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

- 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
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

RAIDR: Retention-Aware Intelligent DRAM Refresh
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

RAIDR: Retention-Aware Intelligent DRAM Refresh
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

RAIDR: Retention-Aware Intelligent DRAM Refresh
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

```
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
```

Rows

```
Row buffer
```

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Refresh

Activate

RAIDR: Retention-Aware Intelligent DRAM Refresh
### DRAM Refresh

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### RAIDR: Retention-Aware Intelligent 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
### DRAM Refresh

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**RAIDR: Retention-Aware Intelligent DRAM Refresh**
### DRAM Refresh

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RAIDR:** Retention-Aware Intelligent DRAM Refresh
### DRAM Refresh

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

RAIDR: Retention-Aware Intelligent 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
```

Refresh

Activate

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Refresh

Precharge

RAIDR: Retention-Aware Intelligent DRAM Refresh
Refresh Overhead: Performance

![Bar chart showing the percentage of time spent refreshing for different device capacities (2 Gb, 4 Gb, 8 Gb, 16 Gb, 32 Gb, 64 Gb) in the present and future. The chart indicates a significant increase in the time spent refreshing as device capacity increases. The future shows a higher percentage of time spent refreshing compared to the present.]

RAIDR: Retention-Aware Intelligent DRAM Refresh
Refresh Overhead: Performance

RAIDR: Retention-Aware Intelligent DRAM Refresh
Refresh Overhead: Energy

% DRAM energy spent refreshing

Present Future

Device capacity

2 Gb 4 Gb 8 Gb 16 Gb 32 Gb 64 Gb

RAIDR: Retention-Aware Intelligent DRAM Refresh
Refresh Overhead: Energy

RAIDR: Retention-Aware Intelligent DRAM Refresh
Refresh Overhead: Energy

RAIDR: Retention-Aware Intelligent DRAM Refresh
Outline

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

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

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

RAIDR: Retention-Aware Intelligent DRAM Refresh
Key Observation and Idea

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

![Graph showing retention failures over refresh intervals for 32 GB DRAM]
Key Observation and Idea

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

![Graph showing retention failures over refresh intervals]

- 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
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

32 GB DRAM

Retention failures

Refresh interval

<1000 retention failures @ 256 ms
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

RAIDR: Retention-Aware Intelligent DRAM Refresh
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. 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
① Retention Time Profiling

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

<table>
<thead>
<tr>
<th></th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>After 64 ms</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>111111111...</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 64 ms</td>
<td>111111111...</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
<tr>
<td>After 128 ms</td>
<td>110111111...</td>
<td>111111111...</td>
<td>111111111...</td>
</tr>
</tbody>
</table>

RAIDR: Retention-Aware Intelligent DRAM Refresh
## 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

<table>
<thead>
<tr>
<th></th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>After 64 ms</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>After 128 ms</td>
<td>11011111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td></td>
<td>(64–128ms)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RAIDR: Retention-Aware Intelligent DRAM Refresh
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

<table>
<thead>
<tr>
<th>Time</th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>After 64 ms</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>After 128 ms</td>
<td>11011111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>(64–128ms)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After 256 ms</td>
<td>11111011...</td>
<td>11111111...</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Initially</th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>After 64 ms</td>
<td>11111111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>After 128 ms</td>
<td>11011111...</td>
<td>11111111...</td>
<td>11111111...</td>
</tr>
<tr>
<td>(64–128ms)</td>
<td>11111011...</td>
<td>11111111...</td>
<td>(128–256ms)</td>
</tr>
<tr>
<td>After 256 ms</td>
<td>1111011...</td>
<td>11111111...</td>
<td>(&gt;256ms)</td>
</tr>
</tbody>
</table>

RAIDR: Retention-Aware Intelligent DRAM Refresh
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
Grouping Rows Into Retention Time Bins

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

DRAM
② Grouping Rows Into Retention Time Bins

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

- **64-128ms**
- **128-256ms**
- **>256ms**
② Grouping Rows Into Retention Time Bins

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

Row 1

Row 2

Row 3
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:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Hash function 1  Hash function 2  Hash function 3
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
-Hash function 1- Hash function 2- Hash function 3-

Insert Row 1

RAIDR: Retention-Aware Intelligent DRAM Refresh
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Hash function 1

Hash function 2

Hash function 3

Insert Row 1
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Hash function 1

Hash function 2

Hash function 3

Insert Row 1
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Hash function 1
- Hash function 2
- Hash function 3

Insert Row 1
Example with 64–128ms bin:

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Hash function 1

Hash function 2

Hash function 3

Row 1 present?
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 1 present?
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

<table>
<thead>
<tr>
<th>Hash function 1</th>
<th>Hash function 2</th>
<th>Hash function 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 &amp; 1 &amp; 1 = 1</td>
<td>0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&amp;</td>
<td>1</td>
<td>&amp;</td>
<td>1</td>
<td>=1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Hash function 1  
Hash function 2  
Hash function 3

Row 1 present?  Yes
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

| 0 | 0 | 1 | 0 | 1 | 0 | 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:

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Hash function 1  
Hash function 2  
Hash function 3  
Row 2 present?
Storing Retention Time Bins Using Bloom Filters

Example with 64-128ms bin:

<table>
<thead>
<tr>
<th>Hash function 1</th>
<th>Hash function 2</th>
<th>Hash function 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 1 0 0 0</td>
<td>0 0 1 0 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
<td>=0</td>
</tr>
</tbody>
</table>

Row 2 present?

0 1 0

RAIDR: Retention-Aware Intelligent DRAM Refresh
Storing Retention TimeBins Using Bloom Filters

Example with 64-128ms bin:

<table>
<thead>
<tr>
<th>Hash function 1</th>
<th>Hash function 2</th>
<th>Hash function 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0</td>
<td>0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0</td>
<td>0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Row 2 present? No

RAIDR: Retention-Aware Intelligent DRAM Refresh
Storing Retention Time Bins Using Bloom Filters

Example with 64-128ms bin:

```
0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0
```

Hash function 1  Hash function 2  Hash function 3

Insert Row 4
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0
Hash function 1 Hash function 2 Hash function 3
Row 5 present?
Example with 64–128ms bin:

<table>
<thead>
<tr>
<th>Hash function 1</th>
<th>Hash function 2</th>
<th>Hash function 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 0</td>
<td>1 &amp;</td>
<td>1 &amp; 1 =1</td>
</tr>
</tbody>
</table>

Row 5 present?
Storing Retention Time Bins Using Bloom Filters

Example with 64–128ms bin:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Hash function 1

Hash function 2

Hash function 3

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
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
  - Lower temperature ⇒ longer period ⇒ less frequent refreshes
  - Higher temperature ⇒ shorter period ⇒ more 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
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 ⇒ longer period ⇒ less frequent refreshes
    - Higher temperature ⇒ shorter period ⇒ 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

- 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)
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)

RAIDR: Retention-Aware Intelligent DRAM Refresh
Performance

RAIDR: Retention-Aware Intelligent DRAM Refresh
Performance

RAIDR: Retention-Aware Intelligent DRAM Refresh
Performance

Weighted speedup

<table>
<thead>
<tr>
<th></th>
<th>Auto</th>
<th>Distributed</th>
<th>Smart</th>
<th>RAIDR</th>
<th>No Refresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>6.1%</td>
<td>8.4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td></td>
<td></td>
<td>9.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75%</td>
<td>9.6%</td>
<td>9.6%</td>
<td>9.8%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.6%</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory-intensive benchmarks in workload

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Energy Efficiency

![Energy per access (nJ)](chart)

- Auto
- Distributed
- Smart
- RAIDR
- No Refresh

Energy-intensive benchmarks in workload:

- 0%
- 25%
- 50%
- 75%
- 100%

Avg

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Energy Efficiency

![Energy per access (nJ) vs. Memory-intensive benchmarks in workload](image)

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Energy Efficiency

- Auto
- Distributed
- Smart
- RAIDR
- No Refresh

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Device Capacity Scaling: Performance

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Device Capacity Scaling: Performance

The graph illustrates the weighted speedup for different device capacities: 4 Gb, 8 Gb, 16 Gb, 32 Gb, and 64 Gb. The performance improvements for RAIDR (Retention-Aware Intelligent DRAM Refresh) are highlighted, with a significant 108% speedup at 64 Gb compared to Auto mode.
DRAM Device Capacity Scaling: Energy

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Device Capacity Scaling: Energy

Energy per access (nJ)

- **Auto**
- **RAIDR**

Device capacity:
- 4 Gb
- 8 Gb
- 16 Gb
- 32 Gb
- 64 Gb

RAIDR: Retention-Aware Intelligent DRAM Refresh
Outline

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

RAIDR: Retention-Aware Intelligent DRAM Refresh
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

RAIDR: Retention-Aware Intelligent DRAM Refresh
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 Array Organization

Word lines

Bit lines

Cell

Sense amp

Sense amp

Sense amp

Row

Row buffer

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Activation

\( V_{DD}/2 \)

Ssense amp

\( 0V \)

\( + \)

\( V_{DD} \)

\( - \)
DRAM Activation

$V_{PP}$

$V_{DD}/2$

Sence amp

$V_{DD}$

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Activation

\[ V_{PP} \]

\[ V_{DD}/2 + \delta \]

Sense amp

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Activation

![Diagram of DRAM Activation]

- $V_{PP}$
- $V_{DD}$

Sense amp
DRAM Precharge

RAIDR: Retention-Aware Intelligent DRAM Refresh
DRAM Precharge

$V_{DD}$

Sense amp

0V

$V_{DD}$

$V_{DD}$
DRAM Precharge

$V_{DD}/2$

Sense amp

0V

$V_{DD}$
RAIDR Components

Period Counter → Row Counter

64-128ms Bloom Filter
128-256ms Bloom Filter

Refresh Rate Scaler Counter → Refresh Rate Scaler Period
Idle Power Consumption

RAIDR: Retention-Aware Intelligent DRAM Refresh
Performance: 85°C

Memory-intensive benchmarks in workload

RAIDR: Retention-Aware Intelligent DRAM Refresh
Energy: 85°C

Memory-intensive benchmarks in workload

Energy per access (nJ)

<table>
<thead>
<tr>
<th>Auto</th>
<th>Distributed</th>
<th>Smart</th>
<th>RAIDR</th>
<th>No Refresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1%</td>
<td>9.0%</td>
<td>7.9%</td>
<td>6.9%</td>
<td>6.4%</td>
</tr>
<tr>
<td>8.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RAIDR: Retention-Aware Intelligent DRAM Refresh
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: Retention-Aware Intelligent DRAM Refresh

38
### RAIDR Configurations

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