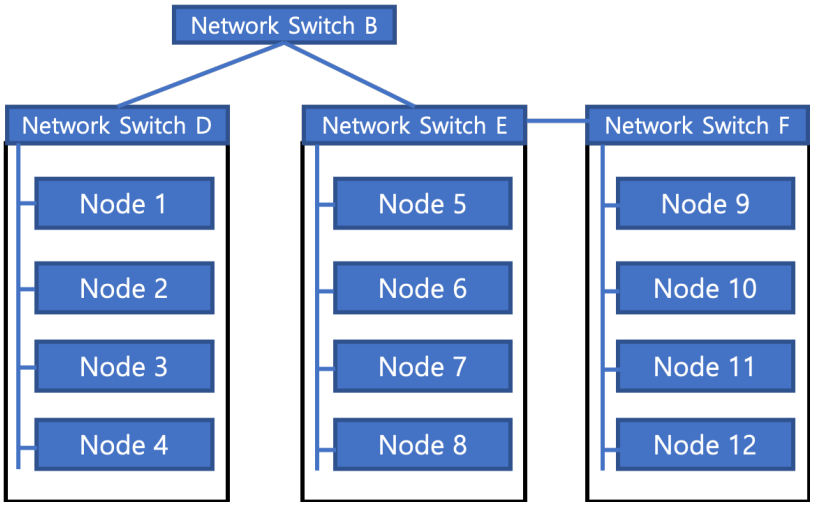
7. [Concurrency: 15pts] Implement a lock-free version of `Enqueue()` function using the atomic compare-and-swap function `CAS(old,expected_old,new)`, which returns `false` if `old` and `expected_old` differ and `true` if it succeeds replacing `old` with `new`. Hint: wrap a do-while around vulnerable parts of the code you identified in the previous problem.

```
    void Enqueue(Object newobject)  {
          Node* newEntry = new Node(newobject);
          // Insert code here


         Node* oldtail;
         do {
             oldtail = tail;
         } while {!CAS(tail,oldtail,newEntry);

         oldtail = newEntry;
}
```

8. [Hadoop: 15pts] Consider the following Hadoop cluster, which sets the replication factor and block size to 4 and 128 MB respectively. If your client running on Node 1 generates an output file of 128 MB, where the NameNode running on Node 4 should place replica blocks? Explain why each replica should be placed on the chosen node.



Replica 1:
Why?

Replica 2:
Why?

Replica 3:
Why?

Replica 4:
Why?

At least one node per rack.
One replica on the client node, i.e., Node 1.
Node 4 must not be chosen since it is the bottleneck.