Computer Architecture HW4

Fabian Wüthrich

November 30, 2020

1 Critical Paper Reviews [500 points]

see here

2 Memory Systems [200 points]

a)

Byte on bus:	`Addr[2:0]`
Channel bits:	`Addr[11:10]`
Bank bits:	`Addr[13:12]`
Column bits:	`Addr[9:3]`
Row bits:	`Addr[31:14]`

b) **App 1**

All requests of App 1 can be served from the same row, because App 1 accesses 16 64-bytes cache blocks i.e. the size of a row (1024 bytes). As A comes before B and all later requests are row hits, App 2 never interrupts the request of App 1 so App 1 has the same running time alone as with App 2 together i.e. a slowdown of 1.

App 2

If App 2 runs alone the first request is a row miss followed by to row hits i.e. a runtime of $2T + 2 \times T = 4T$. If App 2 runs together with App 1, then the requests of App 1 are executed before the requests of App 2 i.e. a runtime of $2T + 15 \times T + 2T + 2 \times T = 21T$. Thus, the slowdown is $\frac{21}{4}$.

c) App 2 slows down more.

App 1 is memory intensive and has a high row-buffer locality. Coupled with the FR-FCFS policy, App 1 denies App 2 access to the shared DRAM, which incurs the slow down of App 2.

d) A solution which reduces the slowdown of App 1 without increasing the slowdown of App 2, requires some sort of parallelism. The data of both applications are mapped to the same bank, so memory channel partitioning is not an option. Similarly, interleaving the data of App 1 and App 2 to reduce the row hits of App 1 or source throttling would increase the slowdown of App 2. A last option would be some kind of inter-bank parallelism but this requires fundamental changes to the DRAM organization.

3 DRAM Scheduling and Latency [200 points]

a) The DRAM command bus operates at 1 GHz i.e. 1 cycle is equal to 1 ns, so we omit an explicit conversion from cycles to nanoseconds in each subtask.

i) `ACTIVATE` to `READ`: 5 ns

In cycle 13, a `ACTIVATE` is scheduled followed by a `READ` in cycle 18. Similarly, the `ACTIVATE` in cycle 20 is followed by a `READ` in cycle 25 (the `READ` in cycle 22 goes to a different bank).

ii) `ACTIVATE` to `PRECHARGE`: *unknown*
There is no `PRECHARGE` followed by an `ACTIVATE` in the above command trace.

iii) `PRECHARGE` to `ACTIVATE`: 12 ns
The `PRECHARGE` command in cycle 1 is followed by an `ACTIVATE` in cycle 13 (same for cycle 8 and 20).

iv) `READ` to `PRECHARGE`: 8 ns
After the `READ` command in cycle 0, the `PRECHARGE` command is delayed by 8 cycles (the `PRECHARGE` in cycle 1 must go to a different bank to make the other timing constraints consistent)

v) `READ` to `READ`: 4 ns
The `READ` command in cycle 18 and the `READ` command in cycle 22 go both to bank 0 and they are 4 cycles apart.

b) The `READ` command in cycle 0 goes to bank 1 so the bank must have been open with row address 43 (deduced from `0xD780`). Bank 0 is pre-charged in cycle 1 so the bank was open but we cannot determine the row address.

c) The following command trace has a total runtime of 23 cycles as required:

```
Cycle 0  --- READ       // bank 1 row 43
Cycle 1  --- PRECHARGE  // bank 0 row unknown (near segment)
Cycle 8  --- PRECHARGE  // bank 1 row 43 (near segment)
Cycle 10 --- ACTIVATE   // bank 0 row 20 (near segment)
Cycle 14 --- READ       // bank 0 row 20 (near segment)
Cycle 17 --- ACTIVATE   // bank 1 row 50 (far segment)
Cycle 18 --- READ       // bank 0 row 20 (near segment)
Cycle 23 --- READ       // bank 1 row 50 (far segment)
```

All other traces don't have a runtime of 23 cycles. Thus, in bank 0 row 20 and in bank 1 the unknown row and row 43 has to be in the near segment. If row 50 is in the near segment, the runtime would be shorter. Therefore, N is between 44 and 50.

4 Memory Scheduling [200 points]

4.1 Application-Unaware Scheduling Policies

a) **App. A:** $9 \times 45ns + 15ns = 420ns$
App. B: $8 \times 45ns + 2 \times 15ns = 390ns$
App. C: $10 \times 45ns + 15ns = 465ns$

b) **App. A:** $6 \times 45ns + 4 \times 15ns = 330ns$
App. B: $5 \times 45ns + 5 \times 15ns = 300ns$
App. C: $7 \times 45ns + 4 \times 15ns = 375ns$

c) row buffer locality

d) prioritize row hits and use FCFS as tie breaker

4.2 Application-Aware Scheduling Policies

a) **App. A:** $330ns + 45ns + 2 \times 15ns = 405ns$ (add R27 and `ACT`/`PRE` for R12)
App. B: $300ns + 45ns + 2 \times 15ns = 375ns$
App. C: $45ns$

b) C, B, A

c) **App. A:** $8 \times 45ns + 3 \times 15ns = 405ns$
App. B: $4 \times 45ns + 3 \times 15ns = 125ns$
App. C: $45ns$

d) Y < X < FR-FCFS < FCFS

5 Memory Request Scheduling [200 points]

a) The scheduler forms a batch with all requests in the request buffer for each bank. Both applications have the same number of requests outstanding, so the scheduler prioritizes the application with the oldest request. For bank 0, application A is prioritized and for bank 1 application B is prioritized.

Application A has a stall time of $45ns + 3 \times 15ns + 45ns3 \times 15ns = 180ns$ and application B has a stall time of $4 \times 15ns + 45ns + 15ns + 45ns + 15ns = 180ns$.

X prioritizes application A at bank 0 and application B at bank 1, which destroys bank level parallelism of both applications and gives a performance penalty. Additionally, the resulting scheduling order is exactly the same as with FR-FCFS so there is no improvement.

b) Scheduler Y could prioritize the same application across all banks. That would allow for bank level parallelism and guarantees a performance improvement. Within a batch, the application with the shortest stall time could be prioritized. In this example, application A would be prioritized over application B because application A has a better row buffer locality i.e. shorter stall time.

With the new policy, all requests of application A are scheduled before the requests of application B. Thus, the stall time of application A reduces to $45ns + 3 \times 15ns = 90ns$.

The requests of application B in bank 0 take the longest to complete. Thus, the stall time of application B is $60ns + 45ns + 15ns + 45ns + 15ns = 180ns$.

c) For application A nothing changes because no requests were delayed due to interference of other applications. So the stall time is still $90ns$.

Application B has now its own channel without any interference of application A. The improved stall time is $45ns + 15ns + 45ns + 15ns = 120ns$

Channel partitioning is better than scheduler Y because it removes the interference between the two applications which enables a higher throughput.

6 Emerging Memory Technologies [100 points]

a) The latencies for sending a request to ETH-RAM is given, so we need to find the latency that ETH-RAM requires to process the request. In the student's measurement, each cell received 10^6 writes and to touch all cells $\frac{2^{30}}{2^2} \times 10^6$ writes are necessary (ETH-RAM is word addressable). Thus, we get

$$365 \times 24 \times 3600 \times 10^9 ns = \frac{2^{30}}{2^2} \times 10^6 \times (l_{RAM} + 5ns + 28ns)$$

$$l_{RAM} = \frac{365 \times 24 \times 3600 \times 10^3}{2^{28}} - 33 = 84.5ns$$

Overall, the total memory access latency is $84.5ns + 5ns + 28ns = 117.5ns$

b) We are using ETH-RAM in SLC mode so each cell should receive $10 \times 10^6 = 10^7$ writes and the memory capacity is halved to 2^{29} bytes. The test requires $\frac{2^{29}}{2^2} \times 10^7$ writes and the latency drops to $0.3 \times 84.5ns = 25.35ns$. With these new parameters, we get a lifetime of

$$\frac{2^{29}}{2^2} \times 10^7 \times (25.35ns + 5ns + 28ns) \approx 2.49\, years$$

7 BossMem [100 points]

i) Some sort of prefetching or out-of-order execution allows the core to fetch the required data before BossMem has to cool down.

ii) This is a bad idea because a row miss/conflict has a much higher latency in DRAM than in BossMem. A better approach would be to place data with high row buffer locality in DRAM and data with low row buffer locality in BossMem to fully benefit from the reduced memory timing.

iii) A cache replacement policy that improves the performance of the hybrid memory system could replace data based on where it is mapped. For example, if it's more costly to fetch data from BossMem then the policy is less likely to replace data that is mapped to BossMem than data that is mapped to DRAM.

iv) PCM is non-volatile and endurance attacks are possible

v) DRAM should be of least interest. Despite attacks like RowHammer, DRAM is well studied and scrutinized by the security research community.