Lecture: Memory Basics and Innovations

• Topics: VM wrap-up, memory organization basics, memory scheduling policies
Superpages

• If a program’s working set size is 16 MB and page size is 8KB, there are 2K frequently accessed pages – a 128-entry TLB will not suffice

• By increasing page size to 128KB, TLB misses will be eliminated – disadvantage: memory waste, increase in page fault penalty

• Can we change page size at run-time?

• Note that a single page has to be contiguous in physical memory
Superpages Implementation

• At run-time, build superpages if you find that contiguous virtual pages are being accessed at the same time

• For example, virtual pages 64-79 may be frequently accessed – coalesce these pages into a single superpage of size 128KB that has a single entry in the TLB

• The physical superpage has to be in contiguous physical memory – the 16 physical pages have to be moved so they are contiguous
Ski Rental Problem

• Promoting a series of contiguous virtual pages into a superpage reduces TLB misses, but has a cost: copying physical memory into contiguous locations

• Page usage statistics can determine if pages are good candidates for superpage promotion, but if cost of a TLB miss is $x$ and cost of copying pages is $Nx$, when do you decide to form a superpage?

• If ski rentals cost $50 and new skis cost $500, when do I decide to buy new skis?
  ➢ If I rent 10 times and then buy skis, I’m guaranteed to not spend more than twice the optimal amount
DRAM Main Memory

- Main memory is stored in DRAM cells that have much higher storage density

- DRAM cells lose their state over time – must be refreshed periodically, hence the name Dynamic

- DRAM access suffers from long access time and high energy overhead
• DIMM: a PCB with DRAM chips on the back and front
• Rank: a collection of DRAM chips that work together to respond to a request and keep the data bus full
• A 64-bit data bus will need 8 x8 DRAM chips or 4 x16 DRAM chips or..
• Bank: a subset of a rank that is busy during one request
• Row buffer: the last row (say, 8 KB) read from a bank, acts like a cache
• DDR standards
DRAM Array Access

16Mb DRAM array = 4096 x 4096 array of bits

12 row address bits arrive first
Row Access Strobe (RAS)

12 column address bits arrive next
Column Access Strobe (CAS)

4096 bits are read out
Column decoder

Some bits returned to CPU
Row Buffer
Organizing a Rank

• DIMM, rank, bank, array → form a hierarchy in the storage organization

• Because of electrical constraints, only a few DIMMs can be attached to a bus

• One DIMM can have 1-4 ranks

• For energy efficiency, use wide-output DRAM chips – better to activate only 4 x16 chips per request than 16 x4 chips

• For high capacity, use narrow-output DRAM chips – since the ranks on a channel are limited, capacity per rank is boosted by having 16 x4 2Gb chips than 4 x16 2Gb chips
Organizing Banks and Arrays

- A rank is split into many banks (8-16) to boost parallelism within a rank

- Ranks and banks offer memory-level parallelism

- A bank is made up of multiple arrays (subarrays, tiles, mats)

- To maximize density, arrays within a bank are made large → rows are wide → row buffers are wide (e.g., 8KB read for a 64B request, called overfetch)
Problem 1

• What is the maximum memory capacity supported by the following server: 2 processor sockets, each socket has 4 memory channels, each channel supports 2 dual-ranked DIMMs, and x4 4Gb DRAM chips?

What is the memory bandwidth available to the server if each memory channel runs at 800 MHz?
Problem 1

- What is the maximum memory capacity supported by the following server: 2 processor sockets, each socket has 4 memory channels, each channel supports 2 dual-ranked DIMMs, and x4 4Gb DRAM chips?

  2 sockets x 4 channels x 2 DIMMs x 2 ranks x 16 chips x 4Gb capacity = 256 GB

- What is the memory bandwidth available to the server if each memory channel runs at 800 MHz?
  2 sockets x 4 channels x 800M (cycles per second) x 2 (DDR, hence 2 transfers per cycle) x 64 (bits per transfer) = 102.4 GB/s
Problem 2

- A basic memory mat has 512 rows and 512 columns. What is the memory chip capacity if there are 512 mats in a bank, and 8 banks in a chip?
Problem 2

• A basic memory mat has 512 rows and 512 columns. What is the memory chip capacity if there are 512 mats in a bank, and 8 banks in a chip?

Memory chip capacity = 512 rows x 512 cols x 512 mats x 8 banks = 1 Gb
Row Buffers

• Each bank has a single row buffer

• Row buffers act as a cache within DRAM
  ➢ Row buffer hit: ~20 ns access time (must only move data from row buffer to pins)
  ➢ Empty row buffer access: ~40 ns (must first read arrays, then move data from row buffer to pins)
  ➢ Row buffer conflict: ~60 ns (must first precharge the bitlines, then read new row, then move data to pins)

• In addition, must wait in the queue (tens of nano-seconds) and incur address/cmd/data transfer delays (~10 ns)
Open/Closed Page Policies

• If an access stream has locality, a row buffer is kept open
  ▪ Row buffer hits are cheap (open-page policy)
  ▪ Row buffer miss is a bank conflict and expensive because precharge is on the critical path

• If an access stream has little locality, bitlines are precharged immediately after access (close-page policy)
  ▪ Nearly every access is a row buffer miss
  ▪ The precharge is usually not on the critical path

• Modern memory controller policies lie somewhere between these two extremes (usually proprietary)
Problem 3

- For the following access stream, estimate the finish times for each access with the following scheduling policies:

<table>
<thead>
<tr>
<th>Req</th>
<th>Time of arrival</th>
<th>Open</th>
<th>Closed</th>
<th>Oracular</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>10 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X+1</td>
<td>100 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X+2</td>
<td>200 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y+1</td>
<td>250 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X+3</td>
<td>300 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that X, X+1, X+2, X+3 map to the same row and Y, Y+1 map to a different row in the same bank. Ignore bus and queuing latencies. The bank is precharged at the start.
Problem 3

- For the following access stream, estimate the finish times for each access with the following scheduling policies:

<table>
<thead>
<tr>
<th>Req</th>
<th>Time of arrival</th>
<th>Open</th>
<th>Closed</th>
<th>Oracular</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0 ns</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Y</td>
<td>10 ns</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>X+1</td>
<td>100 ns</td>
<td>160</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td>X+2</td>
<td>200 ns</td>
<td>220</td>
<td>240</td>
<td>220</td>
</tr>
<tr>
<td>Y+1</td>
<td>250 ns</td>
<td>310</td>
<td>300</td>
<td>290</td>
</tr>
<tr>
<td>X+3</td>
<td>300 ns</td>
<td>370</td>
<td>360</td>
<td>350</td>
</tr>
</tbody>
</table>

Note that X, X+1, X+2, X+3 map to the same row and Y, Y+1 map to a different row in the same bank. Ignore bus and queuing latencies. The bank is precharged at the start.
Problem 4

For the following access stream, estimate the finish times for each access with the following scheduling policies:

<table>
<thead>
<tr>
<th>Req</th>
<th>Time of arrival</th>
<th>Open</th>
<th>Closed</th>
<th>Oracular</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>10 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X+1</td>
<td>15 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>100 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y+1</td>
<td>180 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X+2</td>
<td>190 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y+2</td>
<td>205 ns</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that X, X+1, X+2, X+3 map to the same row and Y, Y+1 map to a different row in the same bank. Ignore bus and queuing latencies. The bank is precharged at the start.
Problem 4

• For the following access stream, estimate the finish times for each access with the following scheduling policies:

<table>
<thead>
<tr>
<th>Req</th>
<th>Time of arrival</th>
<th>Open</th>
<th>Closed</th>
<th>Oracular</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>10 ns</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>X+1</td>
<td>15 ns</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>Y</td>
<td>100 ns</td>
<td>160</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>Y+1</td>
<td>180 ns</td>
<td>200</td>
<td>220</td>
<td>200</td>
</tr>
<tr>
<td>X+2</td>
<td>190 ns</td>
<td>260</td>
<td>300</td>
<td>260 (or 285)</td>
</tr>
<tr>
<td>Y+2</td>
<td>205 ns</td>
<td>320</td>
<td>240</td>
<td>320 (or 225)</td>
</tr>
</tbody>
</table>

Note that X, X+1, X+2, X+3 map to the same row and Y, Y+1 map to a different row in the same bank. Ignore bus and queuing latencies. The bank is precharged at the start.

** A more sophisticated oracle can do even better.