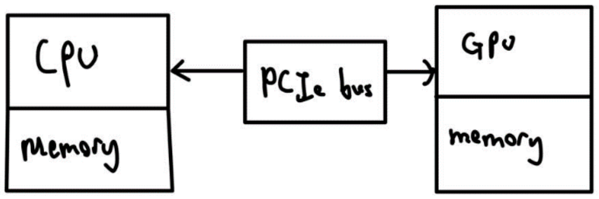
FIT3143 2024 - Applied Session Week 10

Student ID	**Student Name**	**Student Email**
32844700	Teh Jia Xuan	Jteh0015@student.monash .edu

Question 1 – Part A

	GPGPU/GPU	CPU
Number of cores	GPUs can have thousands of cores to perform many tasks simultaneously but with less individual core power compared to CPU core	CPUs have fewer cores usually between 4 – 64 cores. Each core is designed for general purpose tasks
Core complexities	GPU cores are simpler, it designed for executing many tasks concurrently. Each core is amphasize on throughput over latency, so they can focus on parallel execution of many simple tasks	CPU core is highly complex, it is designed to handle a wide variety of tasks. It has advanced features like branch prediction, large caches etc. CPUs are optimised for serial processing
Memory bandwidth	GPUs have much higher memory bandwidth so they can move large amounts of data quickly to and from memory.	CPUs have lower memory bandwidth since they are optimised for latency sensitive tasks that require low-latency access to memory.
ILP exploitation	GPUs doesn't focus much on ILP exploitation. They rely on executing many threads in parallel rather than ILP. GPUs execute simple instructions for thousands of threads simultaneously	CPUs are designed to exploit ILP by executing multiple instructions from a single thread. It helps CPUs achieve high performance

Question 1 – Part B



Here is the procedure of how data can be transferred between GPU and memory

1) CPU decides which data needs to be sent to GPU
2) CPU issues instructions to transfer data from main memory
3) CPU activates the Direct Memory Access (DMA) controller to transfer data
4) DMA controller manage the data transfer over PCIe bus, it is a high speed data connection between CPU and GPU using data packets
5) DMA controller reads data from memory and prepares for transmission through PCIe
6) Data is sent from the main memory over the PCIe bus to GPU.
7) PCIe transfers data in packets, which include data and metadata
8) GPU memory controller receives the data from PCIe bus
9) GPU memory controller place the data in GPU's memory

Question 1 – Part C

Here is the simple processing & data flow of a CUDA programming model

1) CPU initialises the application and prepares the data, and the CUDA kernel that will run on the GPU
    - Matrix multiplication application, CPU initialises two large matrices that need to be multiplied

2) CPU allocates memory on both CPU and GPU
    - CPU allocate memory for the two matrices and for the result matrix on GPU's main memory and CPU's main memory

3) CPU then initiate DMA to transfer the data from main memory to GPU's memory using PCIe bus
    - CPU transfers two matrices to GPU

4) CPU will tell the GPU to start compute by lauching the CUDA kernal on the GPU. Kernal is a function written on CUDA that runs on GPU.
    - GPU launches thousands of threads, where each thread handle multiplication of single row and column

5) GPU divides the work into many threads and execute parallel

- Each thread handles a specific element of the result matrix, computing the dot product of a row from one matrix and a column from another

6) Computation is complete, CPU initiates a function to transfer the result from GPU's memory back to CPU's memory over the PCIe bus
    - Result of matrix multiplication is transferred back to CPU's main memory

Question 2 – Part A
Here are the reasons why time synchronization is important for distribute system:

Event ordering
1) Events occur multiple nodes, which often operate independently. Time synchronization is important for establishing a global order of events. Without synchronised time, determining the correct order of events is difficult

Time based coordination
2) Many algorithms use time-based coordination to ensure all nodes make decisions based on same data and conditions. For example the election algorithms it relies on timeouts to detect leader failures. If nodes don't have synchronised time then they may incorrectly detect a leader failure.

Consistency
3) Time synchronisation is important for performing task, where different nodes must start or complete tasks at the same time or in a specific time intervals. Synchronising time across nodes ensures that such operations happen as expected

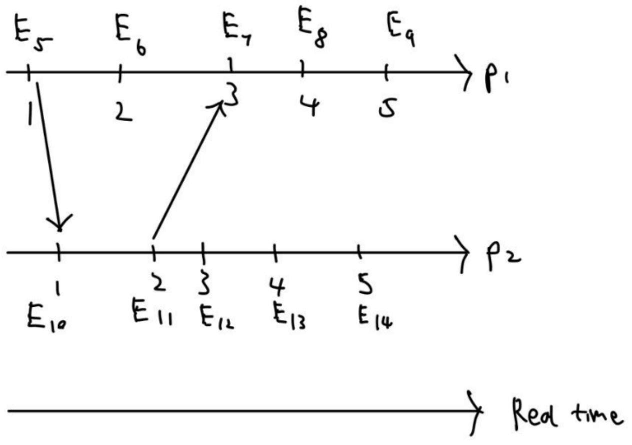
Question 2 – Part B

	Logical Clock	Real Clock Synchronization
Purpose	Logical clocks are used to order events in a distributed system to ensure consistent sequence of events.	Real clock synchronisation aligns the physical clocks. It ensures that all nodes have a consistent view of the actual time
What	Logical clocks do not refer to actual physical time, but it represents a sequence of events.	Real Clock refer to physical time and synchronised
Example	When a message is received the receiving process adjusts its logical clock based on the timestamp of the message	In real time financial systems, where transactions are timestamped then real clock synchronisation is critical. Events need to be correctly ordered with respect to actual time
Advantage	Logical clocks are simple to implement they provide a way to order events without requiring exact real time synchronisation	Real clocks are necessary when systems need to track actual time ensuring the global consistency
Disadvantage	- Only provide a partial ordering of events	- real clock synchronization relies on network connectivity,

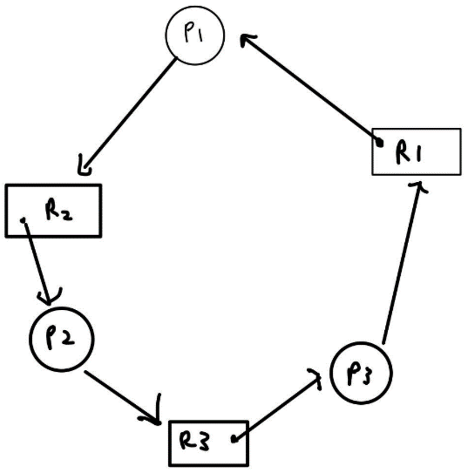
	- Cannot track real world timing - cannot distinguish concurrent events, two events happening at the same time but on different nodes may appear as sequential in logical clock.	if network connections fail it can lead to incorrect time. - keeping clocks synchronized requires continuous communication between nodes, which increase network traffic

Question 2 – Part C
Lamport's clock only captures causal relationships, according to the diagram below event C1(E5) -> C2(E10) then Lamport's clock show C1(E5) < C2(E10). However, if there are concurrent events it is not ordered precisely by the algorithm. For example, events that are concurrent such as E6 and E12, we may have C1(E6) < C2(E12) but there is no causal relationship between them. In another word, C1(E6) < C2(E12) could happen even though E6 not→ E12, meaning E6 did not cause E12 to happen. Thus, lamport clocks establish a partial ordering of events, meaning they can order some events based on causality. But they cannot establish total ordering as the independent events like E6, E12, E13 cannot be ordered by lamport clocks. For instance, E8 and E13 happen in separate processes there is no way to say which event happened first. This total order doesn't accurately reflect, and these two independent events are ordered as if one happened before the other.

Besides, lamport's clock only tracks logical progress and it doesn't correspond to real world time, so the timestamp comparison does not reflect actual real time ordering.



Question 3 – Part A



Condition for a deadlock:

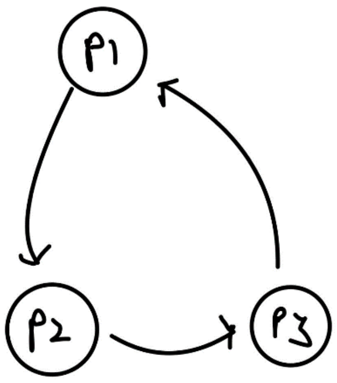
1) Mutual exclusion, it states that a resource can be assigned to at most one process. In my diagram R1 is held by P1, R2 is held by P2 and R3 is held by P3

2) Hold and wait, the processes holding a resource and wait for another resource. In my diagram P1 is holding R1 and P1 is waiting for R2. P2 is holding R2 and P2 waiting for R1. P3 is holding R3 and waiting for R1

3) Non-preemption, it states that when a resource is granted and it cannot be assigned to another process. In my diagram P2 is waiting for R1 as R1 is held by P1 and it cannot be assigned to P2. Same goes to P2 and R2, P3 and R3

4) Circular wait, it states that two or more processes remain idle as they are waiting for resources that occupied by the other processes. In my diagram P1, P2 and P3 are waiting for resources that assigned to other processes.

Question 3 – Part B



A wait for graph is a graph that used to represent dependencies between processes in a system. Each node in the graph represents a process and a directed edge from P1 to P2 means that P1 is waiting for a resource held by P2. If there is a cycle in wait for graph then it has potential deadlock.

Whereas the Chandy Misra Haas algorithm, is a deadlock detection algorithm. It works by sending messages along the edges of wait for graph to detect cycles. The Chandy Misra

Haas algorithm uses the wait for graph structure to propagate information about which processes are waiting on others. If the algorithm detects a cycle then the deadlock exist

How Chandy Misra Haas works is

1. according to the wait for graph, P1 initiates a probe message sends to P2
2. P2 forwards to P3
3. P3 forwards to P1 which completes the cycle.
4. Since the initiator received the probe, thus the system detects a deadlock.

Question 3 – Part C

A centralised coordinator is each process will continuously communicate with the coordinator regarding its state, whether it is requesting or releasing resources. The centralised coordinator will then analyse the messages and maintain the resource allocation graph. A false deadlock might happen when the centralised coordinator receives delayed or outdated messages. For example,

1. P1 holding resource R1
2. P2 holding resource R2
3. P1 releases R1 and then request R2, and P2 releases R2 and then request R1
4. P1 sends a release message of R1 to coordinator
5. P1 sends a request message for R2 to coordinator
6. P2 sends a release message for R2 to coordinator
7. P2 sends a request message for R1 to coordinator
8. Due to delays coordinator receives the request messages before the release messages
9. Coordinator will construct the RAG and discovered a cycle
10. False Deadlock detected

Thus, if the message requesting for the resource is sent to coordinator before the message of releasing it, the coordinator will construct the RAG that has a cycle. Thus, it detects a false deadlock.

Moreover, sometimes processes may have formed a temporary circular waiting condition in the RAG, but it is quickly resolved. If the coordinator collects data during this state. It could detect a false deadlock. For example,

1. Processes P1, P2 and P3 form a circular wait for a moment, but they resolved the situation by releasing resources.
2. The coordinator processes the RAG at this moment, it is interpreting the situation as a deadlock, even though it was temporary

AI Acknowledgment Statement

I would like to acknowledge the contributions of artificial intelligence technologies (ChatGPT) that have supported my work.