Nonblocking Memory Refresh

Kate Nguyen, Kehan Lyu, Xianze Meng, Vilas Sridharan, Xun Jian
History of DRAM

- 1968: DRAM is patented
- 2000: DDR
- 2003: DDR2
- 2007: DDR3
- 2012: DDR4
- 2018: 50th Anniversary of DRAM patent

Graph showing:
- Refresh Latency
- Bus Cycle Time
- Min. Read Latency

- Latency (ns):
  - 1968: 16 ns
  - 2000: 16 ns
  - 2003: 16 ns
  - 2007: 16 ns
  - 2012: 550 ns
  - 2018: 512 ns

- Bus Cycle Time:
  - 1968: 0.75 ns
  - 2000: 0.75 ns
  - 2003: 0.75 ns
  - 2007: 0.75 ns
  - 2012: 0.5 ns
  - 2018: 0.5 ns

- Min. Read Latency:
  - 1968: 13.5 ns
  - 2000: 13.5 ns
  - 2003: 13.5 ns
  - 2007: 13.5 ns
  - 2012: 13.5 ns
  - 2018: 13.5 ns

Skipping Refresh:
(ISCA ‘12, HPCA ‘13 HPCA ‘14, ISCA ‘15, ISCA ‘17, MICRO ‘17)
Issues with Skipping Refresh

Tested DRAM chips from different manufacturers


Skipping refresh reduces memory security
Why DRAM Refresh Hurts Performance

### DRAM
- **address line**
- **transistor**
- **storage capacitor**
- **bit line**

Blocking Refresh

### SRAM
- **word line**
- Transistors T1, T3, T4, T2
- Transistors T5, T6

Nonblocking Refresh
Our Proposal: Nonblocking Refresh

• Improve performance while retaining the same level of security as the conventional baseline.

• Transform DRAM refresh into the static/background refresh in SRAM at the system level.

• Refresh DRAM in the background without stalling read accesses to refreshing memory blocks.
How Nonblocking Refresh Works

Nonblocking Refresh

- Refreshing Memory Block
- Calculate
- Refreshing Data
- Redundant Data

Conventional Refresh

- Refreshing Memory Block
- Pending read requests to the block are stalled
For server systems, Nonblocking Refresh can leverage existing underutilized redundant data without storage overheads.
Example Memory Rank

Data chip 1  Data chip 2  Data chip 3  Data chip 4  Redundant Chip 1  Redundant Chip 2

Fetched Memory Block from Rank
Nonblocking Refresh for Server Memory

Example Memory Rank

- Data chip 1
- Data chip 2
- Data chip 3
- Data chip 4
- Redundant Chip 1
- Redundant Chip 2

- Inaccessible data due to refresh
- Accessible data

Fetched Memory Block from Rank

Calculate
Challenge 1: Ensuring Same Amount of Refresh

Conventional (blocking) refresh

Refreshing Memory Rank

- Data chip 1
- Data chip 2
- Data chip 3
- Data chip 4
- Redundant Chip 1
- Redundant Chip 2

Time

Chip ID

- Refreshing (inaccessible)
- Not refreshing (accessible)
Challenge 1: Ensuring Same Amount of Refresh

Nonblocking Refresh

Chip ID

Not refreshing (accessible)

Refreshing (inaccessible)

Time

Refreshing Memory Rank

Data chip 1
Data chip 2
Data chip 3
Data chip 4
Redundant Chip 1
Redundant Chip 2
Challenge 1: Ensuring Same Amount of Refresh

Nonblocking Refresh

Refresh Interval

Not refreshing (accessible)

Refreshing (inaccessible)

Refreshing Memory Rank

Data chip1  Data chip 2  Data chip 3  Data chip 4  Redundant Chip 1  Redundant Chip 2
Challenge 1: Ensuring Same Amount of Refresh

Nonblocking Refresh

Time

Refresh Interval

Chip ID

1 2 3 4 5 6

Refreshing (inaccessible)

Not refreshing (accessible)

Data chip 1
Data chip 2
Data chip 3
Data chip 4
Redundant Chip 1
Redundant Chip 2

Refreshing Memory Rank

Data chip 1
Data chip 2
Data chip 3
Data chip 4
Redundant Chip 1
Redundant Chip 2
Challenge 2: Ensuring Memory Write Bandwidth

### Conventional Systems

- **Shared Memory Bus**
- **Write Queue**
- **Rank**
  - Rank 1: 100%
  - Rank 2: 100%
  - Rank N: 100%/N

### Nonblocking Refresh

- **Shared Memory Bus**
- **Write Queue**
- **36 KB/Channel Writeback Cache**
- **Rank 1**
  - 100%
- **Rank 2**
  - 100%
- **Rank N**
  - 0%

Refreshing
Challenge 3: Preserving Baseline Hardware Failure Protection

Read a block from a refreshing rank

Use the block’s existing redundant data:
- to calculate inaccessible data stored in refreshing chips
- to detect unknown hardware errors

Hardware Error detected?

YES
- Wait for refresh to complete

NO
- Re-read block from memory
- Perform error correction

Read completes
Methodology

• Two Memory Systems:
  • Intel/AMD Server Memory Systems
  • IBM Server Memory System
• Baseline:
  • Conventional Refresh: fully compliance with manufacturer specification
  • Insecure Refresh: skips 75% of refresh operations
• Evaluated 7 multi-threaded and 7 multi-program workloads
• 16gb and future 32gb DRAM
• 4 memory channels with 4 ranks per channel
Performance Improvement

Performance Improvement vs. Conventional Refresh

<table>
<thead>
<tr>
<th>Intel/AMD Server Mem</th>
<th>IBM Server Mem</th>
<th>Intel/AMD Server Mem</th>
<th>IBM Server Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>16Gb</td>
<td></td>
<td>32Gb</td>
<td></td>
</tr>
</tbody>
</table>

17
Performance Improvement vs. Insecure Refresh

-10% -8% -6% -4% -2% 0% 2% 4% 6% 8% 10%

Intel/AMD Server Mem
IBM Server Mem

16Gb

Intel/AMD Server Mem
IBM Server Mem

32Gb
Power Consumption

-5%  -3%  -1%   1%   3%   5%   7%   9%

vs. Conventional Refresh  vs. Insecure Refresh

Intel/AMD Server Mem
IBM Server Mem

Power vs. Conventional Refresh vs. Insecure Refresh

Intel/AMD Server Mem
IBM Server Mem

16Gb

32Gb

19
Performance of Systems with Faulty Chips

Nonblocking Refresh on faulty systems vs. on fault-free systems

- Intel/AMD Server Mem
- IBM Server Mem

- 3 Faulty Ranks/Channel
- 2 Faulty Ranks/Channel
- 1 Faulty Rank/Channel
- Average

- 16GB
- 32GB
Conclusion

• Since its invention 50 years ago, DRAM has always required expensive refresh operations that stall accesses to refreshing data.

• We propose Nonblocking Refresh to refresh data in DRAM without stalling read accesses to refreshing data.

• For server memory systems, Nonblocking Refresh improves average performance by 16.2% and 30.3% for 16gb and 32gb chips, respectively.

• Nonblocking Refresh preserves conventional baseline level of security by ensuring the same amount of refresh.
Questions?