CS3210 Parallel Computing: Exam Compilation

Combined Past Papers (2019 - 2024)

1 Final Assessment: AY2022/23 Semester 1

Question 1: Parallel Performance

Consider a 5-stage pipeline (IF, ID, OF, IE, OS).

Q1.a [6 marks]

Calculate average instruction throughput (IPC) for the loop body. Show at least two iterations. Assume no forwarding.

Your Answer:

Since the IF stage only happens after the OS stage and each instruction depends on the prebious, there will only be 1/5 instruction per cycle. **Total instructions is 4 * 50000 = 2000000. A single loop takes 20 cycles, but after that each one takes 16 because line8 can pipeline with line 5. So in general it is 0.25.**

Q1.b [2 marks]

Modify the program (C-like pseudo-code) to achieve peak instruction throughput.

Your Answer:

b[i] = 4 + 9 * a[i];. This reduces the numbers of stores and loads to only 1 and reduces dependencies between instructions by only having one instruction (no this doesnt change anything lol) **Unroll the loop to interleave 4 other instructions so that it is now pipelines, or use strided execution with startup and cooldown.**

Q1.c [6 marks]

If the loop is parallelized with OpenMP `#pragma omp parallel for`, what is the smallest number of threads needed on a single core (SMT) to obtain peak throughput?

Your Answer:

Hide X cycle stall by interleaving X other threads which are independnt so processor needs to support X+1 threads in total.

Question 3: SortN in Parallel (MergeSort)

Q3.a [2 marks]

Which parts of the MergeSort code (recursive calls vs merge) would you parallelize? Justify.

Your Answer:

Each recursive call will create a new thread to operate on its half of the array. When merging, one thread will die. **Possible but difficult to parallelize merge.**

Q3.b [6 marks]

What are two problems observed when trying to parallelize the sequential implementation?

Your Answer:

1. Assuming a large enough array, there will eventually be 1 thread created for each element. This results in a massive overhead of creation and destruction of these threads **Task granularity is too small** 2. **Very few tasks in beginnering at end Merge is not parallel, temp array can be shared producing race conditions,**

Q3.c [6 marks]

How would you implement this parallelization using OpenMP? (Refer to line numbers or pseudo-code).

Your Answer:

pragma omp prallel sections

Question 4: SortN in MPI

Q4.a [6 marks]

Sketch an MPI approach dividing the array into chunks, sorting, and merging.

Your Answer:

Split the thing two, send to children. **Create and populate the array, divide the array into chunks, send subarrays to each process, and then collect eveyrthing back**

Q4.b [2 marks]

What parallel pattern did you use?

Your Answer:

Master worker

Q4.c [6 marks]

What is a main performance issue particular to this MPI implementation? Explain using theoretical concepts.

Your Answer:

Amdahls law. The main part still has to be serial, so the maximum speedup even with infinite threads is 2. **But with the large communication overhead**

Question 5: Bitonic Sort in CUDA

Q5.a [2 marks]

Merge sort is $O(N \log N)$. Bitonic is $O(N \log^2 N)$. Why and when would you choose Bitonic sort?

Your Answer:

Bitonic sort is more parallelziable. **Because the number of tasks can be easily scaled as opposed to mergesort**

Q5.b [2 marks]

Which algorithm performs better in CUDA? Justify.

Your Answer:

Bitonic. **Work is oblivious to the input data, and each thread in each step has the same amount of work** Merge has very divergent **also in Bitonic so not a reason** and recursive behavior that wont work on CUDA that ?

Q5.c [6 marks]

Describe tasks done by each CUDA thread for Bitonic sort (Kernel pseudo-code, grid/block dimensions, memory type).

Your Answer:

Need multiple blocks. Split data into chunks? idk hope it doesnt come out lol. For an array of length N = 2$^k, weneed1+2+3+..+kroundsofsorting(totalk(k+1)/2rounds, dividedintokstages, asshowninFigure3).Thekerneliscalledk(k+1)/2.Eachroundcallsakernelfunction, whichcompletesN/2compare-exchangeoperations.Threadsaregroupedinan1Dblock, numberofthreads=N/2.Useglobalmemory; itisnotclearthatsharedmemorywouldbringanyadvantageinthisapproach(astheker$

Q5.d [2 marks]

If $N \geq 4096$, give one reason why CUDA Bitonic might be inefficient and how to fix it.

Your Answer:

2 Final Assessment: AY2021/22 Semester 1

Q1. Heterogeneity and Performance

Q1.A.i [4 marks]

Game Entity update loop. Thread 0 updates positions; Thread 1 updates states. No synchronization. Why do you NOT obtain a 2x speedup on a dual-core cache-coherent system?

Your Answer:

Both lines 9 and 15 are updating on the same cache line, leading to serialized access since it is an invalidation based cache coherence **false sharing. Add assumption about similar processing speeds. Rmb to reference lines of code**

Q1.A.ii [4 marks]

Modify the code to correct the performance problem without changing the computation logic.

Your Answer:

Change liens 1 to 5 to have two vectors instead. This will result in the same amount of memory used, but removing the issue of cache line contention. **Also can make the second loop count from back**

Q1.B [12 marks] Wimpy vs Brawny Cores

- **i.** Wimpy-only chip ($N_{brawny} = 0$). Equation for execution time.
- **ii.** Heterogeneous chip. Equation for execution time.
- **iii.** Calculate most energy efficient design: (a) 20 Wimpy, (b) 5 Brawny, (c) 4 Wimpy + 4 Brawny. $f = 90\%$.

Your Answer:

- i. $200(1-f)+200/N_{wimpy}(f)$. The serialized portion will take the sequntial fraction multipled by the time it takes for a single Wimpy core to run the whole program, while the parallel part will have speedup by a factor of N_{wimpy}
- ii. $100(1-f)+200/(N_{wimpy}+2N_{brawny})(f)$. Use the brawny core for the sequential portion to make it faster. Since each brawny is twice as strong as a wimpy, it will do twice the work of a single wimpy core. Convert all to wimpy cores and it will be the same as part i
- iii. Wimpy with 20 cores will use $200\cdot 0.1+200\cdot 0.9\cdot 20 = 29$ seconds, and uses 20 units of power, with a ratio of 0.68. Branwy with 5 will use $100 \cdot 0.1 + 100 \cdot 0.9/5 = 28$ seconds, also using 20 unitd of power so a ratio of 0.714. THe mixed will use $100 \cdot 0.1 + 200 \cdot 0.9/(12) = 25$ seconds, using 20 units of power again so ratio of 0.8. The heterogenous is the most power efficient. **energy calculation is completely wrong. It uses that amount of power per second, so i uses 200, ii uses 400, and iii uses 340. Mention that wimpy only is the most energy efficient, but the hetero achievs the fastest time while still using less energy than branwy.**

Q1.C

Your Answer:

ii. 17 seconds, critical path of E, G, D iii. Total time: 34 seconds, speedup of 34/17 = 2.

Q2. Distributed Task Scheduling

Q2.a [6 marks]

Design a master-worker parallel algorithm (MPI-like) to solve distributed task scheduling (P cores, N tasks, $P << N$).

Your Answer:

Queue<Task> tasks; while true while task in tasks MPI$_{s}endatasktoanyfreeworker$**NO. All available tasks are scattered to all workers**$MPIrecvnewt$
Worker: receives tasks from master, compute, and sends new tasks back to master

Q2.c [5 marks]

Data or task paralleism

Your Answer:

Task,

Q2.c [5 marks]

Propose a better programming pattern than master-worker. Explain why.

Your Answer:

Task pool. Instead of manually implementing a task pool with a single point of contention (the master), **tasks are added into the pool as they are created, using producer consumer to synchronize in getting tasks. Metric such as resource utilization or idle time. Compare with master worker?**

Q2.d [5 marks]

What interconnection netwokr

Your Answer:

Use a tree. This makes it such that the furthest away is only log n, and the root node can send half the tasks to each of its children to reduce contention. **For a. specifically, star is the best.**

Q2.e [5 marks]

Computation vs Memory

Your Answer:

Idk. Since CPU and IO are diff try to allocate two diff types of jobs to one machine. **Since all workers wait for all to finish before executing, there will be much idle time when waiting for different lengths to finish.**

Q2.f [5 marks]

Use a Ring Topology. Explain solution, communication type, deadlock avoidance, and data transferred.

Your Answer:

Communication is done with neighbors using MPI send and recv calls. The calls will be non blocking and asynchronous as the node itself may have a queue of tasks waiting to be performed. By having it as such, the latency of messages can be hidden as it is computing something while receiving the tasks or sending the tasks to its neighbours. As such, deadlocks are avoided as none of the nodes block. **Wrong. It will receive tasks from worker on the left, take some tasks, add tasks, and then pass the list clockwise. Avoid deadlock also by using odd even send receive.**

Q2.g [5 marks]

Termination

Your Answer:

For task a, when the master sees that the task pool is empty and it has received a FIN (network terminology) from all its workers where no workers add any new tasks, then it has finished and can terminate. This is not possible in f, as there is not central command where one node sees all tasks. A possible solution is to periodically do a sweep (in clockwise of anticlockwise) of the nodes. It will pass a token that is initially true, and will change to false if any nodes still processing. When the token returns to the first node, if it is still true the program can terminate.

Q3. Memory Model

Code snippets executing on P1, P2, P3, P4 involving variables A, B, C, X, T.

Q3.a [2 marks]

Give an execution scenario showing failure under sequential consistency (where Print A != Print B).

Your Answer:

3>10>11>1>2 (prints A=1)>7>8>4>5>6 (prints B=0 as 9 has not been executed)

Q3.b [2 marks]

Modify code (swap statements) to ensure correctness under sequential consistency.

Your Answer:

Swap 8 and 9. This guarantees that A = 1 and B = A = 1 before the print to B is unlocked.

Q3.c [2 marks]

How it fails under TSO, PC, and PSO

Your Answer:

- TSO: Code will not fail
- PC: 9>6>7>1
- PSO: Reorder 8 and 9 back.

Q3.d [2 marks]

Modify code (swap statements) to ensure correctness under TSO.

Your Answer:

Impossible.

Q4 [2 marks]

Q4a [2 marks]

Your Answer:

False. RC models require increased synchronization but do this to hide hardware latencies. The sycnhronization is not unnecessary **since it is needed regardless, while RC increases performance**

Q4b [2 marks]

Your Answer:

False. Cache coherence is the property that when multiple caches refer to the same entity, they will all have the same values.

Q4c [2 marks]

Your Answer:

False. Some algorithms and paradigms are designed specifically for distributed or parallel systems and will not have a serial implementation. **True. Both points are correct**

Q4d [2 marks]

Your Answer:

Agree. Increased awareness of how to design algorithms and how time complexity analysis of algorithms does not paint the picture especially since memory is the bottleneck now. 2.

3 Final Assessment: AY2020/21 Semester 1

Part A: General Questions

Q1 Threads share memory space

Your Answer:

False. Threads by the same process share memory space. Two processes have diff memory space, and thus each thread has its own memory space within each process. This is assuming that shared memory is not explicitly created.

Q2 Hypercube vs CCC

Your Answer:

A hypercube is easier to construct. Hypercube hardware overload stays constant as nodes increase. **CCC is easier to construct as hypercube will have up to N connections while ccc will only have 3 (hardware overload stays constant)**

Q3 Cloud or Data Centre

Your Answer:

Since a cloud service is a type of data centre, it can be classified as both. With the semantics that the NUS cluster provides the service for us to store files, and the service to run our assignments, it seems to be more of a cloud service. Technically, we also pay school fees for the use of this cloud service. **None: No special data centre setup is in place. A data centre is a dedicated physical facility with speical power delivery, centralized cooling, security**

Q4 Cloud or data

Your Answer:

For the same reasons as above, a cloud center. **Data center, with justification. A bunch of GPUs so it is a data center, but it is the cluster itself.**

Q5 Matrix Multiplication

a. **[4 marks]** Two issues with OpenMP implementation for very large matrices fitting in memory.

Your Answer:

1. Cache thrashing, as one program is unable to fully store the matrix. The program would have to continually load and unload at least one of the matrices even though the same data is being used.

2. **Due to accessing matrix B in row major order, there will be a lot of cache misses.**

b. **[4 marks]** How to modify OpenMP program to MPI (distribute run on multiple machines).

Your Answer:

1. Assume that in Open MP, the second matrix is always in memoryy and the first matrix is distributed among threads. For MPI, **use communication instead of accessing shared memory, and divide matrices into chunks that fit in memory for each worker**

c. **[4 marks]** Which performs better (OpenMP vs MPI)? Give two reasons.

Your Answer:

MPI. 1. Increasing the amount of nodes will increase the total memory which will remove cache thrashing that OpenMP will have. 2. The introduced communication delay may be similar or not as significant as the managing of threads and processes in OpenMP **and page faults. No synchronization needed after sending data. MPI is not interrupted by IO (same as 1). MPI can always add more cores while Open MP limited by performance of machine used.**

Q6 Loops

a. **[4 marks]**Appr 1 or 2

Your Answer:

Implementation 2 is better assuming that there are enough resources to run all tasks in parallel and that the overhead of managing so many tasks is not that high. This is because the main writing contention of the same cache line will occur in M3[j][i] in both approach i and ii. **Too many tasks in ii. Too granular**

b. **[4 marks]** Which can be used.

Your Answer:

Neither. Since each iteration depends on the previously calculated value of i, this task cannot be parallelized. **Data dependency at line 13 with previous iteration of loop**

c. **[4 marks]** Which can be used.

Your Answer:

Approach 2 cannot be used. This will cause V2[i] to be a variable updated by many threads but not synchronized **leading to a data race**, although it is meant to be an accumulator for V1[j] * M3[i][j].

Part B: Performance

Q7 Amdahl

Your Answer:

p = 10. 5% is sequential. A speedup of 5 only requires 5 cores. **5** = **1**$\frac{}{0.05+\frac{0.95}{x}meaningxis6.33so7isrequired.}$

Q8 Var size fixed speedup

Your Answer:

Yes. Since $S = p - f(p-1)$, as size increases, f approaches 0, and the speedup becomes the same as p.

Q9

Your Answer:

a. CPI of P1 on CPUI M1: 2.55 b. CPU user time: 51000 cycles and 3.3 * 10 pow 9 cycles per second = 1.54 * 10 pow neg 5 seconds

Q10: Summation Array

Comparing `sum_sequential`, `sum_omp_critical`, `sum_omp_atomic`, and `sum_omp_reduction`.

a. **[3 marks]** Compute speedup for each based on Table 7 (Seq: 3.7ms, Crit: 48.2ms, Atom: 26.7ms, Red: 1.3ms).

Your Answer:

- 1
- 0.0767
- 0.138
- 2.84

b. **[2 marks]** Explain performance differences.

Your Answer:

Treating the sequential as the base, we find that critical and atomic are the same codes (still execute sequentially) but have the downside of introducing overhead in creation, destruction, and management of threads. Atomic may be faster than critical as the use of locks in the critical selection has a higher overhead than simply implementing it as an atomic add. However, these all end up being slower than sequential as there is contention on the sum variable, leading to sequential performance with all the downsides of parallel computing. Reduction removes the contention on the sum variable, accounting for the speedup

c. Two other performance metrics

Your Answer:

1. Memory access patterns. One can measure how often all regions of memory are accessed. The first three implementations will have one high spot, while the reduciton algorithm will have it spread over more spots **checking call stack using perf** 2. Number of instructions. The sequential algorithm will have the least amount of instructions, explaining why it is faster than the 2nd and 3rd which do not take advantage of multiple cores. **Or instructions per cycle 3. Time spent waiting: the only waiting happens for lock contention Checking cache misses of page faults**

d. Diff between seq and red

Your Answer:

reduction has much less contention onthe sum variable, and is able to exploit multiple PUs (at least 2) to obtain a faster execution time

Part C: Atmospheric Model

Atmospheric model on $N \times 2N$ grid. Stencil computation (8 neighbors).

Q11 [6 marks]

Your Answer:

Distribute by data. Split the array into blocks of 5 by 5, as this would mean that for middle node, no communication is required, for middle nodes, only a quarter communication is required, and half for edge nodes. Each node can be further parallelized by calculating the middle node while collecting the values of the four neighbouring regions, since calculation is the bottleneck, followed by the partial calculation of remaining nodes while waiting for the communication. This hides the latency of the communication while also ensuring that each node is calculating something at any point in time.**Equally load the CPUs**

Q12 [6 marks]

Your Answer:

Data. **Checkerboard pattern**

Q13 [6 marks]

Your Answer:

Mesh. This ensures that nodes it needs to communicate with is always one hop away.

Q14 [6 marks]

Sketch MPI pseudocode. Data at Rank 0 initially.

1. Data/Task distribution.

2. Process execution.

3. Communication steps.

Your Answer:

- MPI Scatter the regions
- Create data for top, left, right, then bottom
- MPI Isend to four neighbours
- MPI Irecv from four neighbours
- Start calculating for central node
- Start calculating for the middle nodes, (starting from bottom as that will be received first)
- Start calculating for edge nodes
- Once done, can send next set of 10 nodes while calculating new values for opposite section

Q14 [6 marks]

Your Answer:

Now, only 14* 3 values need to be shared to only two neighbours. **Memory for 4P units need to ensure no page faults, and ensure that 4 processing units are utilized even if not enoygh memory**

Q15 [6 marks]

- i: Reduce grid size by increasing N (as in making it more resolution
- ii: Compute localized model focusing on one region of earth
- Choose most accurate prediction after running many

Your Answer:

- a. Use 2 and 3, and stop 3 when u have a time deadline? Cannot use i
- b. 1 and 3, and stop 3 when accuracy level is reached. Increase resolution for better accruacy although it is much longer . Cannot use 2 as it is for the whole earth

4 Final Assessment: AY2019/20 Semester 1

Q1. General Parallelism

a. **[8 marks]** Advantages/Disadvantages of threads in OpenMP/CUDA.

Your Answer:

Advantages: Allow the sharing of process address space **while still having private registers** for similar programs to prevent costly setup of new processes. Also has faster creation **and destruction** and switching between different threads as compared to processes.

Disadvantages: A crashing or blocking thread will crash and block all other threads. Threads require more explicit synchronization to handle.
Threads in CUDA are warped and must have the same control flow. Threads in CUDA hide latency by always having a readywarp. Threads in OpenMP maximizes instruction throughput with Simultaneous multi-threading and hyperthreading. Threads still have creation and management overhead in OpenMP

b. **[4 marks]** Energy efficiency in heterogeneous platforms.

Your Answer:

1. The computer will use a less powerful but higher efficiency chip to carry out background and/or less intensive tasks. For example, a compute cluster may use its full capacity when training a model, but use a lower power chip to send periodic reports and measurements. **ARM has big cores and little cores, which works for smaller tasks as big cores require more power even at lower performance levels**

2. **NVIDIA Grace Hopper reduces the interconnect time? But doesnt really answer the question.**

c. **[4 marks]** Data Centers vs Cloud Computing differences.

Your Answer:

1. A data centre is usually private to a single entity, while cloud computing is compute power leased out by one company to other companoes, and may have services included with it. **Because cloud computing is a data center with a layer of software over it, and offers services at different levels (platform, software, infrastructure) while a data center is only for the infrastructure**

2. **Cloud services are usually a pay per use model, while data centers are one large investment in the beginning**

d. **[4 marks]** Task granularity on program performance.

Your Answer:

1. A higher granularity may result in higher dependencies between more tasks, possibly decreasing performance

2. A higher granularity may result in lower cache coherence, leading to decreased performance
3. A larger task granularity may result in not enough tasks to run on all cores, differences in runtime in each task making parellism difficult
4. A smaller granulairty introduces a lot of parallelism overhead

Q2. Parallelism and Performance

Q2.a [3 marks]

Intel Xeon Phi Mesh Network (YX routing). Message 1: Node 0 to 14. Message 2: Node 11 to 13. Contention analysis.
[Image of mesh network topology]

Your Answer:

Disagree. On the 4th clock cycle, there is one full packet at nodes 5 and 12. On the 8th clock cycle, there is a full packet at nodes 5, 10, 12, and 13. On the 12th clock cycle, all messages from node 11 have reached node 13, and there is no possible contention from the messages from node 0.

Q2.b Task Dependency Graph

(Graph with nodes A-H provided in context).

- **i.** Plot degree of concurrency.
- **ii.** Length of critical path.
- **iii.** Max speedup with infinite resources.

Your Answer:

i. On notebook. ii. Critical path: 16. **E G H** iii. Max speedup with infinite resources: Total $= 27$. speedup $= \frac{27}{16}$

Q2.c [4 marks]

A program spends 20% of its execution time within inherently sequential code. What is the maximum speedup achievable by a parallel version of the program? Briefly explain what law applies best for this scenario.

Your Answer:

Amdahls law. The maximum speedup is by a factor of 5.

Q2.d [2 marks]

An application running on 10 processors spends 3% of its time in serial code. What is the maximum speedup for this application? Briefly explain what law applies best for this scenario.

Your Answer:

Gustafsons law. Assume that data problem size is very large. Then speedup $= 10 - 0.03(9) = 9.73$
$S = (10 * T + 0.03T)/(T + 0.03T) = \frac{p}{1+f} + \frac{p+f}{1+f} = 10.03/1.03 = 9.737$

Q3. TOP10 Problem

Find $MAX(A)$ and all values $> 0.9 \times MAX(A)$.

Q3.a [4 marks]

Shared memory (OpenMP) algorithm design.

Your Answer:

- There are P processing units
- Each PU obtains N/P parts of the array
- Each PU calculates the local maximum in the array
- Collate to find the global maximum, and return it back to each PU
- Each PU runs through the array and returns all values that are larger than 0.9 * the MAX(A)

Q3.b [2 marks]

What type of parallelism (data or task) does TOP10 problem entail for your solution from point a.? Explain your choice.

Your Answer:

Data parallism. The data is being split among the PUs, resulting in a theoretical speedup by P. There are only 2 tasks (1. finding the maximum, and 2. finding the top 10), and task 2 is dependent on task 1.

Q3.c [4 marks]

You are now solving TOP10 problem for the distributed memory machine. Each core can hold N/P. Which data distribution pattern would you use? Present the data distribution for array A.

Your Answer:

Use block distribution. PU 1 will get the first N/P elements, PU 2 will get the second, and so on. This exploits spatial locality in the cache during sending and receiving of messages, since each PU has their own memory and will only need to operate in contiguous subchunks of the array.

Q3.d [5 marks]

Distributed memory (MPI) implementation using collective operations (Reduce).

Your Answer:

- Data will be distributed
- Each participant process will find the local maximum
- One process will collate and find the global maximum and return it to particpants
- Each participant process will then find the TOP10 and return this array of integers
- MPI Scatter the data so each process obtains N/P data
- Each process will sort this array, and place the last element into a send buffer
- Initialize a receive buffer
- Use MPI Allreduce with a maximum function to return the global maximum to each process
- Each process calculates 0.9 * Global Maximum
- Each process uses binary search to find the elements in its list that should be in the TOP 10, and places it in a send buffer
- MPI Gather to gather all responses into a receive buffer.

Q3.e [5 marks]

Distributed memory WITHOUT collective operations (No MPI_Reduce). Sketch the task dependency graph for solving TOP10 problem in MPI.

Your Answer:

On paper.

- Distribute data: The machine with data distributes the data to all particpating processes. This is done with scatter, employing a tree interconnection network to better distribute data instead of a master slave
- Each PU now has N/P data. Each PU iterates through the whole array to find the local max
- Using the same scatter pattern but in reverse, each leaf sends the local max to the node, and the node will send the local maximum of itself and its two children to its parent until the root is reached
- The root will the broadcast the global maximum using the same tree structure
- Each PU will now run through the array and obtain a list of all elements that are part of top 10
- With the same idea, each parent node will now combine the arrays of itself and its two children before passing it to the parent until the root is reached.
- **Do the two passes at the same time as you can pass up the local top 10. Reduces communication**

Q3.f [5 marks]

Distributed memory WITHOUT collective operations (No MPI_Reduce). Optimize using Gather/Scatter or other methods.

Your Answer:

Same thing? Sort, and then use binary search ig. And for g use tree?

Q4. Memory Model

Code segment on P1, P2, P3.

Q4.a [4 marks]

Explain two performance issues you might observe when using shared and distributed memory architectures, respectively?

Your Answer:

- Shared memory may be in contention, resulting in serialized access
- Shared memory may run into the issue of false sharing if cache lines are not managed properly.
- Distributed memory requires explicit synchronization **Communication Cost**
- **Memory consistency**
- **ache Coherence**

Q4.b [4 marks]

Is there a problem related to cache coherence in distributed memory systems? Explain your answer.

Your Answer:

No. Distributed memory systems focus on the distribution of data, and the caches in each machine are physically unable to write or overwrite the caches of other machines. **Updating the caches are done based on explicit messages**
Also Yes: File system may be shared. Caching files in main memory leads to the same problem

Q4.c [4 marks]

Identify valid/invalid states under sequential consistency for the given table.

Your Answer: