Lecture 4: Memory Scheduling, Refresh

- Topics: scheduling policies, refresh basics

Scheduling Policies Basics

- Must honor several timing constraints for each bank/rank
- Commands: PRE, ACT, COL-RD, COL-WR, REF, Power-Up/Dn
- Must handle reads and writes on the same DDR3 bus
- Must issue refreshes on time
- Must maximize row buffer hit rates and parallelism
- Must maximize throughput and fairness

Address Mapping Policies

- Consecutive cache lines can be placed in the same row to boost row buffer hit rates

- Consecutive cache lines can be placed in different ranks to boost parallelism

- Example address mapping policies:
  row:rank:bank:channel:column:blkoffset

  row:column:rank:bank:channel:blkoffset

Reads and Writes

- A single bus is used for reads and writes
- The bus direction must be reversed when switching between reads and writes; this takes time and leads to bus idling
- Hence, writes are performed in bursts; a write buffer stores pending writes until a high water mark is reached
- Writes are drained until a low water mark is reached

Refresh

- Example, tREFI (gap between refresh commands) = 7.8us, tRFC (time to complete refresh command) = 350 ns
- JEDEC: must issue 8 refresh commands in a 8*tREFI time window
- Allows for some flexibility in when each refresh command is issued
- Elastic refresh: issue a refresh command when there is lull in activity; any unissued refreshes are handled at the end of the 8*tREFI window

Maximizing Row Buffer Hit Rates

- FCFS: Issue the first read or write in the queue that is ready for issue (not necessarily the oldest in program order)

- First Ready - FCFS: First issue row buffer hits if you can

STFM

Mutlu and Moscibroda, MICRO'07

- When multiple threads run together, threads with row buffer hits are prioritized by FR-FCFS
- Each thread has a slowdown: $S = T_{alone} / T_{shared}$, where T is the number of cycles the ROB is stalled waiting for memory
- Unfairness is estimated as S_{max} / S_{min}
- If unfairness is higher than a threshold, thread priorities override other priorities (Stall Time Fair Memory scheduling)
- Estimation of T_{alone} requires some book-keeping: does an access delay critical requests from other threads?

PAR-BS

Mutlu and Moscibroda, ISCA’08

- A batch of requests (per bank) is formed: each thread can only contribute R requests to this batch; batch requests have priority over non-batch requests

- Within a batch, priority is first given to row buffer hits, then to threads with a higher “rank”, then to older requests

- Rank is computed based on the thread’s memory intensity; low-intensity threads are given higher priority; this policy improves batch completion time and overall throughput

- By using rank, requests from a thread are serviced in parallel; hence, parallelism-aware batch scheduling

TCM

Kim et al., MICRO 2010

- Organize threads into latency-sensitive and bw-sensitive clusters based on memory intensity; former gets higher priority

- Within bw-sensitive cluster, priority is based on rank

- Rank is determined based on “niceness” of a thread and the rank is periodically shuffled with insertion shuffling or random shuffling (the former is used if there is a big gap in niceness)

- Threads with low row buffer hit rates and high bank level parallelism are considered “nice” to others

Minimalist Open-Page

Kaseridis et al., MICRO 2011

- Place 4 consecutive cache lines in one bank, then the next 4 in a different bank and so on – provides the best balance between row buffer locality and bank-level parallelism

- Don’t have to worry as much about fairness

- Scheduling first takes priority into account, where priority is determined by wait-time, prefetch distance, and MLP in thread

- A row is precharged after 50 ns, or immediately following a prefetch-dictated large burst

Other Scheduling Ideas

- Using reinforcement learning Ipek et al., ISCA 2008
- Co-ordinating across multiple MCs Kim et al., HPCA 2010
- Co-ordinating requests from GPU and CPU
  Ausavarungnirun et al., ISCA 2012
- Several schedulers in the Memory Scheduling Championship at ISCA 2012
- Predicting the number of row buffer hits Awasthi et al., PACT 2011

Refresh Basics

- A cell is expected to have a retention time of 64ms; every cell must be refreshed within a 64ms window
- The refresh task is broken into 8K refresh operations; a refresh operation is issued every tREFI = 7.8 us
- If you assume that a row of cells on a chip is 8Kb and there are 8 banks, then every refresh operation in a 4Gb chip must handle 8 rows in each bank
- Each refresh operation takes time tRFC = 300ns
- Larger chips have more cells and tRFC will grow

More Refresh Details

- To refresh a row, it needs to be activated and precharged
- Refresh pipeline: the first bank draws the max available current to refresh a row in many subarrays in parallel; each bank is handled sequentially; the process ends with a recovery period to restore charge pumps
- “Row” on the previous slide refers to the size of the available row buffer when you do an Activate; an Activate only deals with some of the subarrays in a bank; refresh performs an activate in all subarrays in a bank, so it can do multiple rows in a bank in parallel

Fine Granularity Refresh

- Will be used in DDR4
- Breaks refresh into small tasks; helps reduce read queuing delays (see example)
- In a future 32Gb chip, tRFC = 640ns, tRFC_2x = 480ns, tRFC_4x = 350ns – note the high overhead from the recovery period

What Makes Refresh Worse

- Refresh operations are issued per rank; LPDDR does allow per bank refresh
- Can refresh all ranks simultaneously – this reduces memory unavailable time, but increases memory peak power
- Can refresh ranks in staggered manner – increases memory unavailable time, but reduces memory peak power
- High temperatures will increase the leakage rate and require faster refresh rates (> 85 degrees C ➔ 3.9us tREFI)

Next Class

- Refresh optimizations: elastic refresh, refresh pausing, smart refresh, Flikker, preemptive command drain, refresh and commands together, etc.

Title

- Bullet