MONASH University

FIT3143 2024 - Applied Session Week 12

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

Question 1 – Part A

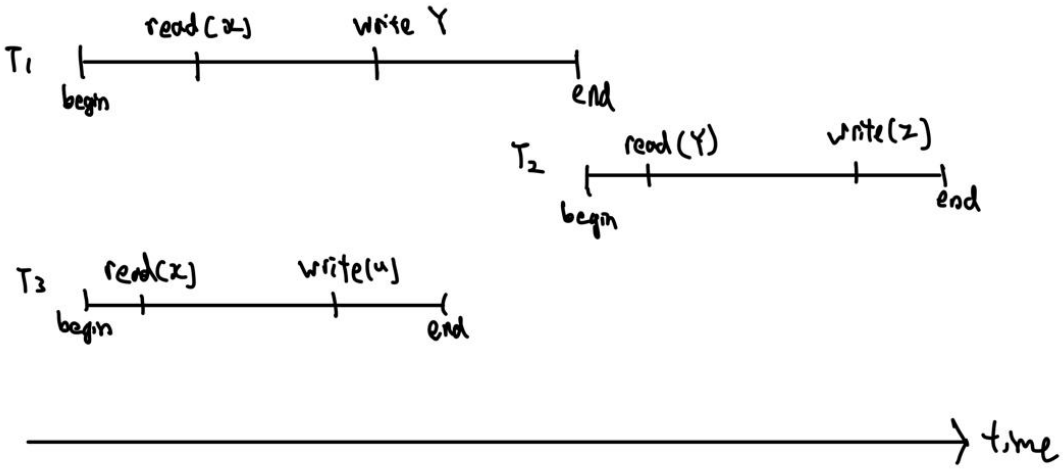
In lock based concurrency control, transactions acquire locks to ensure data consistency when multiple transactions access the same data concurrently. By distinguishing read locks from write locks, we improve concurrency because:

1) Multiple readers can access the data simultaneously, reducing unnecessary delays
2) Writers still maintain data integrity by preventing other transactions from accessing the data while it is being modified

According to the diagram below T1, T2 and T3 where:

1) T1 reads a data item X and write to Y
2) T2 reads a data item Y and write to Z
3) T3 reads a data item X and write to U

Without distinguishing read locks from write locks, T3 may block even though it is not modifying X. By using a read lock, both transactions can proceed without blocking each other. Having write locks T2 needs to wait for T1 to modify Y before reads it. This can prevent data corruption and confusion which can lead to redo the transaction over and over again due to data confusion. Thus improve the performance.

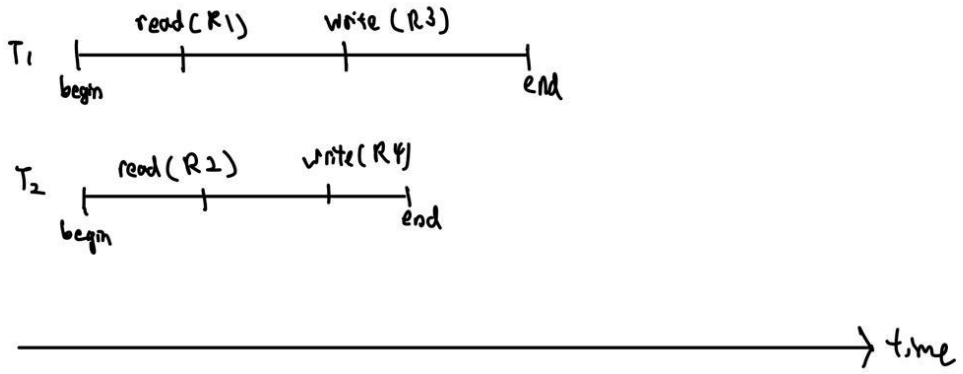


Question 1 – Part B

Lock granularity refers to the size of the data unit over which a lock is applied. Fine grained locks are applied on smaller data units, coarse grained locks are applied on larger data units. The choice of lock granularity impacts the parallel performance.

1) Fine grained locks allow more concurrency because multiple transactions can access different parts of the data simultaneously. For example, a table that consist of data R1, R2 and R3. Consider we have two transaction T1 and T2 which has shared data, T1 wanted to access R1 so it locks R1 and T2 wanted to access R2 it locks R2. Thus, T1 and T2 can acquire locks on different data and proceed in parallel. However, fine grained locks have higher overhead.

2) Coarse grained locks have less overhead but restrict concurrency as fewer transactions can access different parts of the data simultaneously. For example, consider two transactions T1 and T2, a table consist of data R1, R2 and R3. Now T1 wanted to access R1 so it uses coarse grained locks to lock the whole table, and T2 wanted to access R2 but due to T1 locked the whole table T2 needs to wait T1 to release the lock before access to it. So it has low parallelism.



Question 1 – Part C

In a timestamp based concurrency control, each transaction is assigned a unique timestamp, typically based on system's current time. This timestamp is used to determine the order in which transactions are processed, ensuring serializablity. Thus, synchronization algorithms is important in timestamp based concurrency control as it needs synchronise time to keep track of each transaction.

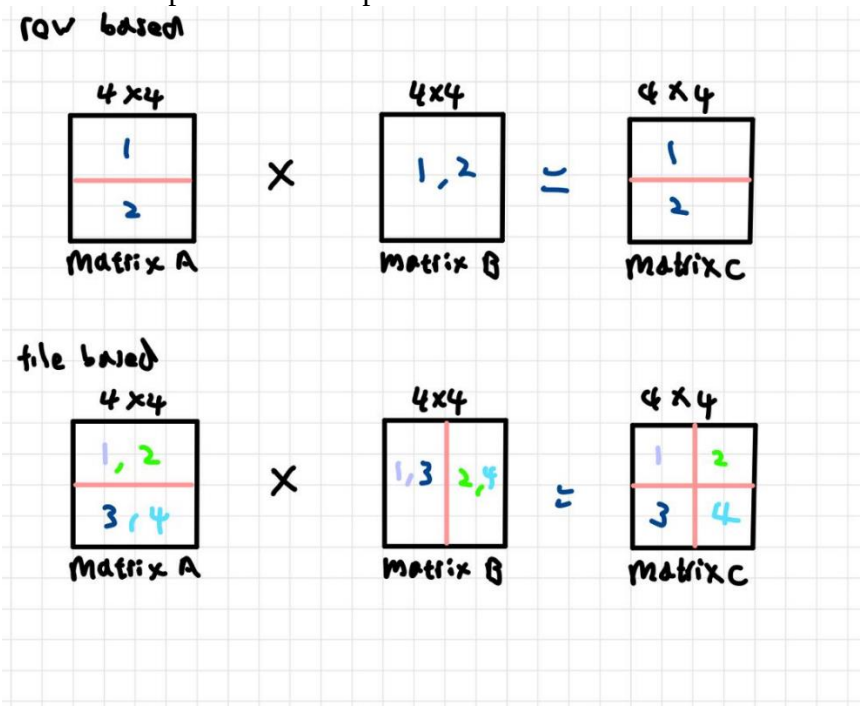
In distributed systems, where transactions occur across multiple machines, each machine has its own system clock and all of machine's clock might be different. Thus to keep track each transactions having synchronised clocks across all machines is essential for timestamp based concurrency control to work correctly.

If clocks are not synchronised different machines may assign timestamps that don't reflec the actual order of transactions execution. For example, Machine A clock is ahead of Machine B clock a transaction on machine A might get a timestamp that indicates it occurred before a transaction on machine B even though it actually happened later. This can lead to inconsistent states. Moreover, there will be write write conflict, for instance, machine A with later timestamp writes to a data item, but transaction on machine B with an earlier timestamp also writes to same data item so the data item has been overwritten.

Question 2 – Part A

	Row Wise block striped	Tile segmentation based
Purpose	Row is split among the processes each process needs to calculate their own result row.	The matrix is split into smaller square tiles to each process. Thus, each process needs to calculate a square tile of result of that result matrix
Decomposition	Each process gets a row block of matrix A. Let P = processes and M = number of rows in matrix A. So each processes get M/p rows of matrix A Each processes get the entire matrix B	Each process gets a row block of matrix A. For example, process 1 needs to calculate result of tile row 1 to 3 and column 1 to 3. Thus row of 1 to 3 of matrix A needs to be sent to process 1 For matrix B column 1 to 3 needs to be sent to process 1
Efficiency	Minimal, as each process only gets its assigned row block in matrix A but in matrix B each process needs to get the entire matrix B	Each process only communicates its assigned row/column of matrix A and B. Less communication overhead

Matrix Multiplication Example:



For row based it split in rows, so for matrix A process 1 only needs to get the upper row of matrix A and entire matrix B to calculate matrix C upper row and same goes to process 2. For tile based process 1 needs to get upper row for matrix A and left column of matrix B in order to calculate its tile result in matrix C.

MONASH University

Question 2 – Part B

Bernstein's conditions are used to determine whether two instructions can be executed parallel without causing any data hazards. The conditions are read-write set, write-read set and write-write set of x and y must be null

y = y + z + 1; // Instruction 1
x = z; // Instruction 2

R(I1) = {y, z}
W(I1) = {y}
R(I2) = {z}
W(I2) = {x}

1. R(I1) n W(I2) = {y, z} n {x} = { }
2. W(I1) n R(I2) = {y} n {z} = { }
3. W(I1) nW(I2) = {y} n {x} = { }

It shows that instruction 1 and 2 satisfies bernstein's conditions. As condition 1 to 3 is null.

Question 2 – Part C

Let P = number of processors and let N = row of matrix
Step 1 shift matrices A and B

1. Each processor is responsible for N/sqrt(P) x N/sqrt(P) block of matrix A
2. Before starting the computation each block of matrix A is shifted to left by amount equal to its row index in grid
3. Similarly matrix B is shifted up by an equal amount to its column index in the grid

For the initial alignment each processor communicates with its neighbouring processors in the same row for matrix A and in the same column for matrix B. This involves shifting blocks to adjacent processors.

Step 2 compute and shifting

1. Compute a partial matrix multiplication for block of C using last step's matrix A and B. (eg. Cij = Cij + Aij * Bij)
2. After that for matrix A, each block of A is shifted one position to the left
3. After that for matrix B each block of B is shifted one position up

For each shift communication is local, this process of shifting and computing repeats for N iterations ensuring that every processor receives every necessary block of matrix A and matrix B for the complete matrix multiplication.

Step 3 result

1. After N-1 iterations each processor will have computed its portion of the result matrix C
2. Each processors send to root then combine into a global matrix.

MONASH University

Question 3 – Part A

Dennard scaling describes how shrinking transistor dimensions results in proportional reductions in both voltage and current, allowing transistors to switch faster and consume less power. So power density remains constant even as more transistors are packed into the same chip area. This is important for exponential growth as when the transistors shrink their capacitance decrease leading to lower power consumption so this allowed for more transistors on a chip without increasing overall power usage. This scalability contributed to rapid advancements in computer performance, and enable exponential growth in processing power while keeping heat and power consumption manageable

Question 3 – Part B

According to Moore's law says that number of transistors double every two years but sometimes our tasks would not have significant improvements in performance due to two key reasons:

1. According to Amdahl's law serial components of the algorithm will limit the speed up as serial component is the portion of task that cannot be parallelised. According to the formula 1/rs + rp/N. As the number of processors increases N will become larger and rp will become smaller. But the rs which is serial component remains constant. Thus, if N becomes very large rp will close to 0 at this point no matter how many processors added the speed up and rs will remain the same. As it already reached the limit of the speed up.

2. According to amdahl's law 1/ rs + rc + rp/N, Increasing the number of cores often introduces communication overhead, when the number of cores increases the communication overhead increases. Thus, rc increases and caused speed up to be slow as the program also need to spent time on sending data to other processors. Moreover, even though the task is theoretically parallelisable, but the actual speed up is affected by the communication costs. As a result, actual speed up often lesser than theoretically speed up

Question 3 – Part C

Keck's law is saying data transmission capacity of optical fibers has increased exponentially over the year. This would benefit the performance of a distribution application.

1) As bandwidth increases, the time it takes to transfer data between distributed systems decreases. For applications that need to exchange large datasets, due to bandwidth increases it can exchange larger datasets at once so it leads to faster communication leads to lower overall latency and quicker responses

2) Enhance scalabiltiy, as higher bandwidth it supports scaling of distributed applications across more nodes. With higher bandwidth the network can handle more trafftic. This allows applications to scale more efficiently, handling a larger number of requests without experiencing bottlenecks.

3) Facilitation complex distributed systems as bandwidth continues to grow, more complex distributed systems can be accommodated. These systems often require frequent communication between processes, if bandwidth increases more data can be sent at once between processes this enable device to transfer data quickly and enable smoother operation of these complex systems.

AI Acknowledgment Statement

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