15-418/618: Spring 2021 Exam #2

- You will prepare your answers using the answer spaces provided in this exam. Write your answers in the spaces provided; you may print and scan the exam or add your answers digitally.
- You will be given 48 hours to complete the exam, i.e., until 4pm EST on Friday 4/23. Submit your exam to Gradescope.
- If there is some reason you do not believe you will be able to make this deadline, you must inform us by Tuesday 4/20 and get an explicit waiver from us.
- You may access any course materials but, unless explicitly told otherwise, *do not access any other materials.*
- All requests for clarifications must be made via *private* Piazza posts, and all clarifications will be responded to via a *public* FAQ within 24 hrs. Do *not* ask for clarifications in public.
- The exam will be designed so that someone with complete preparation can complete it in two hours. We are allowing 48 hours only to provide more flexibility, and to account for the fact that students may be operating in different time zones.

Problem 1	22 points
Problem 2	13 points
Problem 3	19 points
Problem 4	12 points
Problem 5	14 points
Problem 6	20 points
Total	**100 points**

Sign the following pledge:

I do hereby swear that, in generating my answers to this exam, I made use of only course resources or others with explicit permission. I did not have any communication of any form with anyone other than the course instructors and teaching assistants about the contents of this exam.

____
Your signature

Problem 1: Multiple Choice (2pts each)

L13 Directory-Based Cache Coherence

Which of the following methods can help us reduce the storage overhead of directory-based cache coherence?

A. Decrease cache line size
B. Store limited pointers in the directory
C. Use sparse directories
D. Use multiple directory banks

L14 Memory Consistency

Which of the following memory consistency models allows the following program behavior? (assume `a, b` were initialized to `0`) (Select all that apply)

```
Thread1          Thread2
-------          -------
a = 3            b = 4
print(b)         print(a)
```

```
Program Output: 0 0
```

A. Sequential Consistency
B. Release Consistency
C. Total Store Order (TSO)
D. Partial Store Ordering (PSO)

L15 Interconnection Networks

Identify whether the following interconnects introduced in class are blocking or unblocking:

Torus:	blocking	non-blocking
Crossbar:	blocking	non-blocking
Hypercube:	blocking	non-blocking

L16 Implementing Synchronization

Select all statements that are true about a ticket lock:

A. Ticket locks provide fairness
B. Ticket locks need an atomic operation to release the lock
C. Ticket locks require transactional memory
D. Ticket locks require O(P) interconnect traffic per lock acquire/release with P threads

L18 Fine-Grained Synchronization

Select all statements that are true

A. Lock-free code is always faster than code with locks

B. Lock-free code requires a sequential consistency model.

C. Lock-free code doesn't need memory fences

D. Lock-free code cannot eliminate contention

L19 Transactional Memory

Suppose your program has a short, frequently executed critical section that is rarely contended. What is the advantage of transactional memory over lock-based synchronization?

L20 MPI, OpenMP, and Cilk Implementation

In the Cilk implementation shown in lecture, why does cilk_sync (potentially) create a continuation?

L21 Heterogeneous Parallelism

2. Which of the following is often used to prototype an ASIC?

    A. Apple Neural Engine™

    B. DSP

    C. FPGA

    D. GPU

L22&L23 Domain-Specific Languages & Frameworks

1. Which of the following statements is generally true of a domain-specific programming system:

    A. It increases the level of abstraction for writing programs.

    B. It is implemented as a new programming language.

    C. It can be used to solve any problem.

    D. It can optimize for a specific machine “under the hood”.

2. Why are domain-specific languages & frameworks important as architectural heterogeneity increases?

L24 Deep Neural Networks (DNNs)

In an uncompressed deep neural network, which kind of layer tends to dominate the memory requirements of the network?

A. Fully-connected layers.
B. Convolutional layers.
C. Pooling layers.
D. Softmax layers.

Why does training require more memory than inference?

Problem 2: Memory Consistency

```
// global variables and initial values
volatile bool var_0 = false
volatile bool var_1 = false;
volatile int turn = 0;

// P0                                      // P1
var_0 = true;                              var_1 = true;
turn = 1;                                  turn = 0;
while (turn == 1 && var_1);                while (turn == 0 && var_0);
// critical section                        // critical section
var_0 = false;                             var_1 = false
```

The code snippet above was written with the intent to synchronize the two processes P0 and P1. This code is correct on a sequentially consistent processor.

Question 2.1: Errors w/ relaxed consistency (4 pts)

Why will this code fail correctness on a multi-processor system that uses a relaxed memory consistency mode such as TSO? Explain your answer.

Question 2.2: Ensuring correctness (3 pts)

Insert **MFENCE** instructions in the code snippet above to ensure correct behavior on a multi-processor system that uses the Total Store Order (TSO) relaxed memory consistency model. Use as few **MFENCE** instructions as possible.

```
// global variables and initial values
volatile bool var_0 = false
volatile bool var_1 = false;
volatile int turn = 0;
```

```
// P0

var_0 = true;

turn = 1;

while (turn == 1 && var_1);

// critical section

var_0 = false;
```

```
// P1

var_1 = true;

turn = 0;

while (turn == 0 && var_0);

// critical section

var_1 = false;
```

Question 2.3: Relaxing further (3 pts)

Now suppose you are going to run this code on a system with the PSO consistency model. Does this change your number or placement of the fences? Why or why not?

Question 2.4: Spatial locality and consistency (3 pts)

Now assume that `var_0` and `var_1` are on the same cache line, does this change your answer to TSO or PSO models? Why or why not?

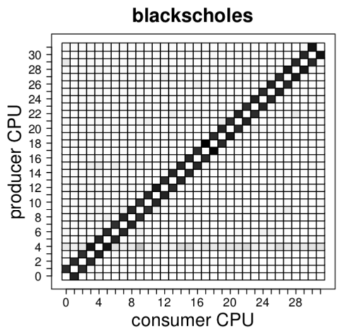
Problem 3: Interconnection Networks & Directory-based Cache Coherence

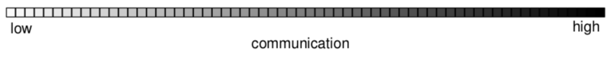
In this problem, we are going to analyze how different interconnection networks perform on a few multithreaded applications running on a multicore CPU using directory-based cache coherence.

We are going to do this based on communication patterns shown graphically below. These patterns are 2D grids (“heat maps”) which show the frequency of communication between pairs of cores in a multicore. The y-axis represents the core that is producing data (i.e., the “sender”), and the x-axis represents the core that is consuming data (i.e., the “receiver”). The shade of each cell shows how much a given producer-consumer pair communicate with each other: a dark shade means that the cores communicate a lot, whereas a light shade means they communicate little.

Your task is to understand the communication pattern depicted for each application and then reason about its *best-case* performance on different interconnection networks. Also assume that cores are numbered in the “natural” fashion: sequentially in a 1D network, and left-to-right, top-to-bottom in a 2D network. For example, in a bidirectional ring, core 0 connects to cores 1 and 31, core 1 connects to cores 0 and 2, etc.

Question 3.1: “Blackscholes” Application (8 pts)





Consider the provided communication pattern shown in the above graph for the “blackscholes” application. Compare the performance (latency) and cost (number of switches, links) of bus, ring and torus interconnects. Which interconnect achieves the best performance with the lowest cost (prioritize performance)?

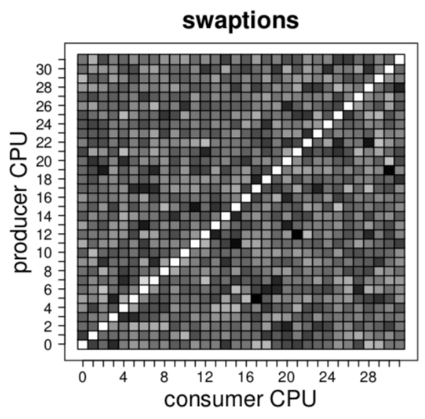
Bus:

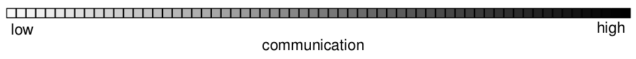
Ring:

Torus:

Best overall:

Question 3.2: “Swaptions” Application (8 pts)





Now consider another communication pattern shown in the above graph for the “swaptions” application. Compare the performance (latency) and cost (number of switches, links) of mesh, H-tree and crossbar. Which interconnect achieves the best performance with the lowest cost (prioritize performance)?

Mesh:

H-tree:

Crossbar:

Best overall:

Problem 3.3: Directory placement (3 pts)

In the prior questions you were asked to reason about best-case performance. For this problem, now consider that home directory nodes are typically placed at random throughout the interconnection network. (Equivalently, addresses are assigned a home directory “at random”, e.g., by hashing.) How does this fact change the *expected* performance for the two above applications on different networks? A short, qualitative answer suffices.

Problem 4: Vaccination Synchronization

Question 4.1: Lock implementations (6 pts)

You're a technical lead for a private health-care company issuing vaccines out to the public. You've been tasked with developing an online registration system that allows clients to sign up for a time-slot. Your website is in high demand, and can service many different clients at once in parallel, but only one client is allowed to sign up for each time-slot.

The website can only handle 100 people at a time, and anyone else who joins is redirected to another 'waiting list' site until a spot opens up and they are redirected back to the main registration site. You deploy your website on a web-service that provides **128 bytes of memory**. On this machine, **cache lines are 2 bytes**.

A. You decide to implement the waiting list using a spin-lock. After deploying the site, you receive some negative feedback from the users on the waiting list that some users seem to wait forever, while others get to the main site while having a minimal wait. What's the issue?

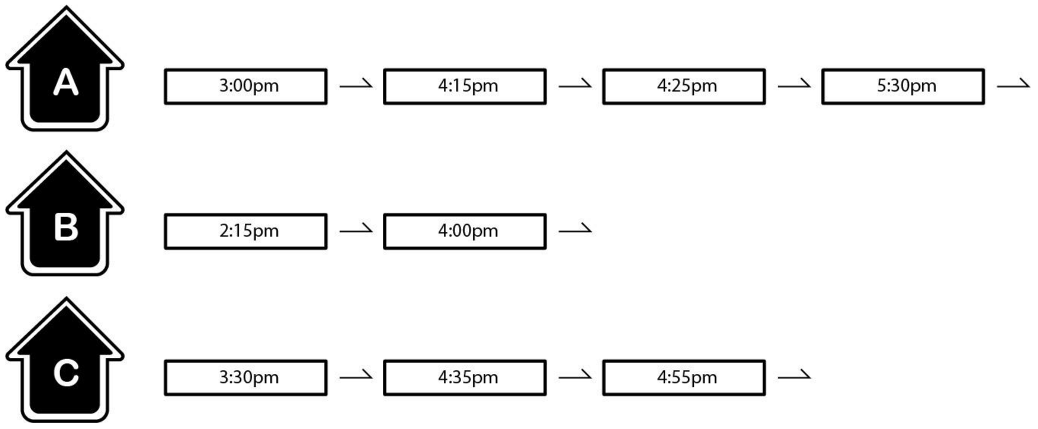
B. You decide to re-implement the waiting list using an array-based lock on the web server, where the memory is kept on the web server. What is a potential issue with this approach on this machine?

C. You instead want to use an approach that avoids the memory limitation imposed by the web-service, while still addressing the negative feedback in part 1. What is another approach we can use? How would we implement it?

Question 4.2: Synchronization (6 pts)

The appointment times for each vaccination clinic is represented as a linked list sorted by the times of the vaccination appointments. Your company just received a surplus of doses and are allowing users to register for whatever time they would like at whichever facility they prefer. You may assume each request is a unique time (i.e., you'll never get two requests for the same time, so there is always a unique sorting).

You want your system to be able to handle multiple requests at once while keeping the list sorted. If you receive a registration request for 4:18pm at clinic A, then your system will need to insert a node between the existing 4:15pm and 4:25pm nodes.



In order to handle concurrency issues, you implement the following lock mechanism:

1. Search for the two nodes in the requested clinic that fall directly before and after the requested time.
2. Send requests for both locks asynchronously.
3. Wait for both locks to be acquired.
4. Insert the new node before unlocking both nodes.

You reason that it is safe to traverse the list since the code never deletes any appointments. Consider two users scheduling 4:18pm and 4:21pm for clinic A.

D. When implementing the above strategy, these users report that their registration freezes. Why is that?

E. To fix the above problem, you instead send a request to lock the earlier time and wait for the lock before requesting to lock the later time. Registration doesn't freeze anymore, but one user reported that their appointment was not made, even though your system said it was successful. Why is that?

Problem 5: Programming Environments & Domain-Specific Frameworks

You are trying to find the answer to the question of life. Some researchers have posited that it is 42, but you are not convinced of this. To find your own solution, you attempt to run John Conway's Game of Life simulator for a very large number of iterations.

Conway's rules are very simple, but you wish to augment them to make it more "realistic". Conway's original game operates on a grid of cells, using the following summarized rules:

1. Any live cell with two or three live neighbours survives.
2. Any dead cell with three live neighbours becomes a live cell.
3. All other live cells die in the next generation. Similarly, all other dead cells stay dead.

You decide to augment this further by allowing each cell to make a "life decision" if they survived by rule 1, or they are "birthed" by rule 2. Specifically, a life decision takes a random amount of time depending on where they are in "Life" (row, column).

After writing a sequential implementation, you have improved its performance using OpenMP, as shown on the next page:

```
void life_blocked(int iterations, int **o_grid, int **n_grid) {
  for (int i = 0; i < iterations; ++i) {
    int num_r_blocks = (HEIGHT + B_SIZE - 1) / B_SIZE;
    int num_c_blocks = (WIDTH + B_SIZE - 1) / B_SIZE;

#pragma omp parallel default(none) \
    shared(o_grid, n_grid, num_r_blocks, num_c_blocks)
    {
#pragma omp for schedule(static)
      for (int j = 0; j < num_r_blocks; ++j) {
        for (int k = 0; k < num_c_blocks; ++k) {
          for (int jj = 0; jj < B_SIZE; ++jj) {
            for (int kk = 0; kk < B_SIZE; ++kk) {
              int r_idx = jj + j * B_SIZE;
              int c_idx = kk + k * B_SIZE;

              if (r_idx >= HEIGHT) continue;
              if (c_idx >= WIDTH) continue;

              int neighbors_count = num_neighbors(r_idx, c_idx, o_grid);

              if (neighbors_count == 2 || neighbors_count == 3) {
                n_grid[r_idx][c_idx] = ALIVE;
                make_life_decision(r_idx, c_idx);
              } else
                n_grid[r_idx][c_idx] = DEAD;
            }
          }
        }
      }

    }

    int count = 0;
    /*
     * SOLUTION CODE
     * GOES HERE
     */
    swap(&o_grid, &n_grid);
  }
}
```

Question 5.1: Performance debugging (5 pts)

While improved, you still find the performance lacking, and have timed each individual worker that OpenMP launches, and obtain the following results:

```
Time taken by thread 0 is 0.971275
Time taken by thread 1 is 3.489316
Time taken by thread 2 is 5.731943
Time taken by thread 4 is 6.825055
Time taken by thread 3 is 7.496503
Time taken by thread 5 is 8.995236
Time taken by thread 6 is 11.102605
Time taken by thread 7 is 12.958080
```

In a sentence, describe why the performance disappoints. In addition, describe what *single line* you would modify in the existing code to solve this problem.

Question 5.2: Counting live cells (5 pts)

After each iteration you want to obtain a total count of all cells that are alive. Please write a short amount of code describing how you would do so using an OpenMP pragma (approximate location would be in “SOLUTION CODE GOES HERE”), above:

Question 5.3: Re-implementing the Game of Life (4 pts)

Of the domain-specific languages we have covered in class (Liszt, Halide, Graphlab), which would be most applicable to this problem? Explain why you prefer this language to the others.

Problem 6: Heterogeneous Parallelism & DNNs

You are training a VGG16 model, but you feel that it is too slow on your personal workstation, so you decide to design a system of your own. Your colleagues from ECE have already designed the following three special processors for you, which will be the building blocks of your system:

- **CPU-Lean:** this CPU was designed for area efficiency and takes one unit of area;
- **CPU-Fast:** this CPU was designed for speed. It is twice as fast as the CPU-Lean design, but it takes four units of area;
- **GPU-Conv:** this GPU was only designed for speed up convolutional computation. It is 8 times as fast as the CPU-Lean design on convolutional layers. But it is only as fast as the CPU-Lean design on pooling layers, and it cannot be used for the other parts of the program. It also takes four units of area.

You are considering the following four designs for machines and want a quantitative way of determining which one is better:

- **Lean-Only Machine:** contains 12 CPU-Lean cores;
- **Mixed Machine 1:** contains 8 CPU-Lean cores + 1 CPU-Fast core + 0 GPU.
- **Mixed Machine 2:** contains 4 CPU-Lean cores + 1 CPU-Fast cores + 1 GPUs.
- **Mixed Machine 3:** contains 4 CPU-Lean cores + 0 CPU-Fast cores + 2 GPUs.

Please assume the following:

- The benchmark you are working on is a CNN program deployed on edge devices. The benchmark's computation consists of sequential and parallel parts.
- The parallel part (which can be run on CPU and GPU simultaneously) consists of pooling layers and convolutional layers.
- The parallel portion of the benchmark will experience linear speedup when it runs on multiple CPUs/GPUs (i.e., there are no inefficiencies while running in parallel).

Consider the following parameters when answering questions below:

- t_s: Time to run the sequential region on one CPU-Lean core.
- t_p: Time to run the pooling layers on one CPU-Lean core.
- t_c: Time to run the convolutional layers on one CPU-Lean core.
- N_L (the number of CPU-Lean cores)
- N_F (the number of CPU-Fast cores)
- N_G (the number of GPUs)

Write your answers as expressions using these parameters.

Question 6.1: Sequential part (3 pts)

What is the execution time for the sequential part on a machine with N_L CPU-Lean cores, N_F CPU-Fast cores, and N_G GPU cores?

Question 6.2: Parallel part (3 pts)

What is the execution time for the parallel part (pooling and convolutional layers) with N_L CPU-Lean cores, N_F CPU-Fast cores, and N_G GPU cores?

Question 6.3: Speedup (3 pts)

Using your results for the last two questions, give an equation (or set of equations) for the overall speedup on the CNN:

Question 6.4: Comparing systems (5 pts)

Using your answers above, calculate the minimum execution time (in seconds) for several CNNs using the values shown in the table on the four different machine configurations. *(You may write a short program or spreadsheet to perform the computations, if desired.)*

Benchmark Time (secs) =

t_s	t_p	t_c	Lean-only machine (P_L = 12, P_F = 0, P_G = 0)	Mixed machine 1 (P_L = 8, P_F = 1, P_G = 0)	Mixed machine 2 (P_L = 4, P_F = 1, P_G = 1)	Mixed machine 3 (P_L = 4, P_F = 0, P_G = 2)
50	20	30				
50	10	40				
10	20	70				
10	10	80				

Question 6.5: Analysis and takeaways (6 pts)

Comparing your results above for different CNNs, what would you say about how different architectures perform on a variety of different applications? Specifically, discuss how architectural heterogeneity affects performance, and how architectural specialization affects performance.