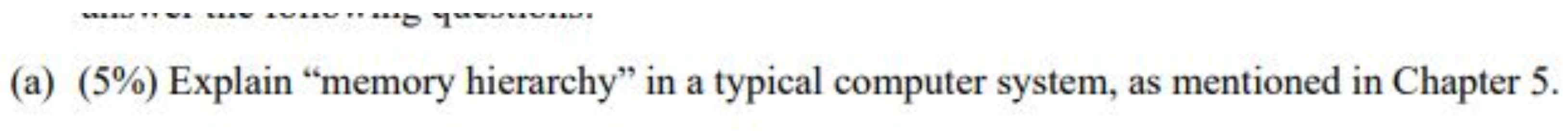
CA2024 期末考評分標準

a



answer the following questions.

(a) (5%) Explain "memory hierarchy" in a typical computer system, as mentioned in Chapter 5.

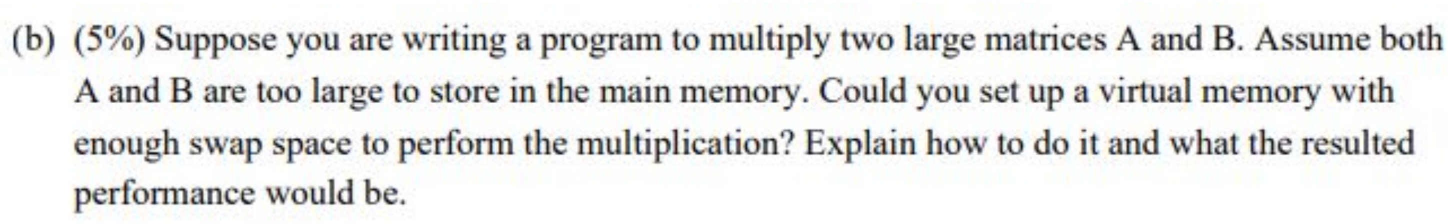
有提到 locality 或 階層的速度，大小關係　　-> 全對

沒提到但看得出來方向正確 / 有提到但有小錯誤 -2

方向錯誤但部分正確 -4

錯誤 -5

b

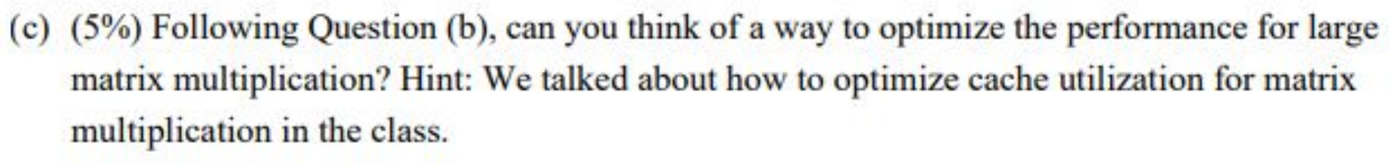

(b) (5%) Suppose you are writing a program to multiply two large matrices A and B. Assume both A and B are too large to store in the main memory. Could you set up a virtual memory with enough swap space to perform the multiplication? Explain how to do it and what the resulted performance would be.

(1pt)yes

(2pt)回答有大概介紹 vm 設立的機制

(2pt)本題比較的對象是能放在main memory 的矩陣乘法，所以因為還要到Disk 讀資料，所以應為變慢。但若同學比較的不是記憶體而是其他的，並且有明確寫出比較的對象，回答正確的話一樣會給分

c


(c) (5%) Following Question (b), can you think of a way to optimize the performance for large matrix multiplication? Hint: We talked about how to optimize cache utilization for matrix multiplication in the class.

loop-blocking,

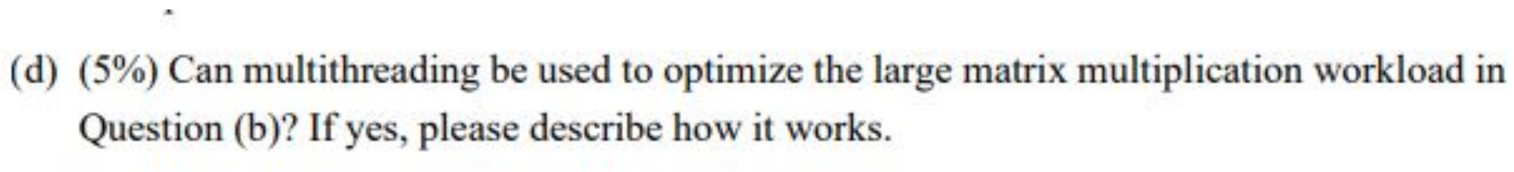
partition,

spatial locality

有關鍵字且敘述完整得 5pt

敘述不夠完整得 2pt

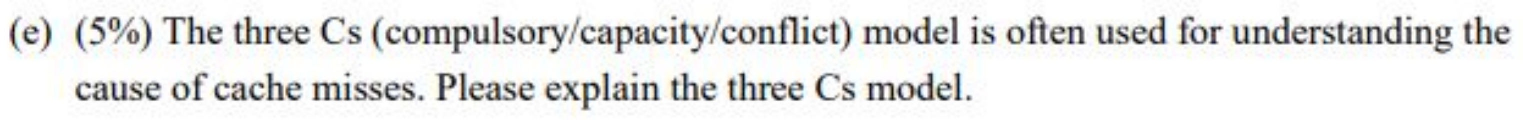
d


(d) (5%) Can multithreading be used to optimize the large matrix multiplication workload in Question (b)? If yes, please describe how it works.

(1pt)可以

(4pt)兩種答案，一種是前面提到的 block 做 multithread, 或把 A, B 的 row column apply 給不同 thread (因為矩陣乘法的資料相依性很低，所以可以平行化)

e


(e) (5%) The three Cs (compulsory/capacity/conflict) model is often used for understanding the cause of cache misses. Please explain the three Cs model.

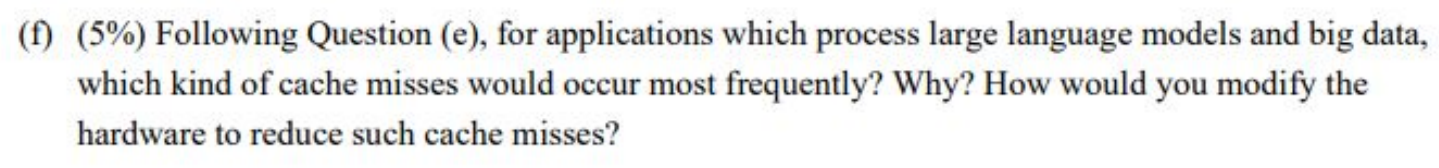
Compulsory: 冷開機 miss，因為一開始 cache 沒東西所以必定會發生 miss

capacity: in fully-associative cache, 因為資料空間比 cache 還大，沒辦法整個放在其中而發生的 miss

conflict: in direct-mapping / set-associative cache, 資料因為競爭同一個 index 而造成的 miss

錯第一個 -1, 第二三個各 -2

f


(f) (5%) Following Question (e), for applications which process large language models and big data, which kind of cache misses would occur most frequently? Why? How would you modify the hardware to reduce such cache misses?

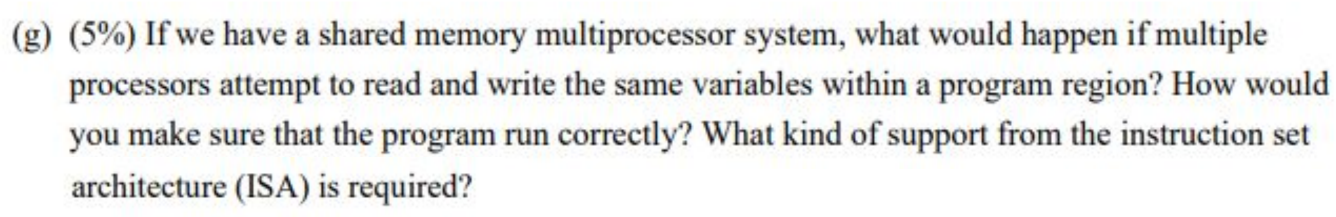
(2pt) llm 和 大數據的資料會反覆用到，且容量巨大，cache 擺不下

(1pt) 所以最常出現的 miss 應為 capacity

(2pt) 解法為增加 cache size

若回答 conflict 解法則為增加關聯度 or 增加 cache size

若回答 compulsory 解法則為增加block size, prefetch g



(g) (5%) If we have a shared memory multiprocessor system, what would happen if multiple processors attempt to read and write the same variables within a program region? How would you make sure that the program run correctly? What kind of support from the instruction set architecture (ISA) is required?

(2pt) race condition, might led to incorrect result or get the incorrect data.

(2pt) provide mutex/spin lock, atomic operation

(1pt) provide atomic operation，應為重點是要避免 instruction 間的 overlap

(h)

這題寫到可以增加 memory bandwidth 或減少 memory bandwidth 用量的方法即可得分。像是使用 NUMA 架構、使用 multi-layer cache 、使用 Interleaved Memory，增加 bus width 等~

有提到這些方法之一就可得滿分

其他情形則按情況給分

(i)

這題是問怎麼解決 cache coherence 問題，標準答案為課本中的 snooping-based, directory-based cache coherence protocol，有提到這二者，或是提到 broadcast write-invalidate 等方式即可得分

若寫其他答案，則按情況給分

(j)

#pragma omp parallel for 是放在最外層迴圈，故會針對最外層迴圈進行平行化。i = 0 ~ 15 由 processor 0 執行；i = 16 - 31 由 processor 1 執行 ;i = 32- 47 由 processor 2 執行;i = 48- 63 由 processor 3 執行 。且各 processor 只會存取到自己的 C[i][j]，故不會有 inter-processor dependency 。

平行化的方式不一定要照這個，如果是針對最外二層迴圈(i, j)進行平行化，均為可接受答案。

若有正確平行化且說明沒有 inter-processor dependency 則可得滿分。只有正確回答一種答案則得 3 分。其餘按情況給分。

(k)

message passing 的運作方式跟 shared memory 不同在於，節點都各自有自己的私有 memory，並透過 send/receive 交換資料及同步。各節點透過網路 (Ethernet 或 Infiniband) 連接。它的特性是擴展性高，也更容易替換故障節點~ 以相同算力和相同 memory 來看，用 message passing 方式的多節點架構可能比較是 cost effective 的方式，因為單節點的算力及 memory 通常有一定的上限，要再買跟高擋的機器可能也買不到，或是很貴。且若單台機器，要讓更多顆處理器要共享 memory，也會遇到頻寬上限，或是其線路要更為複雜。

有提到 messaging passing 運作方式及其比較 cost effective，即可得滿分。

其他情形按情況給分。

(I)

同樣可以將最外層迴圈(i)進行平行化。但不同的點是：開始進行矩陣運算前，需要主要節點把各節點會用到的 A及 B 矩陣用 send/receive 方式傳到其他節點，再開始進行矩陣乘法。各節點運算完後，再把各節點的 C 的結果用 send/receive 傳回主節點 (Gather) 。

若有寫出類似上述的答案，則可得滿分

其他情形按情況扣分

(m)

- Memory bound 區域斜率: 72/75 = 0.96
- Peak throughput：(180+144)/2 = 162 TFLOPS
- 轉折點的 A.I：162 / 0.96 = 168.75
- Context Stage: 因其 A.I > 168.75，位於 compute bound，故其 Peak throughput 為 162 TFLOPS
- Generation Stage: 1 * 0.96 = 0.96 TFLOPS

有分別列出 Context Stage: 162 TFLOPS,   Generation Stage: 0.96 TFLOPS 即全對。只寫對一個或二者寫相反, 則得 2.5 分, 其餘答案不給分。

(n)

因為 Generation Stage A.I 只有 1 為 Memory bound，且它又佔了幾乎所有運算的時間 (310ms/320ms)，故 TinyChat is mostly memory bound 。

有正確解釋且有寫出 memory bound 給滿分。

寫 memory bound，未解釋或解釋錯誤得 2 分。

若寫 Compute bound 不給分。

3. (o)

參考解答:

- Number of floating-point operations

  context stage = 162 TFLOPs/s * (10 ms / 1000) = 1.62 TFLOPs
  generation stage = 72/75 TFLOPs/s * (310 ms / 1000) = 0.298 TFLOPs
  Total = 1.918 TFLOPs = 1.9 * 10^12 FLOPs

- Number of bytes memory access

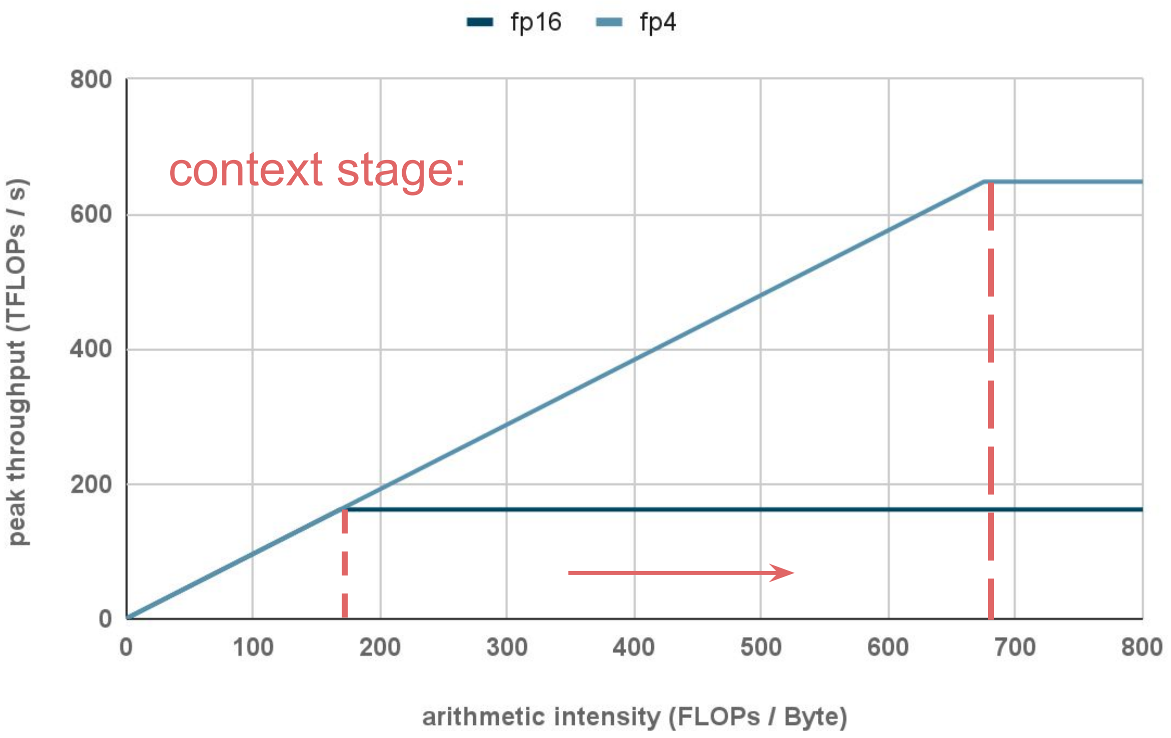
  context stage = 1.62 TFLOPs / 170 FLOPs/Byte = 0.0095 TBytes
  generation stage = 0.298 TFLOPs/s / 1 FLOPs/Byte = 0.298 TBytes
  Total = 0.3075 TBytes = 3 * 10^11 Bytes

3. (p)

參考解答:

Yes, the execution time is reduced to 80ms.

When we change to fp4, A.I. becomes 4 times larger, and peak throughput also becomes 4 times larger.



4. (q)

參考解答:

Aliasing happens when two processes share a memory region. Two different virtual addresses (from each process) are mapped to the same physical address. Since the cache is virtually indexed, virtually tagged, a piece of data from the shared memory might be cached in two different cache blocks. When one process writes the data, the other process will not know. (cache coherence issue)

VA1	data = 0
...	...
VA2	data = 0
...	...

VA1	data = 0
...	...
VA2	data = 40
...	...

should be the same

4. (q)

參考解答 (continued):

To address this issue, we can (but not limited to):

1. Let the cache be virtually indexed, physically tagged instead, and the index bit should be the lower bit of virtual address, normally within the page offset bits.
2. Let the cache be physically indexed, physically tagged instead.
3. Resolve the cache coherence issue by methods similar to the snooping protocol.

4. (r)

參考解答:

這題想要問的是因為 GPU 的記憶體有限，當一個問題太大的時候我們怎麼把它切成小問題，分次送給 GPU 做運算。

這題針對的問題是矩陣運算，而題目說我們可以用 block by block 的方式拆解矩陣運算，所以解法就是 block by block 的矩陣乘法，然後計算每個 block 的乘法時都交給 GPU 算（把資料從 CPU 傳給 GPU 做運算，GPU 再把結果傳回 CPU）

4. (s)

參考解答:

這題要問的是 prefetching 能不能加速上題的運算。

就同學們的情形我們分兩個情況來改, 一個是單論 GPU 的運算而沒有考慮跟 CPU 的來回傳輸, 另一個則是有考慮(題目想問的是這種情況)只要邏輯合理就會給分!

兩種情況有不同的 bottleneck:

情況一(考慮單純的 GPU 運算): 從 memory 存取資料的時間
情況二: CPU 跟 GPU 之間資料傳輸的時間
兩種情況都可以用 prefetching 來優化!

4. (s)

參考解答:

No, it can’t perform the task effectively.

The bottleneck will be the communication time between CPU and GPU. (we need to send the matrices from CPU to GPU and receive the results from GPU)

Yes, prefetching is useful.

Since computation and communication can be handled parallelly, when GPU is performing matrix multiplication for a block, it can prefetch the next block in parallel.

4. (t)

參考解答:

For any partition, there are at least 6 NVLINK connecting between the two groups. Therefore, the bisection bandwidth = 6*50 = 300 GB/s