RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu Ben Jaiyen Richard Veras Onur Mutlu

SAFARI Carnegie Mellon University

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells
- Problem: All cells refreshed at the same high rate

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells
- Problem: All cells refreshed at the same high rate
- Idea: RAIDR decreases refresh rate for most DRAM cells while refreshing weak cells at a higher rate

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells
- Problem: All cells refreshed at the same high rate
- Idea: RAIDR decreases refresh rate for most DRAM cells while refreshing weak cells at a higher rate
  - Group parts of DRAM into different bins depending on their required refresh rate

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells
- Problem: All cells refreshed at the same high rate
- Idea: RAIDR decreases refresh rate for most DRAM cells while refreshing weak cells at a higher rate
  - Group parts of DRAM into different bins depending on their required refresh rate
    - Use Bloom filters for scalable and efficient binning

Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells
- Problem: All cells refreshed at the same high rate
- Idea: RAIDR decreases refresh rate for most DRAM cells while refreshing weak cells at a higher rate
  - Group parts of DRAM into different bins depending on their required refresh rate
    - Use Bloom filters for scalable and efficient binning
  - Refresh each bin at the minimum rate needed

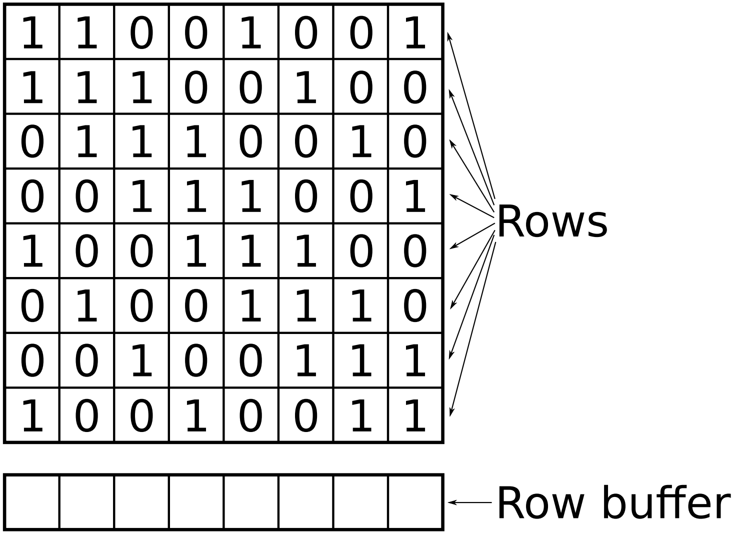
Executive Summary

- DRAM requires periodic refresh to avoid data loss due to capacitor charge leakage
- Refresh operations interfere with memory accesses and waste energy
- Refresh overhead limits DRAM scaling
- Observation: High refresh rate caused by few weak DRAM cells
- Problem: All cells refreshed at the same high rate
- Idea: RAIDR decreases refresh rate for most DRAM cells while refreshing weak cells at a higher rate
  - Group parts of DRAM into different bins depending on their required refresh rate
    - Use Bloom filters for scalable and efficient binning
  - Refresh each bin at the minimum rate needed
- RAIDR reduces refreshes significantly with low overhead in the memory controller

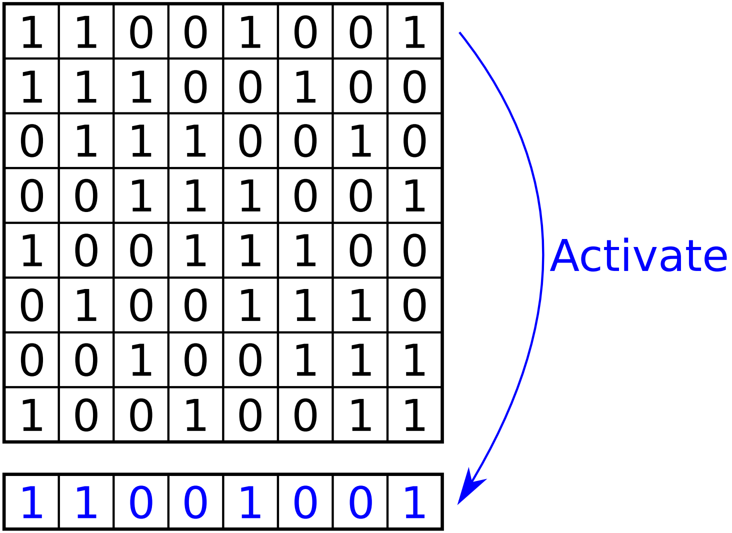
Outline

- Executive Summary
- Background & Motivation
- Key Observation & Our Mechanism: RAIDR
- Evaluation
- Conclusion

DRAM Refresh



DRAM Refresh



1	1	0	0	1	0	0	1
1	1	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1

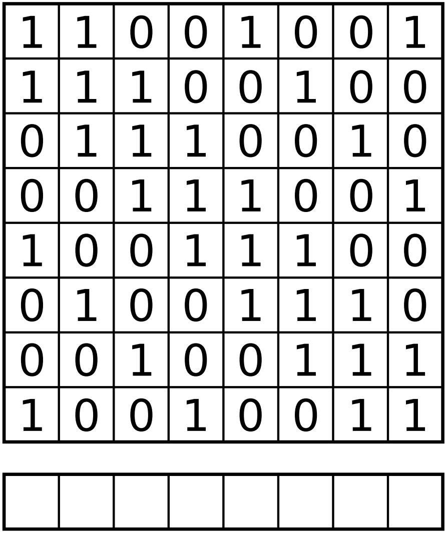
1	1	0	0	1	0	0	1

DRAM Refresh

1	1	0	0	1	0	0	1
1	1	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1

1	1	0	0	1	0	0	1

DRAM Refresh



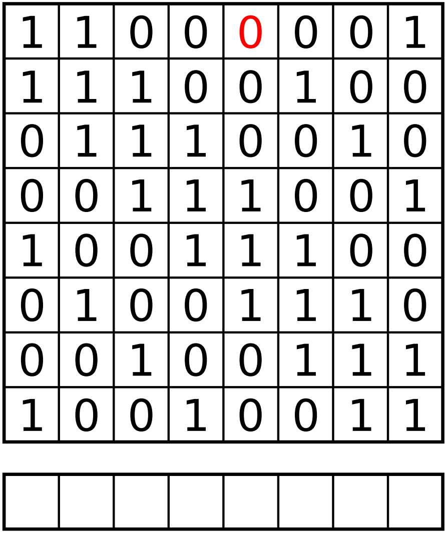
1	1	0	0	1	0	0	1
1	1	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1

Precharge

DRAM Refresh

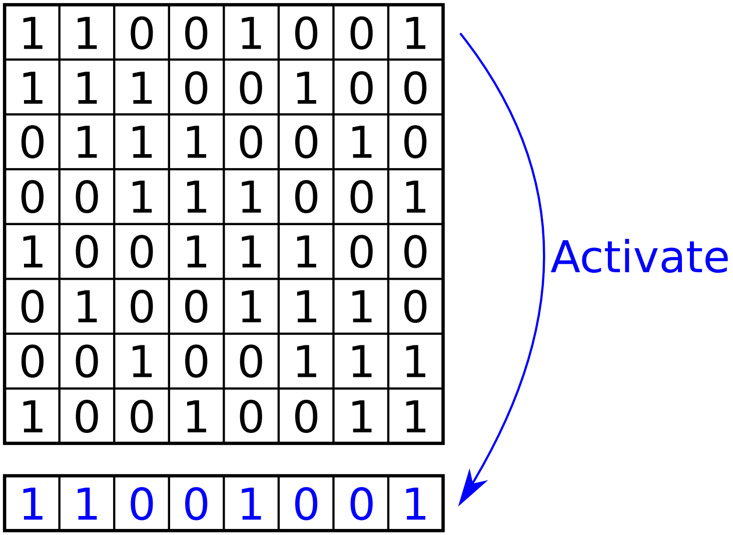
1	1	0	0	1	0	0	1
1	1	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1

DRAM Refresh

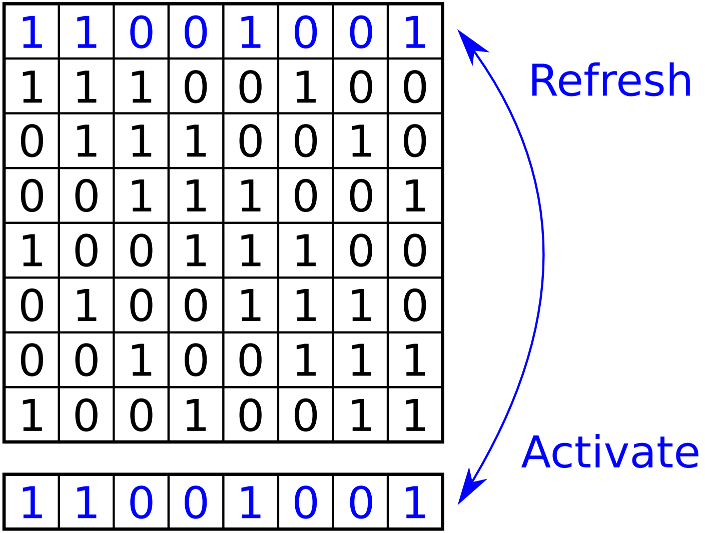


1	1	0	0	0	0	0	1
1	1	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1

DRAM Refresh



DRAM Refresh

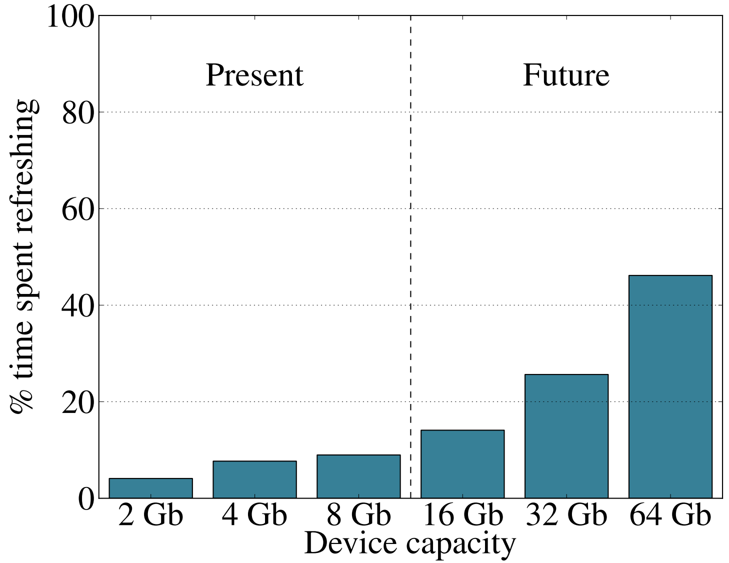


DRAM Refresh

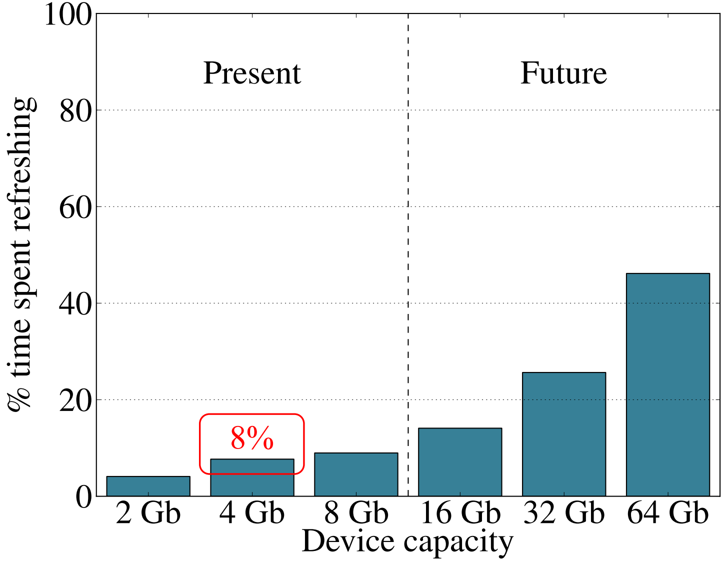
1	1	0	0	1	0	0	1
1	1	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	1
1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	1

Precharge

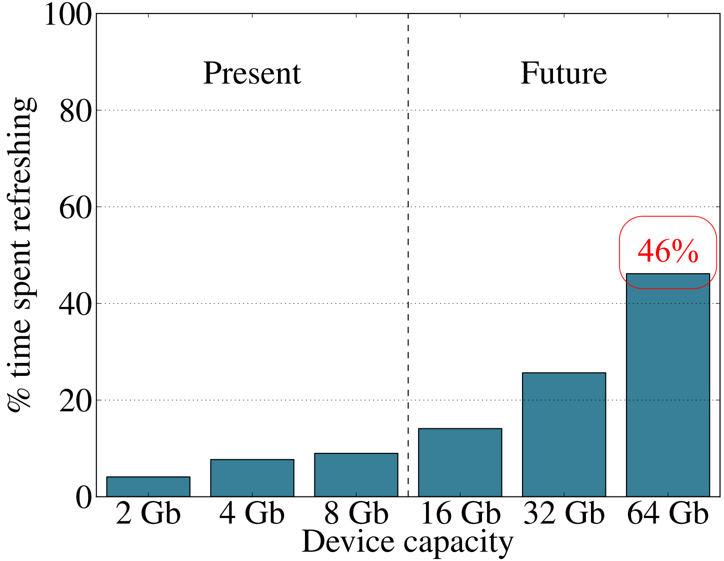
Refresh Overhead: Performance



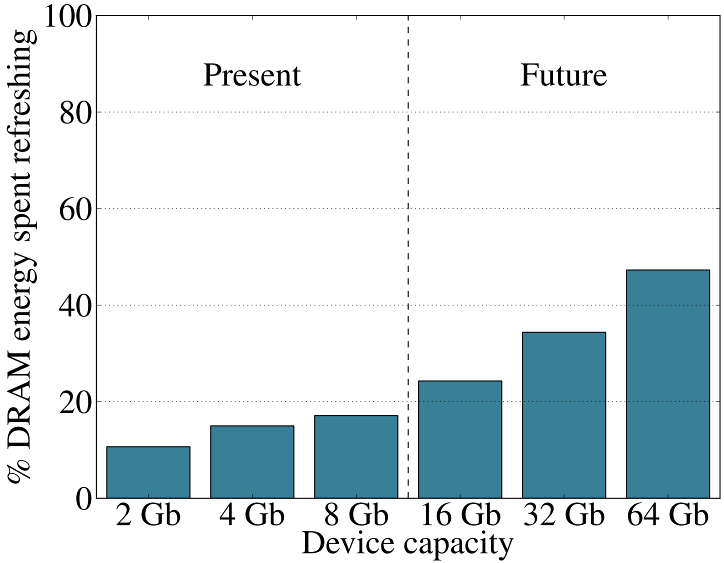
Refresh Overhead: Performance



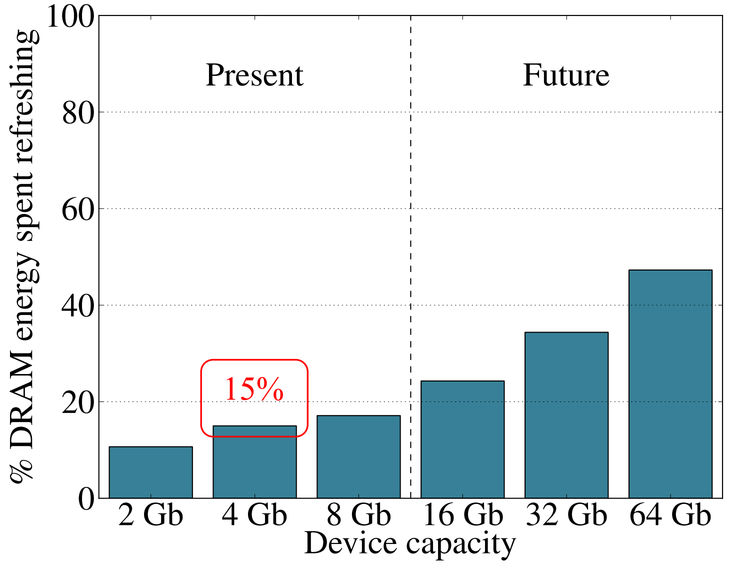
Refresh Overhead: Performance



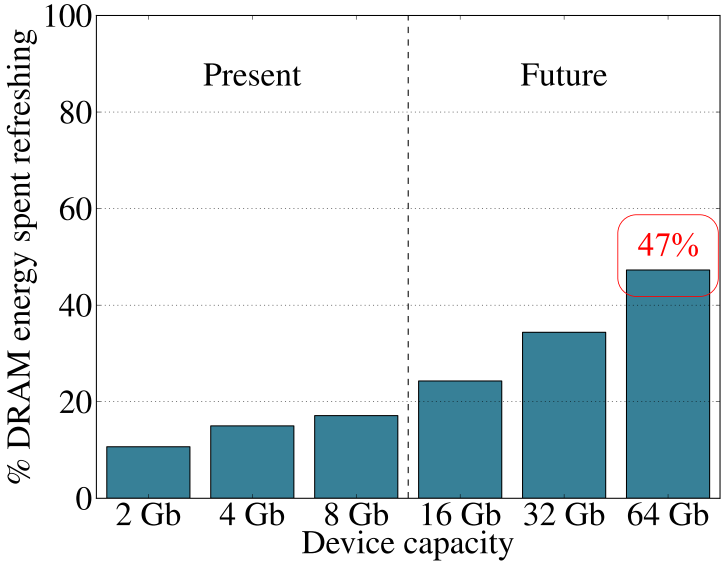
Refresh Overhead: Energy



Refresh Overhead: Energy



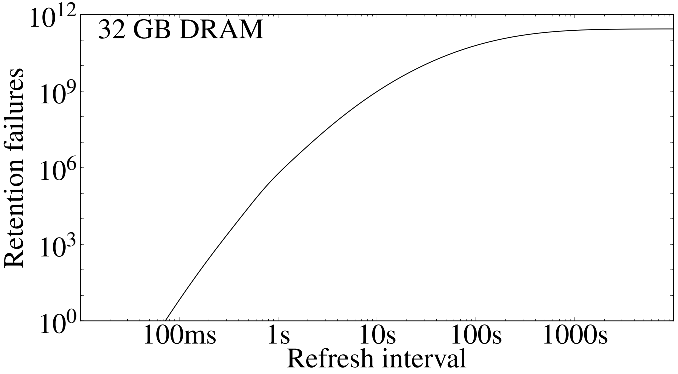
Refresh Overhead: Energy



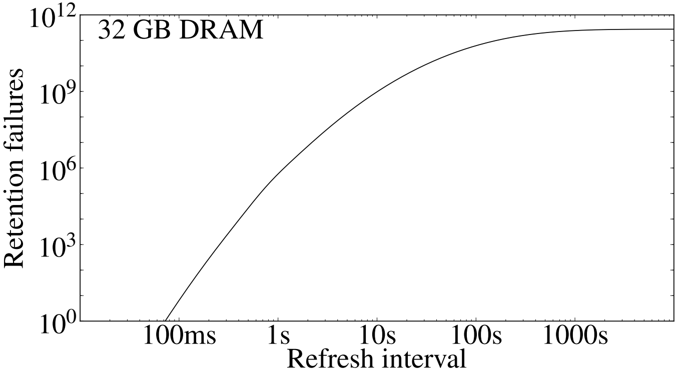
Outline

- Executive Summary
- Background & Motivation
- Key Observation & Our Mechanism: RAIDR
- Evaluation
- Conclusion

Key Observation and Idea

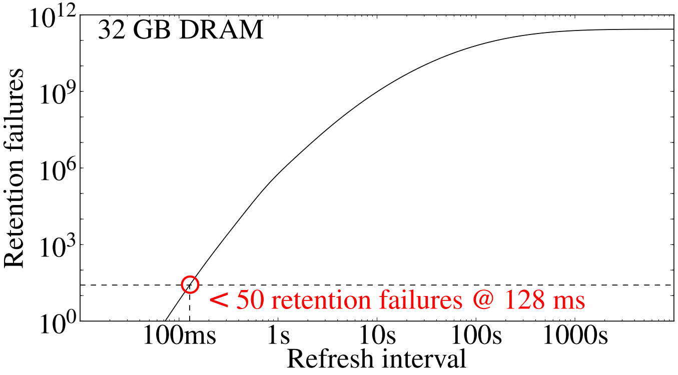


Key Observation and Idea



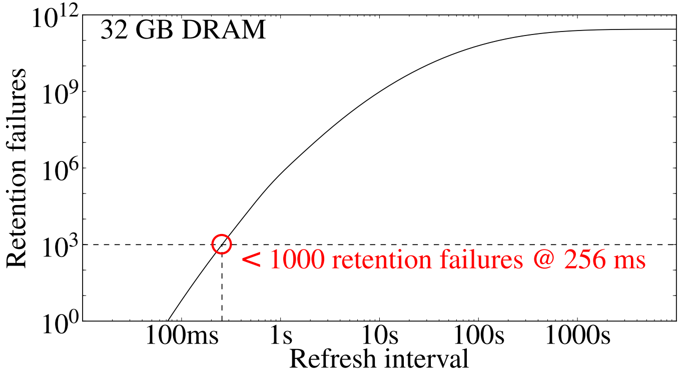
- Key observation: Most cells can be refreshed infrequently without losing data [Kim+, EDL ’09]

Key Observation and Idea



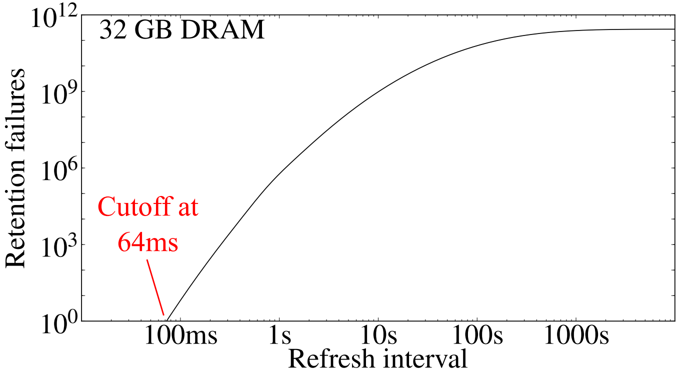
- Key observation: Most cells can be refreshed infrequently without losing data [Kim+, EDL '09]

Key Observation and Idea



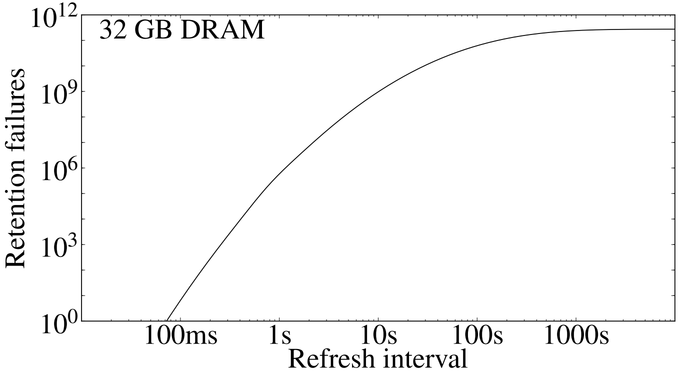
- Key observation: Most cells can be refreshed infrequently without losing data [Kim+, EDL ’09]

Key Observation and Idea



- Key observation: Most cells can be refreshed infrequently without losing data [Kim+, EDL ’09]
- Problem: All cells are refreshed at the same worst-case rate

Key Observation and Idea



- Key observation: Most cells can be refreshed infrequently without losing data [Kim+, EDL '09]
- Problem: All cells are refreshed at the same worst-case rate
- Key idea: refresh rows containing weak cells more frequently; refresh other rows less frequently

Retention-Aware Intelligent DRAM Refresh

1. Profiling
    - Determine each row’s retention time (how frequently each row needs to be refreshed to avoid losing data)

Retention-Aware Intelligent DRAM Refresh

1. Profiling
   - Determine each row's retention time (how frequently each row needs to be refreshed to avoid losing data)

2. Binning
   - Group rows into different retention time bins based on their retention time

Retention-Aware Intelligent DRAM Refresh

1. Profiling
   - Determine each row’s retention time (how frequently each row needs to be refreshed to avoid losing data)

2. Binning
   - Group rows into different retention time bins based on their retention time

3. Refreshing
   - Refresh rows in different bins at different rates

Retention-Aware Intelligent DRAM Refresh

1. Profiling
   - Determine each row's retention time (how frequently each row needs to be refreshed to avoid losing data)

2. Binning
   - Group rows into different retention time bins based on their retention time

3. Refreshing
   - Refresh rows in different bins at different rates

① Retention Time Profiling

- To profile a row:

① Retention Time Profiling

- To profile a row:
  1. Write data to the row

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...
After 128 ms	11011111...	11111111...	11111111...

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...
After 128 ms	11011111...	11111111...	11111111...
	(64–128ms)		

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...
After 128 ms	11011111... (64–128ms)	11111111...	11111111...
After 256 ms		11111011...	11111111...

① Retention Time Profiling

- To profile a row:
  1. Write data to the row
  2. Prevent it from being refreshed
  3. Measure time before data corruption

	Row 1	Row 2	Row 3
Initially	11111111...	11111111...	11111111...
After 64 ms	11111111...	11111111...	11111111...
After 128 ms	11011111... (64–128ms)	11111111...	11111111...
After 256 ms		11111011... (128–256ms)	11111111... (>256ms)

Retention-Aware Intelligent DRAM Refresh

1. Profiling
   - Determine each row’s retention time (how frequently each row needs to be refreshed to avoid losing data)
2. Binning
   - Group rows into different retention time bins based on their retention time
3. Refreshing
   - Refresh rows in different bins at different rates

② Grouping Rows Into Retention Time Bins

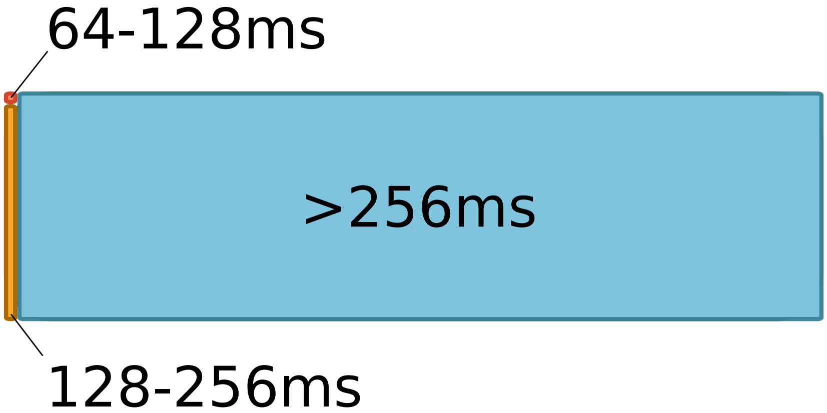
- Rows are grouped into different bins based on their profiled retention time

② Grouping Rows Into Retention Time Bins

- Rows are grouped into different bins based on their profiled retention time

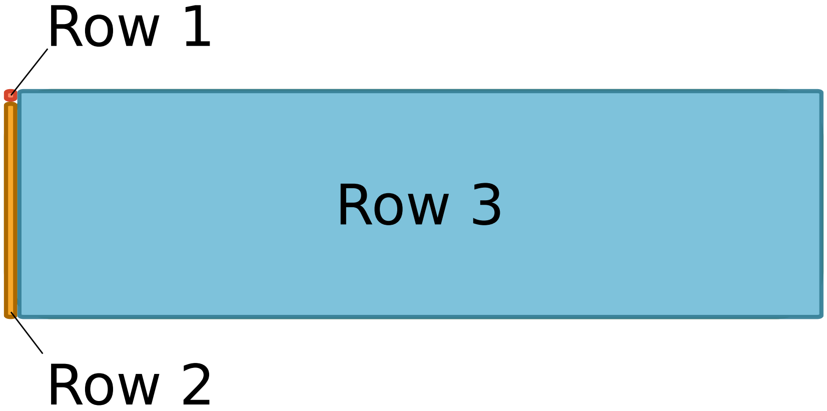
② Grouping Rows Into Retention Time Bins

- Rows are grouped into different bins based on their profiled retention time



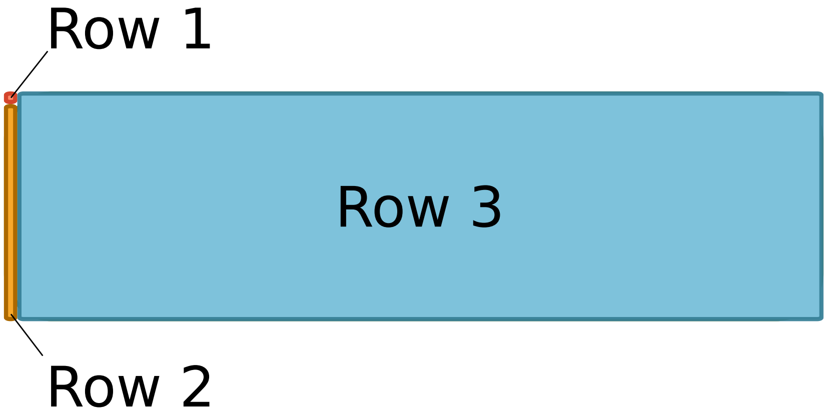
② Grouping Rows Into Retention Time Bins

- Rows are grouped into different bins based on their profiled retention time



② Grouping Rows Into Retention Time Bins

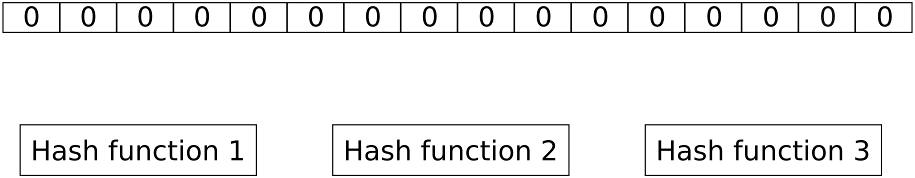
- Rows are grouped into different bins based on their profiled retention time



- Store bins using Bloom filters [Bloom, CACM ’70]

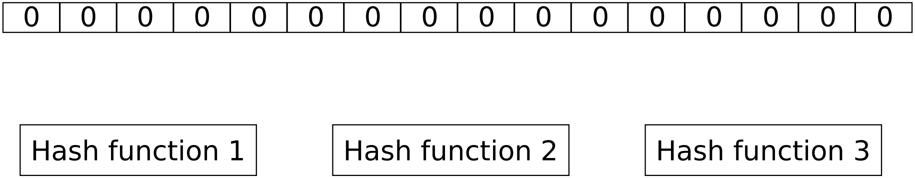
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



Storing Retention Time Bins Using Bloom Filters

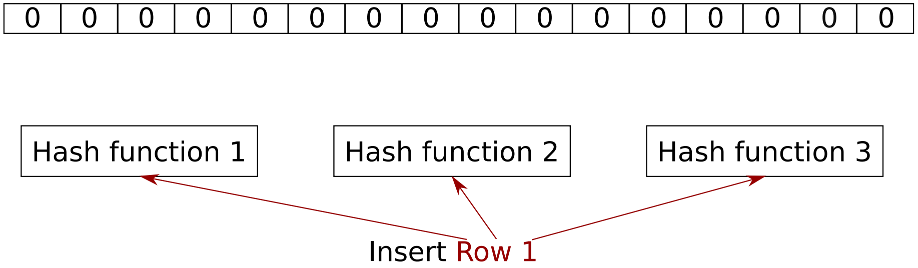
Example with 64–128ms bin:



Insert Row 1

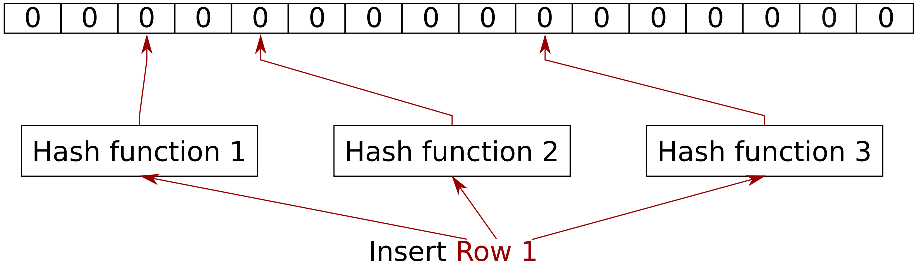
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



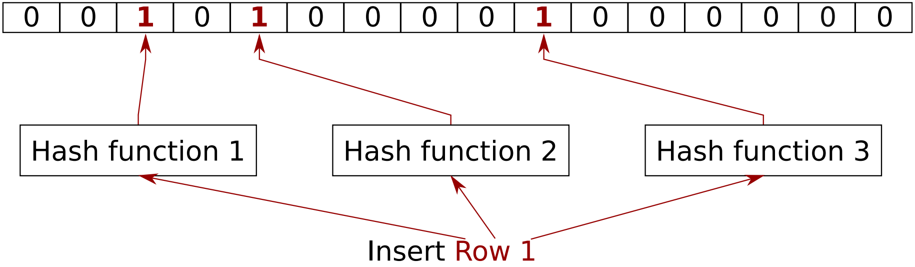
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



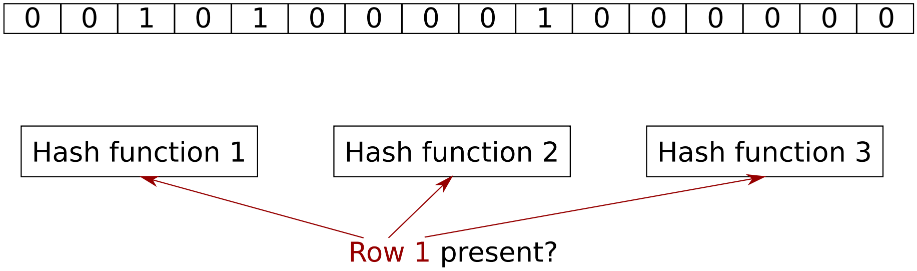
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



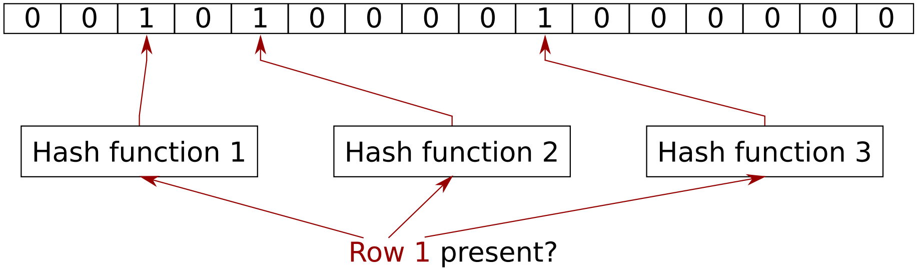
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



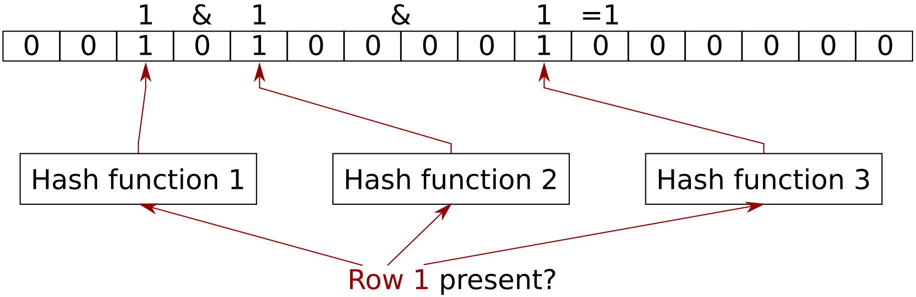
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

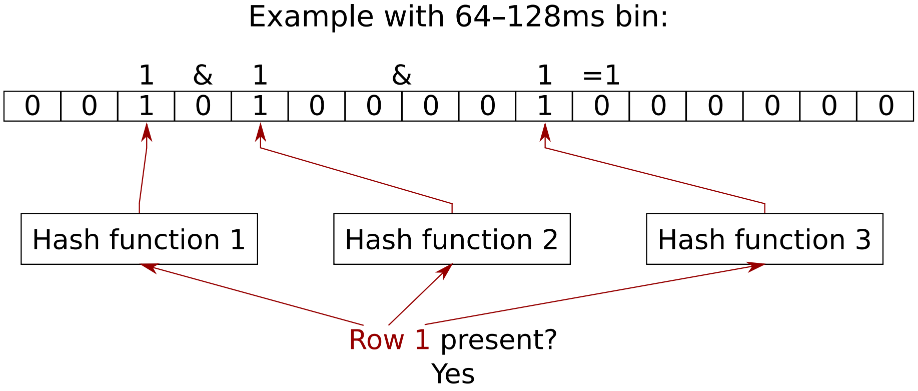


Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



Storing Retention Time Bins Using Bloom Filters



Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0

Hash function 1

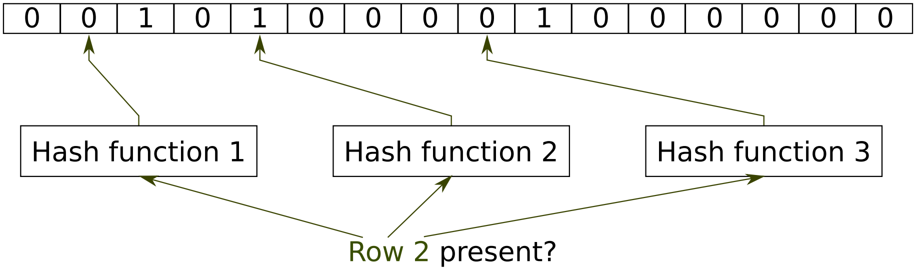
Hash function 2

Hash function 3

Row 2 present?

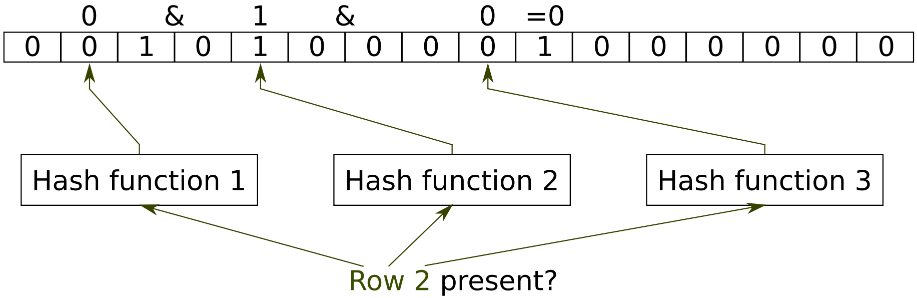
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



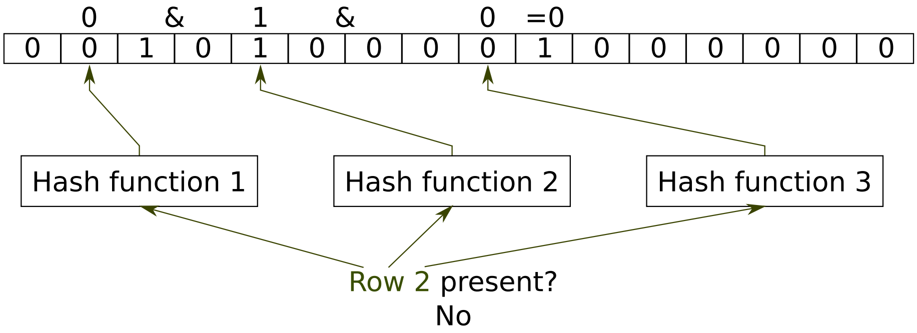
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



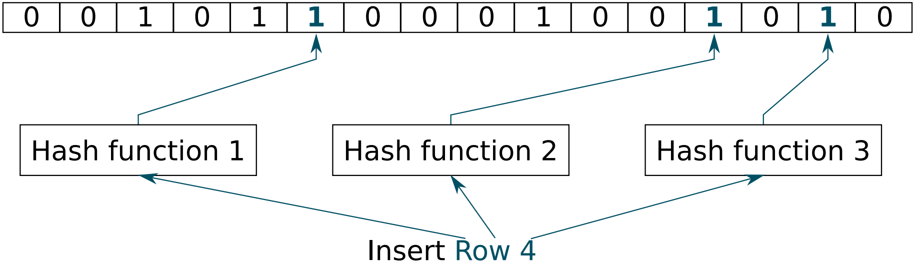
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



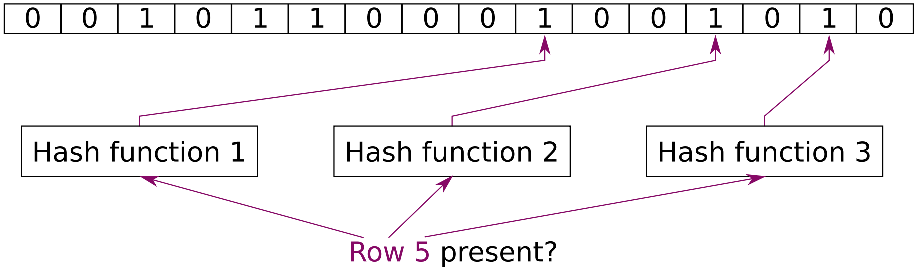
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



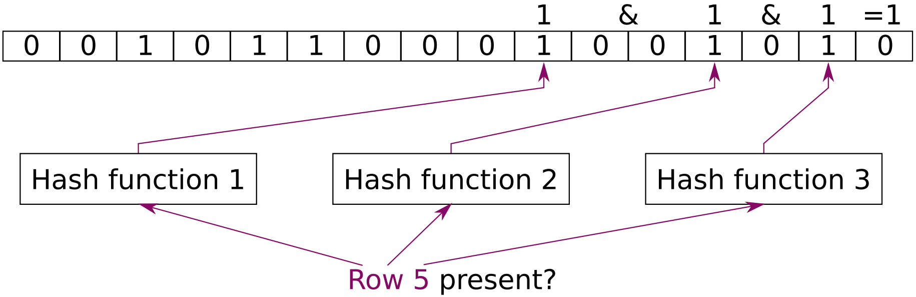
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



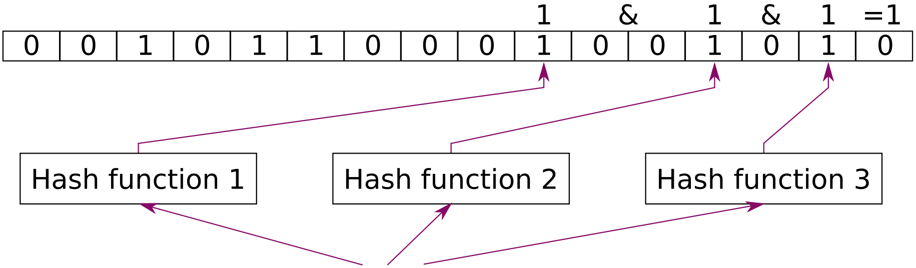
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:



Row 5 present?
Yes (false positive)

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed
- No false negatives: a row will always be declared present in the Bloom filter if it was inserted

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed
- No false negatives: a row will always be declared present in the Bloom filter if it was inserted
  - No correctness problems: Rows are never refreshed less frequently than needed

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed
- No false negatives: a row will always be declared present in the Bloom filter if it was inserted
  - No correctness problems: Rows are never refreshed less frequently than needed
- No overflow: any number of rows may be inserted into a Bloom filter

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed
- No false negatives: a row will always be declared present in the Bloom filter if it was inserted
  - No correctness problems: Rows are never refreshed less frequently than needed
- No overflow: any number of rows may be inserted into a Bloom filter
  - Scalable: contrast with straightforward table implementation

Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed
- No false negatives: a row will always be declared present in the Bloom filter if it was inserted
  - No correctness problems: Rows are never refreshed less frequently than needed
- No overflow: any number of rows may be inserted into a Bloom filter
  - Scalable: contrast with straightforward table implementation
- Bloom filters allow implementation of retention time bins with low hardware overhead

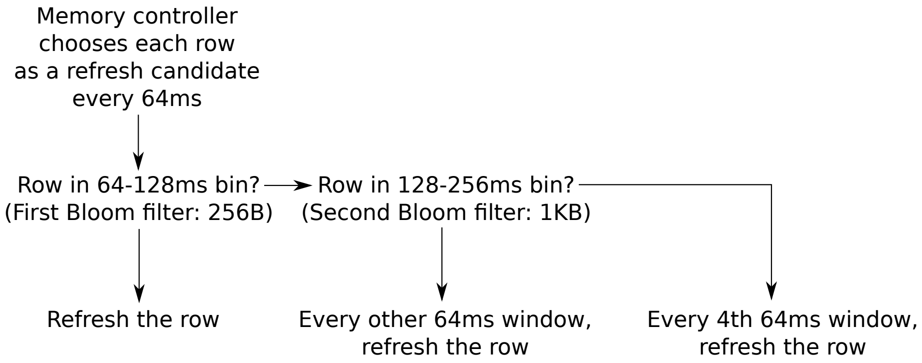
Bloom Filters: Key Characteristics

- False positives: a row may be declared present in the Bloom filter even if it was never inserted
  - Not a problem: Rows may be refreshed more frequently than needed
- No false negatives: a row will always be declared present in the Bloom filter if it was inserted
  - No correctness problems: Rows are never refreshed less frequently than needed
- No overflow: any number of rows may be inserted into a Bloom filter
  - Scalable: contrast with straightforward table implementation
- Bloom filters allow implementation of retention time bins with low hardware overhead
  - 1.25 KB storage overhead (2 Bloom filters) for 32 GB DRAM system

Retention-Aware Intelligent DRAM Refresh

1. Profiling
   - Determine each row’s retention time (how frequently each row needs to be refreshed to avoid losing data)
2. Binning
   - Group rows into different retention time bins based on their retention time
3. Refreshing
   - Refresh rows in different bins at different rates

③ Refreshing Rows at Different Rates


Memory controller chooses each row as a refresh candidate every 64ms
Row in 64-128ms bin? (First Bloom filter: 256B)
Row in 128-256ms bin? (Second Bloom filter: 1KB)
Refresh the row
Every other 64ms window, refresh the row
Every 4th 64ms window, refresh the row

Tolerating Temperature Variation: Refresh Rate Scaling

- Change in temperature causes retention time of all cells to change by a uniform and predictable factor

Tolerating Temperature Variation: Refresh Rate Scaling

- Change in temperature causes retention time of all cells to change by a uniform and predictable factor
- Refresh rate scaling: increase the refresh rate for all rows uniformly, depending on the temperature

Tolerating Temperature Variation: Refresh Rate Scaling

- Change in temperature causes retention time of all cells to change by a uniform and predictable factor

- Refresh rate scaling: increase the refresh rate for all rows uniformly, depending on the temperature

- Implementation: counter with programmable period

Tolerating Temperature Variation: Refresh Rate Scaling

- Change in temperature causes retention time of all cells to change by a uniform and predictable factor

- Refresh rate scaling: increase the refresh rate for all rows uniformly, depending on the temperature

- Implementation: counter with programmable period
  - Lower temperature $\Rightarrow$ longer period $\Rightarrow$ less frequent refreshes

Tolerating Temperature Variation: Refresh Rate Scaling

- Change in temperature causes retention time of all cells to change by a uniform and predictable factor
- Refresh rate scaling: increase the refresh rate for all rows uniformly, depending on the temperature
- Implementation: counter with programmable period
  - Lower temperature $\Rightarrow$ longer period $\Rightarrow$ less frequent refreshes
  - Higher temperature $\Rightarrow$ shorter period $\Rightarrow$ more frequent refreshes

Outline

- Executive Summary
- Background & Motivation
- Key Observation & Our Mechanism: RAIDR
- Evaluation
- Conclusion

Methodology

- 8-core, 4 GHz, 512 KB 16-way private cache per core
- 32 GB DDR3 DRAM system (2 channels, 4 ranks/channel)
- 1.25 KB storage overhead for 2 Bloom filters
- Extended temperature range (85–95°C) characteristic of server environments
- SPEC CPU2006, TPC-C, TPC-H benchmarks in 8-core multiprogrammed workloads
  - Benchmarks categorized by memory intensity (LLC misses per 1000 instructions)
  - Workloads categorized by fraction of memory-intensive benchmarks
  - 32 workloads per category, 5 workload categories

Comparison Points

- Auto-refresh [DDR3, LPDDR2, . . .]:
  - Memory controller periodically sends auto-refresh commands
  - DRAM devices refresh many rows on each command
  - Baseline typical in modern systems
  - All rows refreshed at same rate

Comparison Points

- Auto-refresh [DDR3, LPDDR2, . . .]:
  - Memory controller periodically sends auto-refresh commands
  - DRAM devices refresh many rows on each command
  - Baseline typical in modern systems
  - All rows refreshed at same rate
- Distributed refresh:
  - Memory controller refreshes each row individually by sending activate and precharge commands to DRAM
  - All rows refreshed at same rate

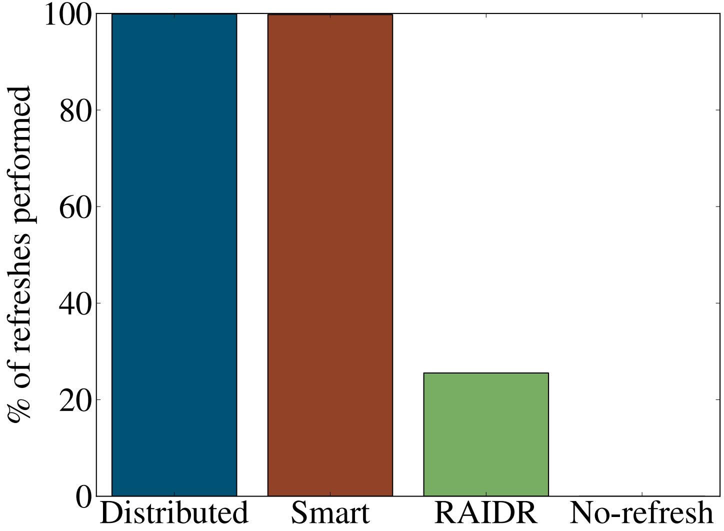
Comparison Points

- Auto-refresh [DDR3, LPDDR2, . . .]:
  - Memory controller periodically sends auto-refresh commands
  - DRAM devices refresh many rows on each command
  - Baseline typical in modern systems
  - All rows refreshed at same rate
- Distributed refresh:
  - Memory controller refreshes each row individually by sending activate and precharge commands to DRAM
  - All rows refreshed at same rate
- Smart Refresh [Ghosh+, MICRO ’07]:
  - Memory controller refreshes each row individually
  - Refreshes to recently activated rows are skipped
  - Requires programs to activate many rows to be effective
  - Very high storage overhead (1.5 MB for 32 GB DRAM)

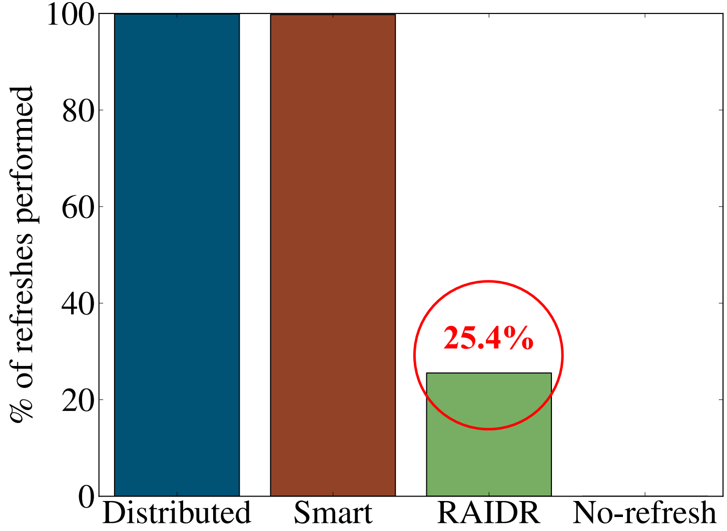
Comparison Points

- Auto-refresh [DDR3, LPDDR2, . . .]:
  - Memory controller periodically sends auto-refresh commands
  - DRAM devices refresh many rows on each command
  - Baseline typical in modern systems
  - All rows refreshed at same rate
- Distributed refresh:
  - Memory controller refreshes each row individually by sending activate and precharge commands to DRAM
  - All rows refreshed at same rate
- Smart Refresh [Ghosh+, MICRO ’07]:
  - Memory controller refreshes each row individually
  - Refreshes to recently activated rows are skipped
  - Requires programs to activate many rows to be effective
  - Very high storage overhead (1.5 MB for 32 GB DRAM)
- No refresh (ideal)

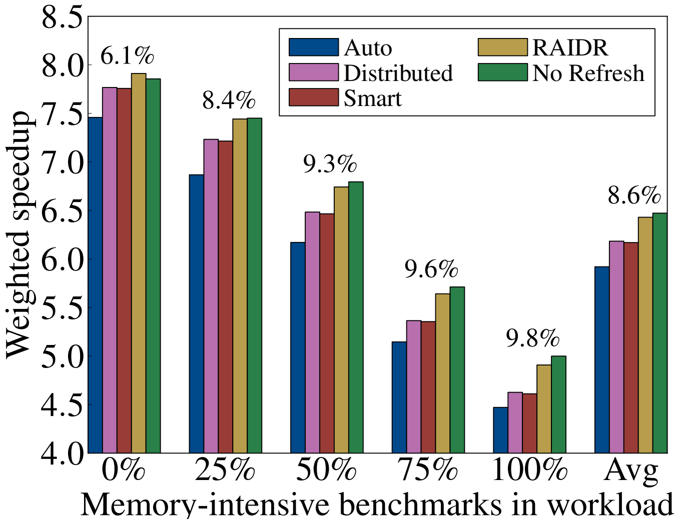
Refresh Operations Performed (32 GB DRAM)



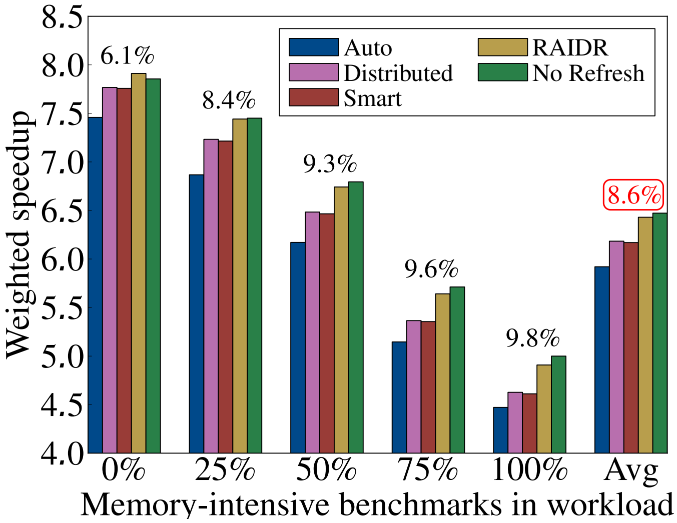
Refresh Operations Performed (32 GB DRAM)



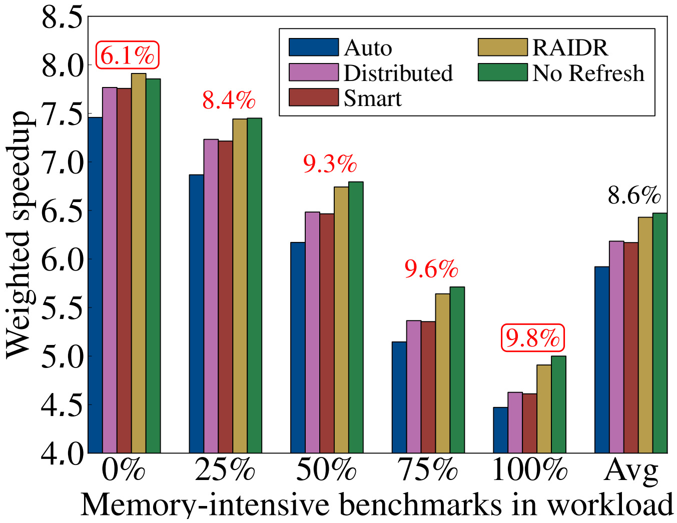
Performance



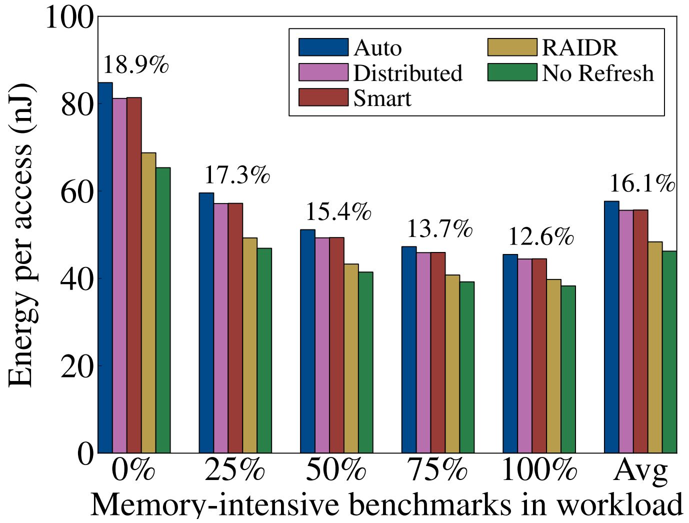
Performance



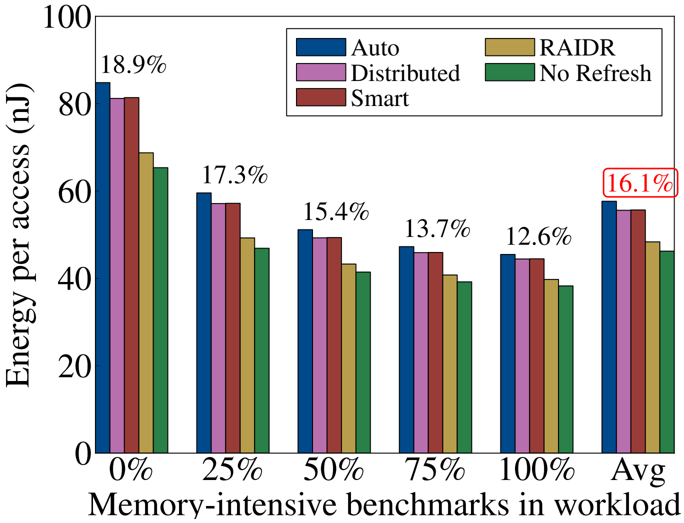
Performance



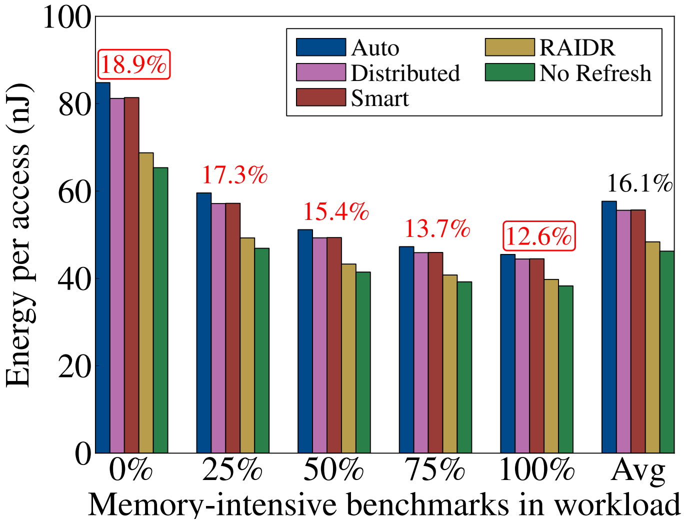
DRAM Energy Efficiency



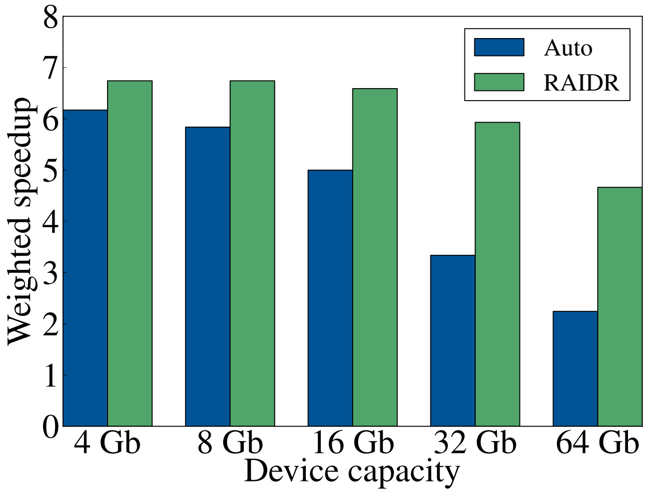
DRAM Energy Efficiency



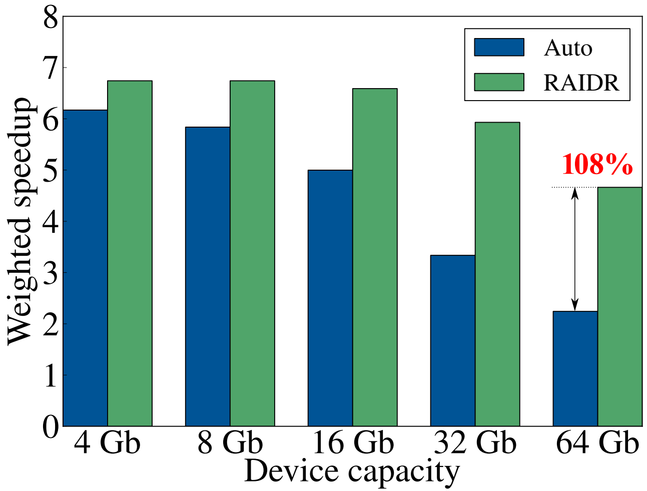
DRAM Energy Efficiency



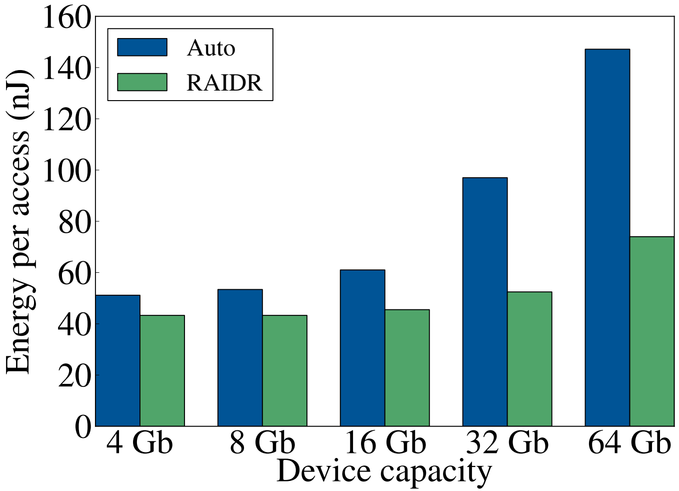
DRAM Device Capacity Scaling: Performance



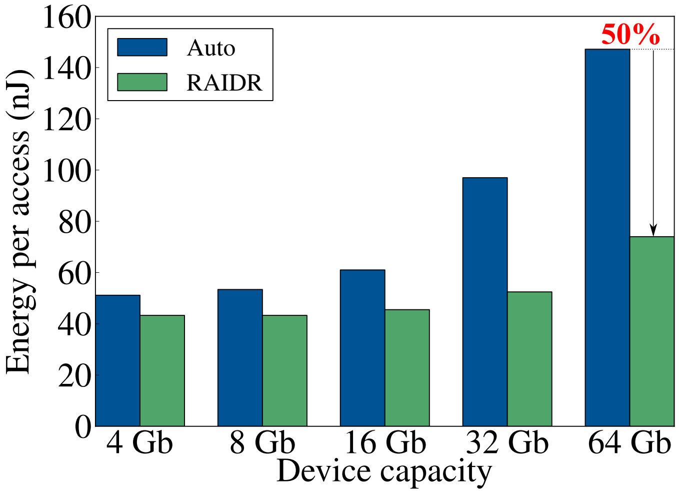
DRAM Device Capacity Scaling: Performance



DRAM Device Capacity Scaling: Energy



DRAM Device Capacity Scaling: Energy



Outline

- Executive Summary
- Background & Motivation
- Key Observation & Our Mechanism: RAIDR
- Evaluation
- Conclusion

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
  - Refresh limits DRAM scaling

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
    - Refresh limits DRAM scaling
- High refresh rate is caused by a small number of problematic cells

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
  - Refresh limits DRAM scaling
- High refresh rate is caused by a small number of problematic cells
- RAIDR groups rows into bins and refreshes rows in different bins at different rates

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
  - Refresh limits DRAM scaling
- High refresh rate is caused by a small number of problematic cells
- RAIDR groups rows into bins and refreshes rows in different bins at different rates
  - Uses Bloom filters for scalable and efficient binning

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
  - Refresh limits DRAM scaling
- High refresh rate is caused by a small number of problematic cells
- RAIDR groups rows into bins and refreshes rows in different bins at different rates
  - Uses Bloom filters for scalable and efficient binning
- 74.6% refresh reduction, 8.6% performance improvement, 16.1% DRAM energy reduction at 1.25 KB overhead

Conclusion

- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
  - Refresh limits DRAM scaling
- High refresh rate is caused by a small number of problematic cells
- RAIDR groups rows into bins and refreshes rows in different bins at different rates
  - Uses Bloom filters for scalable and efficient binning
- 74.6% refresh reduction, 8.6% performance improvement, 16.1% DRAM energy reduction at 1.25 KB overhead
- RAIDR's benefits improve with increasing DRAM density

Conclusion

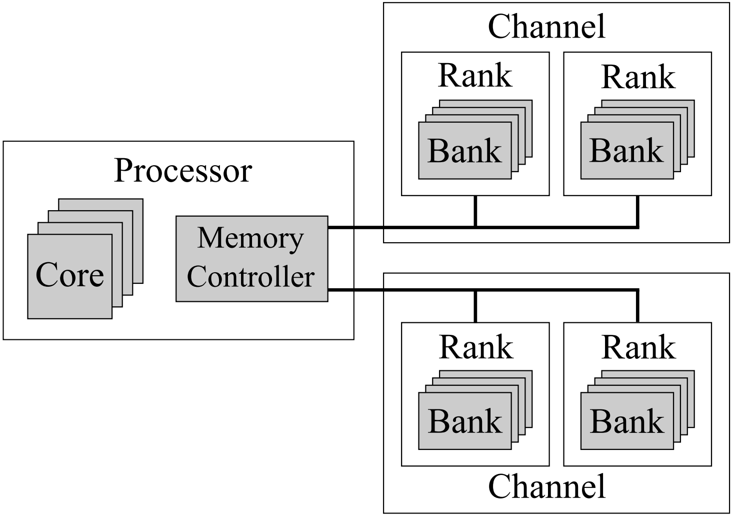
- Refresh is an energy and performance problem that is becoming increasingly significant in DRAM systems
  - Refresh limits DRAM scaling
- High refresh rate is caused by a small number of problematic cells
- RAIDR groups rows into bins and refreshes rows in different bins at different rates
  - Uses Bloom filters for scalable and efficient binning
- 74.6% refresh reduction, 8.6% performance improvement, 16.1% DRAM energy reduction at 1.25 KB overhead
- RAIDR's benefits improve with increasing DRAM density
  - Enables better DRAM scaling

RAIDR:
Retention-Aware Intelligent DRAM Refresh

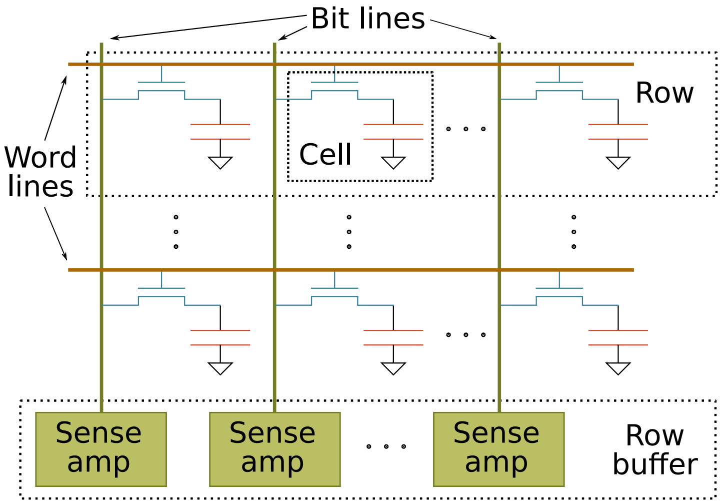
Jamie Liu   Ben Jaiyen   Richard Veras   Onur Mutlu

SAFARI   Carnegie Mellon University

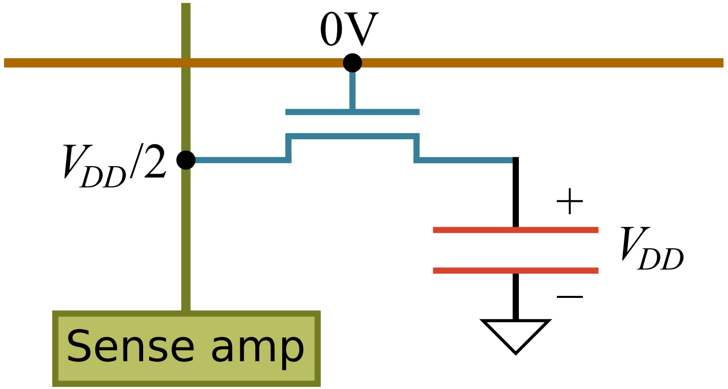
DRAM Hierarchy



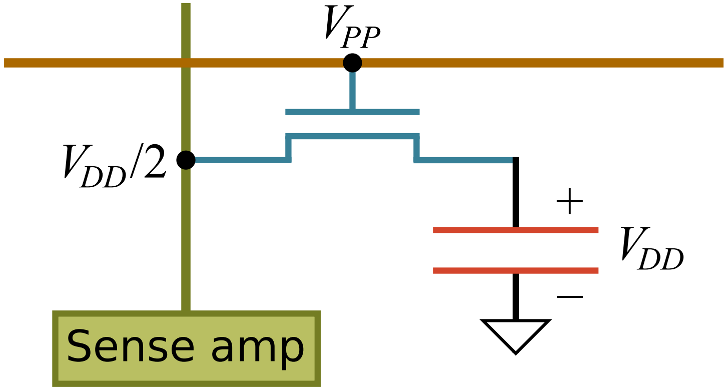
DRAM Array Organization



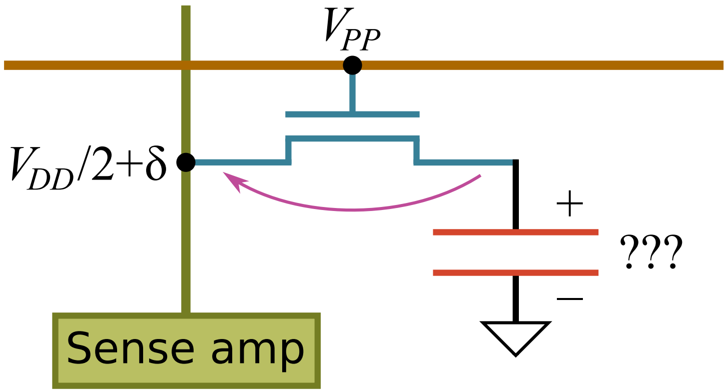
DRAM Activation



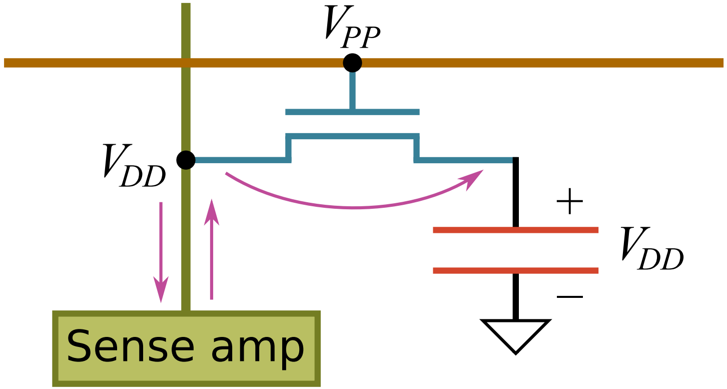
DRAM Activation



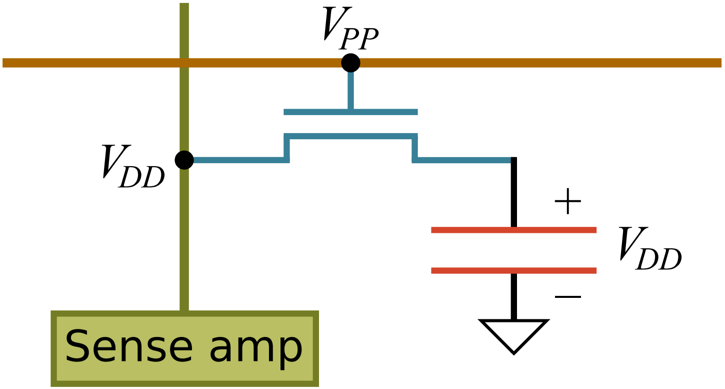
DRAM Activation



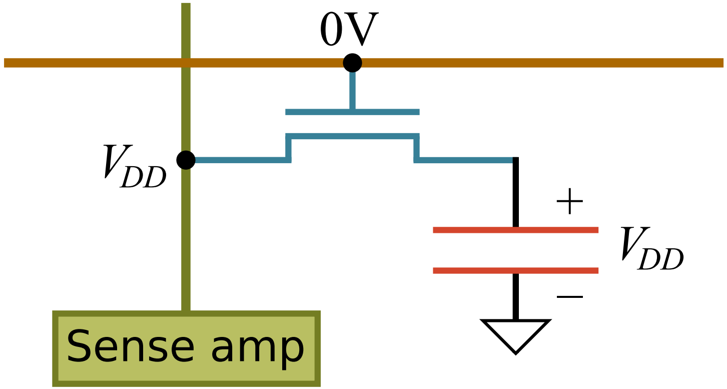
DRAM Activation



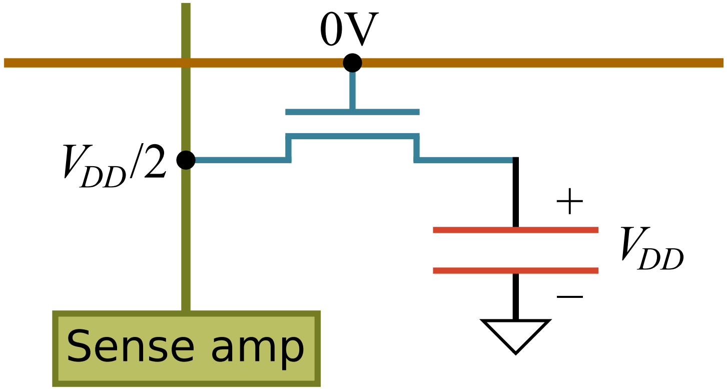
DRAM Precharge



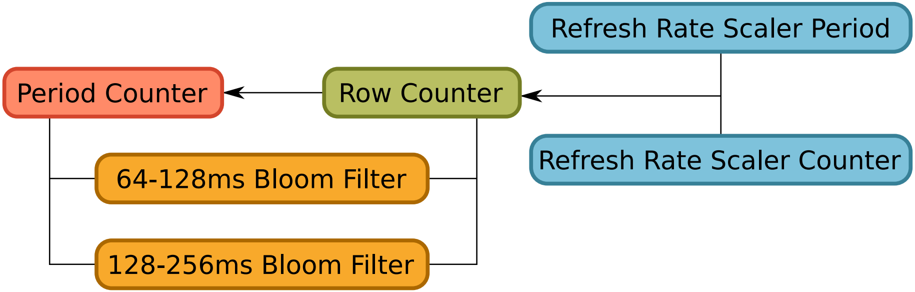
DRAM Precharge



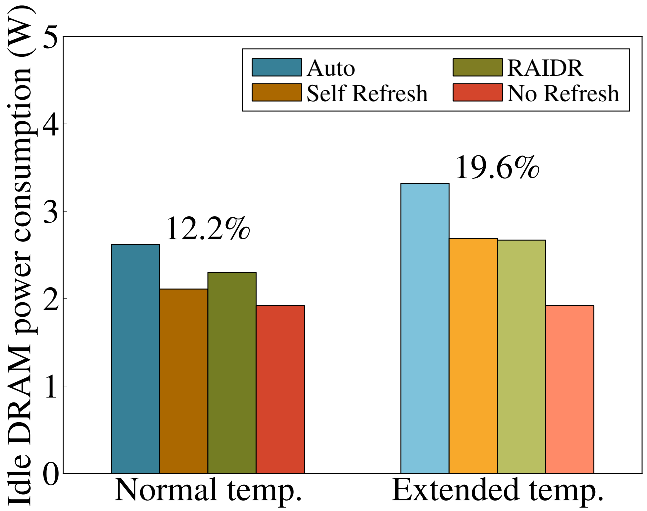
DRAM Precharge



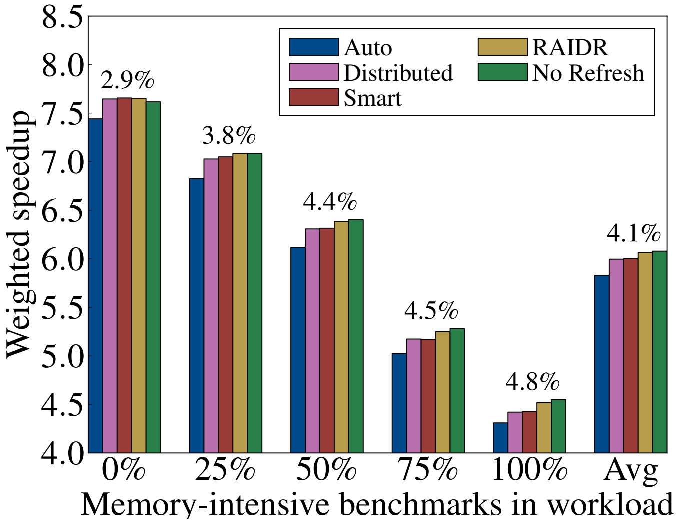
RAIDR Components



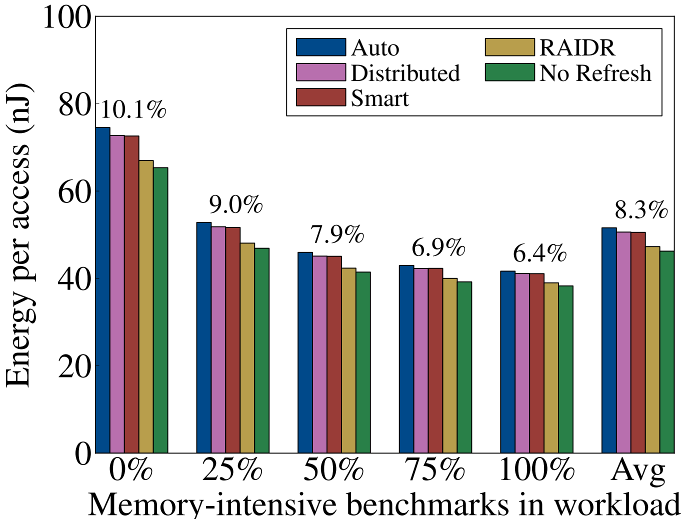
Idle Power Consumption



Performance: 85°C



Energy: 85°C

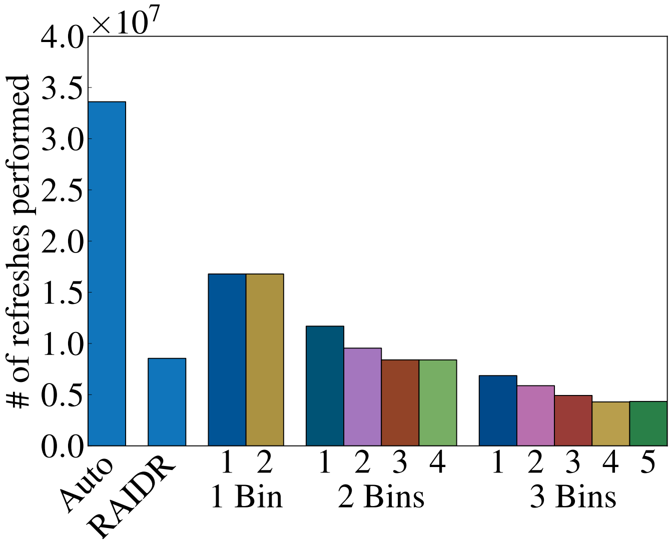


100
80
60
40
20
0
Energy per access (nJ)
Auto
Distributed
Smart
RAIDR
No Refresh
10.1%
9.0%
7.9%
6.9%
6.4%
8.3%
0%
25%
50%
75%
100%
Avg
Memory-intensive benchmarks in workload

RAIDR Default Configuration

- 64–128 ms bin: 256 B Bloom filter, 10 hash functions; 28 rows in bin, false positive probability $1.16 \cdot 10^{-9}$
- 128–256 ms bin: 1 KB Bloom filter, 6 hash functions; 978 rows in bin, false positive probability 0.0179

Refresh Reduction vs. RAIDR Configuration



RAIDR Configurations

Key	Description	Storage Overhead
Auto	Auto-refresh	N/A
RAIDR	Default RAIDR: 2 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 8192$)	1.25 KB
1 bin (1)	1 bin (64–128 ms, $m = 512$)	64 B
1 bin (2)	1 bin (64–128 ms, $m = 1024$)	128 B
2 bins (1)	2 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 2048$)	512 B
2 bins (2)	2 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 4096$)	768 B
2 bins (3)	2 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 16384$)	2.25 KB
2 bins (4)	2 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 32768$)	4.25 KB
3 bins (1)	3 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 8192$; 256–512 ms, $m = 32768$)	5.25 KB
3 bins (2)	3 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 8192$; 256–512 ms, $m = 65536$)	9.25 KB
3 bins (3)	3 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 8192$; 256–512 ms, $m = 131072$)	17.25 KB
3 bins (4)	3 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 8192$; 256–512 ms, $m = 262144$)	33.25 KB
3 bins (5)	3 bins (64–128 ms, $m = 2048$; 128–256 ms, $m = 8192$; 256–512 ms, $m = 524288$)	65.25 KB