Computer Architecture (263-2210-00L), Fall 2018
HW 4: Memory Interference and QoS

Instructor: Prof. Onur Mutlu
TAs: Mohammed Alser, Can Firtina, Hasan Hassan, Jeremie Kim, Juan Gómez Luna,
Geraldo Francisco de Oliveira, Minesh Patel, Giray Yaglikci

Assigned: Wednesday, Nov 21, 2018
Due: **Wednesday, Dec 5, 2018**

- **Handin - Critical Paper Reviews (1).** You need to submit your reviews to `https://safari.ethz.ch/review/architecture18/`. Please check your inbox. You should have received an email with the password you can use to login to the paper review system. If you have not received any email, please contact comparch@lists.ethz.ch. In the first page after login, you should click in "Architecture - Fall 2018 Home", and then go to "any submitted paper" to see the list of papers.

- **Handin - Questions (2-5).** Please upload your solution to the Moodle (`https://moodle-app2.let.ethz.ch/`) as a single PDF file. **Please use a typesetting software (e.g., LaTeX) or a word processor (e.g., MS Word, LibreOfficeWriter) to generate your PDF file. Feel free to draw your diagrams either using an appropriate software or by hand, and include the diagrams into your solutions PDF.**

---

1 Critical Paper Reviews [150 points]

Please read the following handout on how to write critical reviews. We will give out extra credit that is worth 0.5% of your total grade for each good review.

- Lecture slides on guidelines for reviewing papers. Please follow this format.

  `https://safari.ethz.ch/architecture/fall2018/lib/exe/fetch.php?media=onur-comparch-f18-how-to-do-the-paper-reviews.pdf`

- Some sample reviews can be found here: `https://safari.ethz.ch/architecture/fall2018/doku.php?id=readings`

(a) Write a one-page critical review for the following paper:

- Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," Proceedings of the IEEE, 2017. `https://arxiv.org/pdf/1706.08642.pdf`

(b) Write a one-page critical review for at least **three** of the following papers:

- O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," ISCA 2008. `https://people.inf.ethz.ch/omutlu/pub/parbs_isca08.pdf`

- E. Ebrahimi, C. J. Lee, O. Mutlu, and Y. N. Patt, "Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems," ASPLOS 2010. `https://people.inf.ethz.ch/omutlu/pub/fst_asplos10.pdf`

- S. P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir, and T. Moscibroda, "Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning," MICRO 2011.

  `https://people.inf.ethz.ch/omutlu/pub/memory-channel-partitioning-micro11.pdf`

- L. Subramanian, D. Lee, V. Seshadri, H. Rastogi, and O. Mutlu, "BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling," TPDS 2016. `https://people.inf.ethz.ch/omutlu/pub/bliss-memory-scheduler_ieee-tpds16.pdf`

- A. Boroumand, S. Ghose, Y. Kim, R. Ausavarungnirun, E. Shiu, R. Thakur, D. Kim, A. Kuusela, A. Knies, P. Ranganathan, and O. Mutlu "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS 2018. `https://people.inf.ethz.ch/omutlu/pub/Google-consumer-workloads-data-movement-and-PIM_asplos18.pdf`

2 Memory Interference and QoS [100 points]

Row-Buffer Conflicts. The following timing diagram shows the operation of a single DRAM channel and a single DRAM bank for two back-to-back reads that conflict in the row-buffer. Immediately after the bank has been busy for 10ns with a READ, data starts to be transferred over the data bus for 5ns.
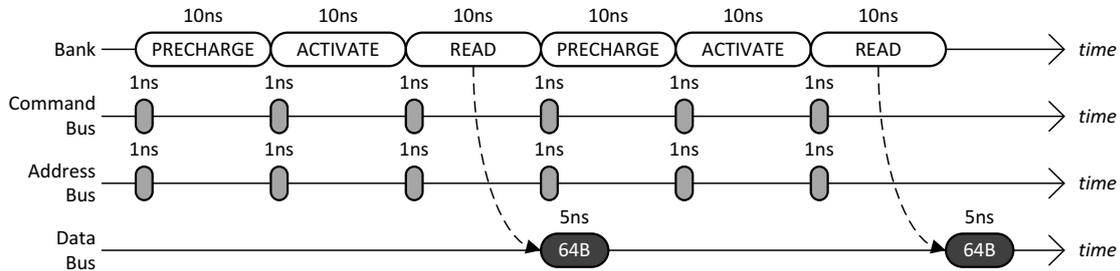
(a) Given a long sequence of back-to-back reads that always conflict in the row-buffer, what is the data throughput of the main memory system? Please state your answer in **gigabytes/second**.

(b) To increase the data throughput, the main memory designer is considering adding more DRAM banks to the single DRAM channel. Given a long sequence of back-to-back reads to all banks that always conflict in the row-buffers, what is the minimum number of banks that is required to achieve the maximum data throughput of the main memory system?

Row-Buffer Hits. The following timing diagram shows the operation of the single DRAM channel and the single DRAM bank for four back-to-back reads that hit in the row-buffer. It is important to note that row-buffer hits to the same DRAM bank are pipelined: while each READ keeps the DRAM bank busy for 10ns, up to at most **half** of this latency (5ns) can be overlapped with another read that hits in the row-buffer.
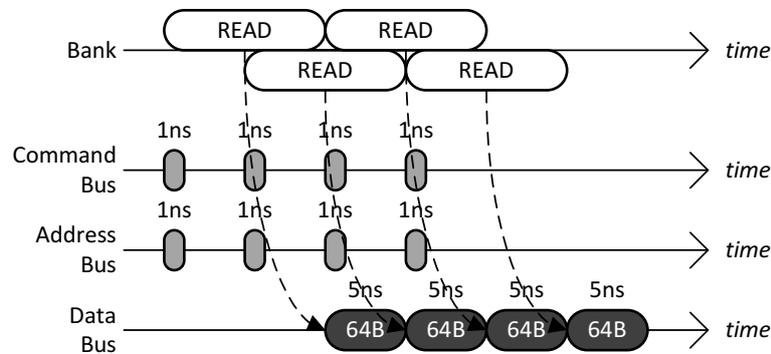

(c) Given a long sequence of back-to-back reads that always hits in the row-buffer, what is the data throughput of the main memory system? Please state your answer in **gigabytes/second**.

(d) When the maximum data throughput is achieved for a main memory system that has a single DRAM channel and a single DRAM bank, what is the bottleneck that prevents the data throughput from becoming even larger? **Circle** all that apply.

BANK   **COMMAND BUS**   **ADDRESS BUS**   **DATA BUS**

Memory Scheduling Policies. The diagram below shows the memory controller's *request queue* at time 0. The shaded rectangles are read requests generated by thread *T0*, whereas the unshaded rectangles are read requests generated by thread *T1*. Within each rectangle, there is a pair of numbers that denotes the request's (*BankAddress, RowAddress*). Assume that the memory system has a **single** DRAM channel and four DRAM banks. Further assume the following.

- All the row-buffers are **closed** at time 0.

- Both threads start to stall at time 0 because of memory.

- A thread continues to stall until it receives the data for all of its requests.

- Neither thread generates more requests.

4/13
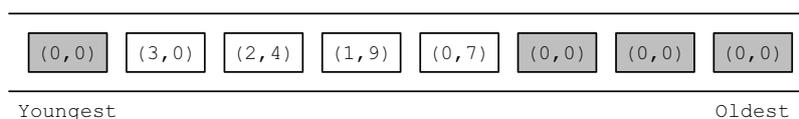
(f) For the $FCFS$ scheduling policy, calculate the memory stall time of *T0* and *T1*.

T0:

T1:

(g) For the $FR - FCFS$ scheduling policy, calculate the memory stall time of *T0* and *T1*.

T0:

T1:

(h) For the $PAR - BS$ scheduling policy, calculate the memory stall time of *T0* and *T1*. Assume that all eight requests are included in the same batch.

T0:

T1:

3   Memory Scheduling [50 points]

In class, we covered "parallelism-aware batch scheduling," which is a memory scheduling algorithm that aims to reduce interference between threads in a multi-core system.

(a) What benefit does request batching provide in this algorithm?

(b) How does the algorithm preserve intra-thread bank parallelism?

(c) If thread ranking was formed in a "random manner" (i.e., threads were assigned a random rank), would each thread's parallelism be preserved? Why or why not? Explain.

4 Memory Request Scheduling [100 points]

A machine has a DRAM main memory organized as 2 channels, 1 rank and 2 banks/channel. An open row policy is used, i.e., a row is retained in the row-buffer after an access until an access to another row is made. The following commands (defined as we discussed in class) can be issued to DRAM with the given latencies:

- ACTIVATE: 15 ns

- PRECHARGE: 15 ns

- READ/WRITE: 15 ns

  Assume the bus latency is 0 cycles.

(a) Two applications A and B are run on the machine. The following is a snapshot of the request buffers at time t0 . Requests are tagged with the index of the row they are destined to. Additionally, requests of applications A and B are indicated with different colors. Row 4 is initially open in bank 0 of channel 0 and row 10 is initially open in bank 1 of channel 1.

Each application is stalled until all of its memory requests are serviced and does not generate any more requests.
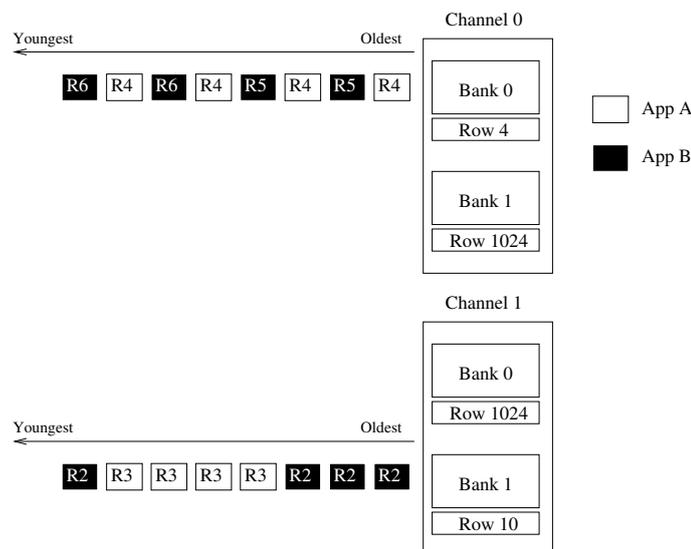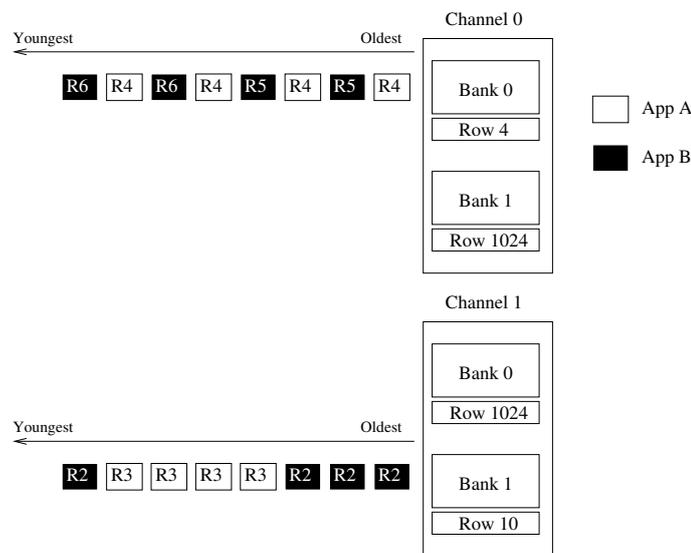

What is the stall time of application A using an FR-FCFS scheduling policy?

What is the stall time of application B using an FR-FCFS scheduling policy?

(b) We studied Parallelism Aware Batch Scheduling (PAR-BS) in class. We will use a PAR-BS-like scheduler that we will call X. This scheduler operates as follows:

- The scheduler forms request batches consisting of the 4 oldest requests from each application at each bank.
- At each bank, the scheduler ranks applications based on the number of requests they have outstanding at that bank. Applications with a smaller number of requests outstanding are assigned a higher rank.
- The scheduler always ranks the application with the oldest request higher in the event of a tie between applications.
- The scheduler prioritizes the requests of applications based on this ranking (higher ranked applications' requests are prioritized over lower ranked applications' requests).
- The scheduler repeats the above steps once all requests in a batch are serviced at all banks.

Channel 0

Youngest ← Oldest

| R6 | R4 | R6 | R4 | R5 | R4 | R5 | R4 |

Bank 0
Row 4

Bank 1
Row 1024

□ App A
■ App B

Channel 1

Bank 0
Row 1024

Youngest ← Oldest

| R2 | R3 | R3 | R3 | R3 | R2 | R2 | R2 |

Bank 1
Row 10

For the same request buffer state as in Part (a) (replicated above for your benefit):
What is the stall time of application A using this scheduler?

What is the stall time of application B using this scheduler?

The PAR-BS scheduler we studied in lecture provided better system performance than the FR-FCFS scheduler. Is this true for X also? (i.e., does X provide better system performance than FR-FCFS?)

Circle one: **YES** **NO**

Explain why or why not. Provide the fundamental reason why X does or does not improve system performance over an FR-FCFS scheduler.

```




```

(c) Can you design a better memory scheduler (i.e., one that provides higher system performance) than X?

Circle one: **YES** **NO**

If yes, answer the questions below. What modifications would you make to scheduler X to design this better scheduler Y? Explain clearly.

```




```

What is the stall time of application A using this scheduler Y?

```




```

What is the stall time of application B using this scheduler Y?

```




```

(d) Consider a simple channel partitioning scheme where application A's data is mapped to channel 0 and application B's data is mapped to channel 1. When data is mapped to a different channel, only the channel number changes; the bank number does not change. For instance, requests of application A that were mapped to bank 1 of channel 1 would now be mapped to bank 1 of channel 0. What is the stall time of application A using this channel partitioning mechanism and an FR-FCFS memory scheduler?

What is the stall time of application B using this channel partitioning mechanism and an FR-FCFS memory scheduler?

Explain why channel partitioning does better or worse than scheduler Y.

Is channel partitioning and FR-FCFS memory scheduling always strictly better or strictly worse than FR-FCFS memory scheduling alone? Explain.

5 Memory System [80 points]

A machine with a 4 GB DRAM main memory system has 4 channels, 1 rank per channel and 4 banks per rank. The cache block size is 64 bytes.

(a) You are given the following byte addresses and the channel and bank to which they are mapped:

Byte: 0x0000   ⇒   Channel   0,   Bank   0
Byte: 0x0100   ⇒   Channel   0,   Bank   0
Byte: 0x0200   ⇒   Channel   0,   Bank   0
Byte: 0x0400   ⇒   Channel   1,   Bank   0
Byte: 0x0800   ⇒   Channel   2,   Bank   0
Byte: 0x0C00   ⇒   Channel   3,   Bank   0
Byte: 0x1000   ⇒   Channel   0,   Bank   1
Byte: 0x2000   ⇒   Channel   0,   Bank   2
Byte: 0x3000   ⇒   Channel   0,   Bank   3

Determine which bits of the address are used for each of the following address components. Assume row bits are higher order than column bits:

- Byte on bus
  Addr [2 : 0]
- Channel bits (channel bits are contiguous)
  Addr [_____ : _____]
- Bank bits (bank bits are contiguous)
  Addr [_____ : _____]
- Column bits (column bits are contiguous)
  Addr [_____ : _____]
- Row bits (row bits are contiguous)
  Addr [_____ : _____]

(b) Two applications App 1 and App 2 share this memory system (using the address mapping scheme you determined in part (a)). The memory scheduling policy employed is FR-FCFS. The following requests are queued at the memory controller request buffer at time t. Assume the first request (A) is the oldest and the last one (A + 15) is the youngest.

A   B   A + 1   A + 2   A + 3   B + 10   A + 4   B + 12   A + 5   A + 6   A + 7
A + 8   A + 9   A + 10   A + 11   A + 12   A + 13   A + 14   A + 15

These are cache block addresses, not byte addresses. Note that requests to A + x are from App 1, while requests to B + x are from App 2. Addresses A and B are row-aligned (i.e., they are at the start of a row) and are at the same bank but are in different rows.

Assuming row-buffer hits take T time units to service and row-buffer conflicts/misses take 2T time units to service, what is the slowdown (compared to when run alone on the same system) of

- App 1?

- App 2?

(c) Which application slows down more?

Why?

(d) In class, we discussed memory channel partitioning and memory request scheduling as two solutions to mitigate interference and application slowdowns in multicore systems. Propose another solution to reduce the slowdown of the more-slowed-down application, without increasing the slowdown of the other application? Be concrete.