THE UNIVERSITY OF MAINE

MSST 2010

# Energy and Thermal Aware Buffer Cache Replacement Algorithm

**Jianhui Yue**, Yifeng Zhu,
Zhao Cai, and Lin Lin

Electrical and Computer Engineering
University of Maine

# Research Summary

- Memory power consumption
  - Consumes 50% more power than CPU on server
  - Consumes  2 times more power than disks in a specific supercomputing center
  - Power dissipation increases cooling cost
- Power and thermal aware buffer cache replacement  algorithm
  - Limit replacements on a smaller set of memory ranks without hurting hit rate
  - Save up to 27% energy
  - Reduce temperature 5.45 ℃ than LRU

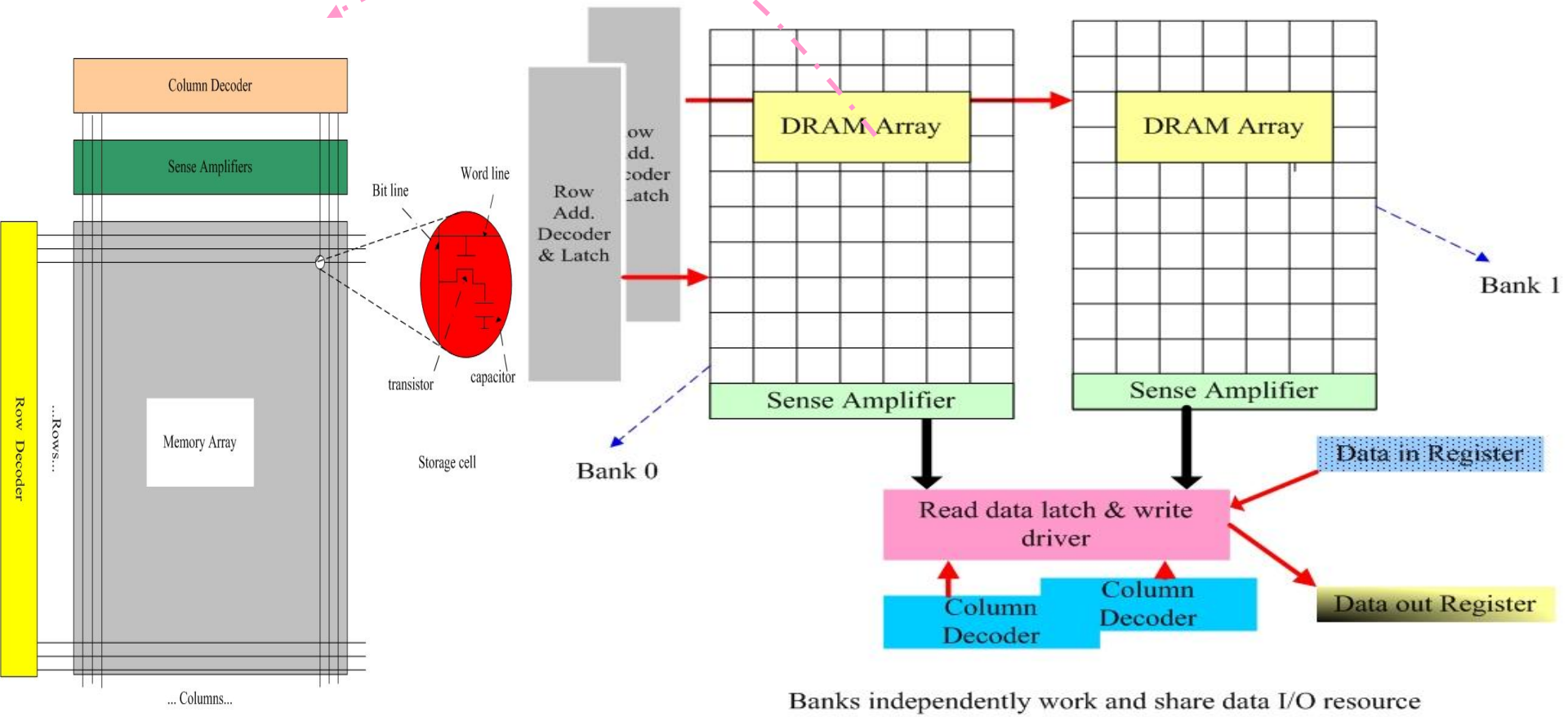
# Outline

- ***Background of memory systems***
- New buffer cache replacement algorithm
- Evaluation
- Conclusion and future work

# Memory Chip Organization

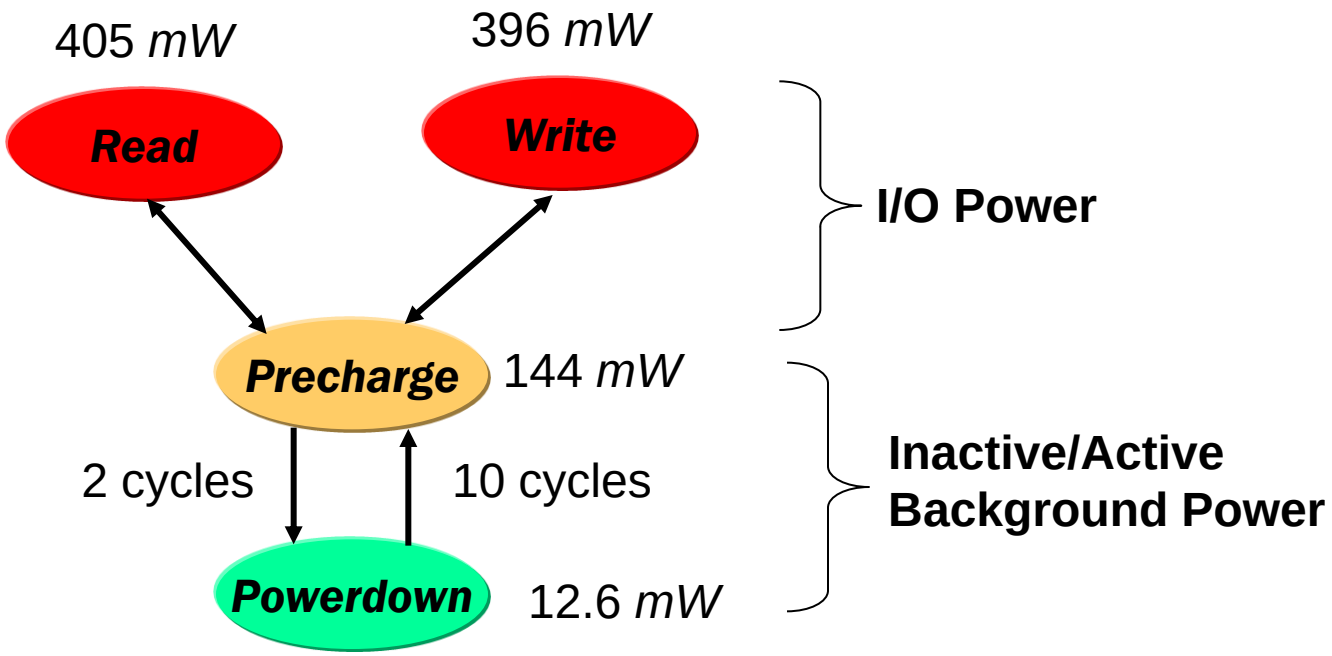
DIMM includes one/two ranks
which is an independent pwr. mgt. unit

DRAM Device Diagram



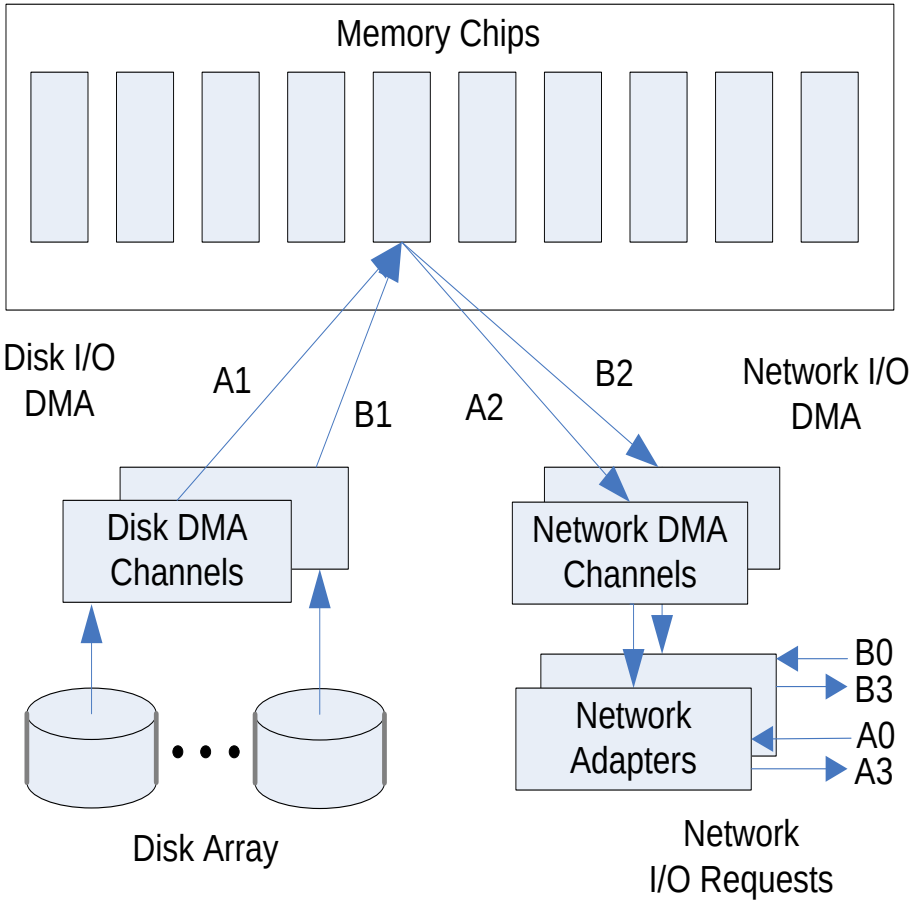

Banks independently work and share data I/O resource

# DRAM Power States




- Each rank includes multiple DDRx devices and provides data to data bus in union
- The rank is basic memory power management unit

# DMA Overlapping

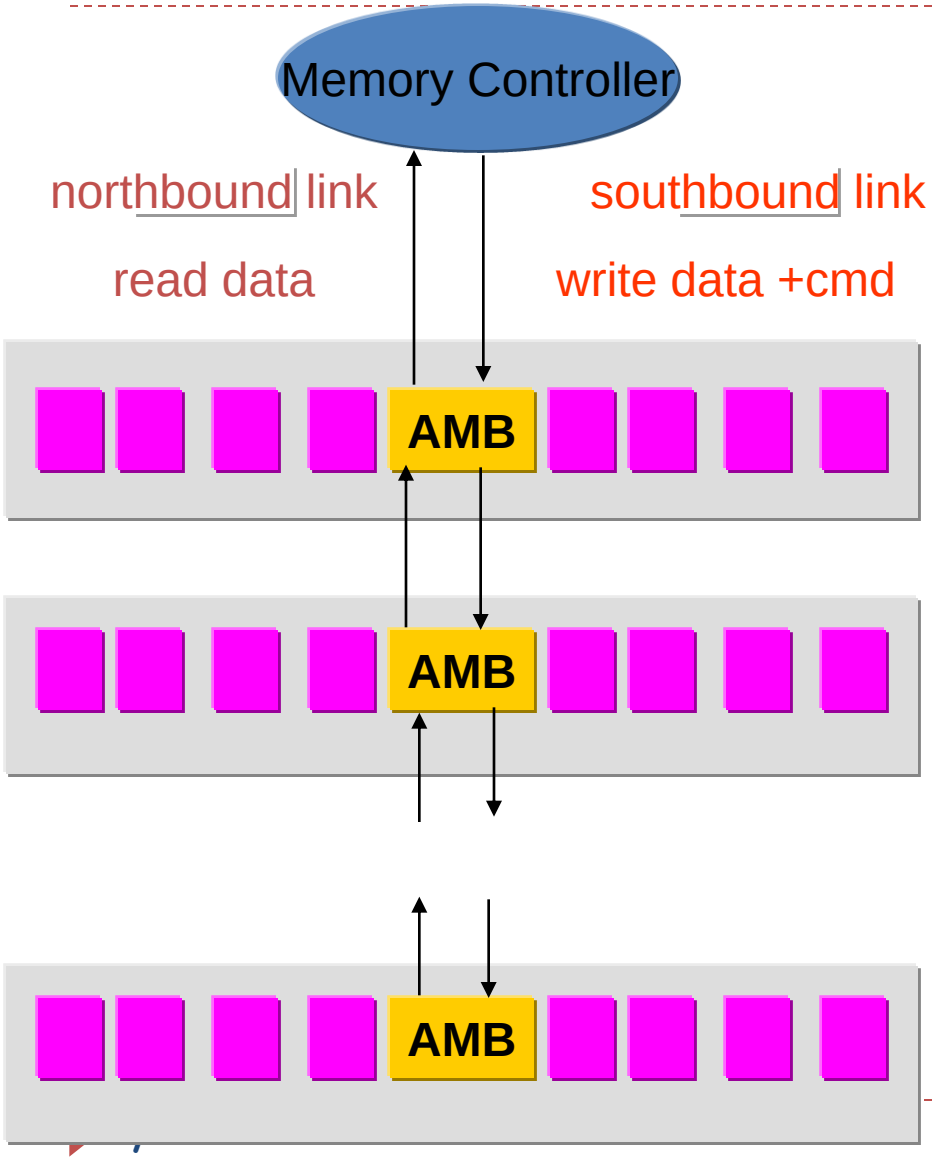



Multiple DMA transfers on different PIC buses to access the same memory rank simultaneously in a time multiplex way.

Multiplexing multiple slow DMA transmissions to one memory rank can

reduce active background power

# Fully-Buffered DIMM(FBDIMM)




1. **Increases memory density, reliability and speed Includes read data link and write data link**
2. **AMB acts as pass-through switch**
3. **AMB consumes power**

n is up to 7

# Outline

- *Background of memory systems*
- **New buffer cache replacement algorithm**
- Evaluation
- Conclusion and future work

# Bottom Most Replacement Algorithm (BMA)

- Main Idea
  - Limit replacements to a smaller number of memory ranks without hurting hit rate much.
- Method
  - Choose the victim rank with the **most** number of cold blocks in the **bottom** of LRU stack
  - Find victim block from the specified victim rank rather than LRU block for energy saving
  - Update victim rank for adaption to blocks' recency changes

# Choose Victim Rank

## LRU

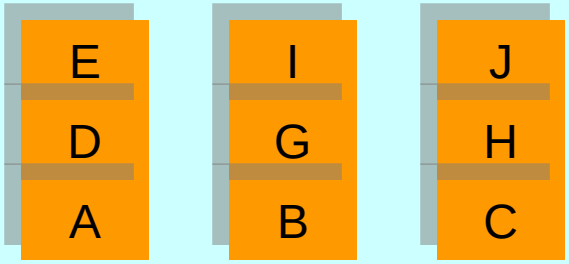
LRU stack

top

| J |
|---|
| I |
| H |
| G |
| E |
| D |
| C |
| B |
| A |

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

## Bottom Most Algorithm (BMA)

LRU stack

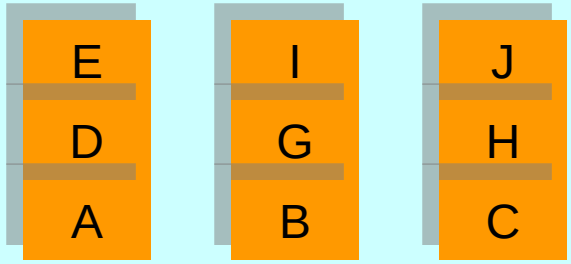
top

| J |
|---|
| I |
| H |
| G |
| E |
| D |
| C |
| B |
| A |

Window = 5

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

*last_victim_cnt = 3*

*victim_rank = 0*

# Access Sequence: *W, X, Y, Z*

## LRU

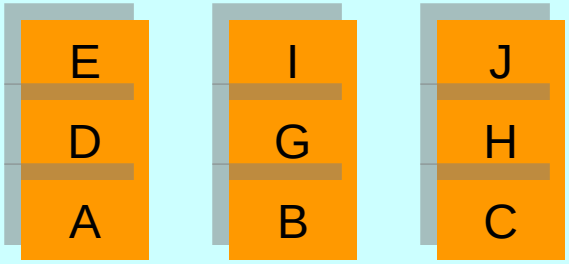
**LRU stack**

*top*

| J |
|---|
| I |
| H |
| G |
| E |
| D |
| C |
| B |
| A |

*bottom*

**Memory Physical Layout**




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| | | |
| | | |
| | | |

## Bottom Most Algorithm (BMA)

**LRU stack**

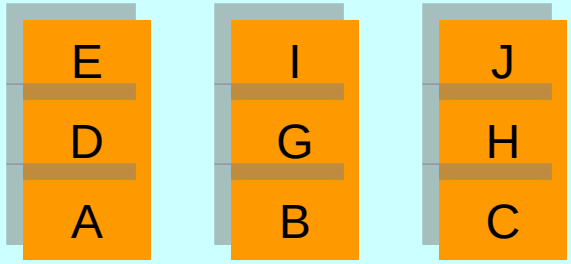
*top*

| J |
|---|
| I |
| H |
| G |
| E |
| D |
| C |
| B |
| A |

Window = 5

*bottom*

**Memory Physical Layout**




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| | | |
| | | |
| | | |

*last_victim_cnt = 3*

*victim_rank = 0*

# Access Sequence: W, X, Y, Z

## LRU

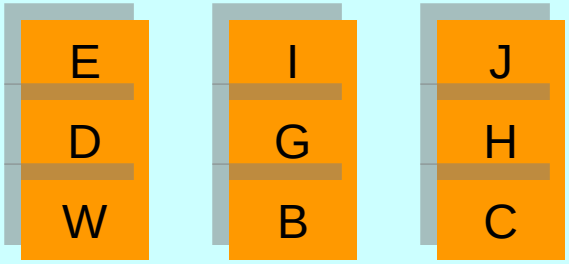
LRU stack

top

| W |
|---|
| J |
| I |
| H |
| G |
| E |
| D |
| C |
| B |

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | B | 1 |
| | | |
| | | |

## Bottom Most Algorithm (BMA)

LRU stack

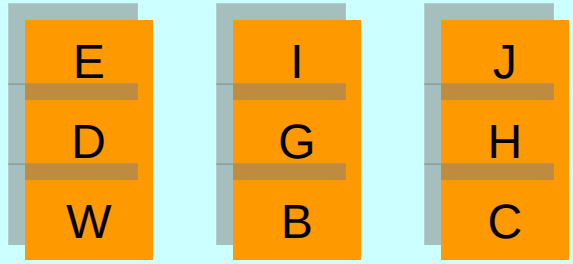
top

| W |
|---|
| J |
| I |
| H |
| G |
| E |
| D |
| C |
| B |

Window = 5

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | D | 0 |
| | | |
| | | |

*last_victim_cnt = 2*

*victim_rank = 0*

# Access Sequence: *W, X, Y, Z*

## LRU

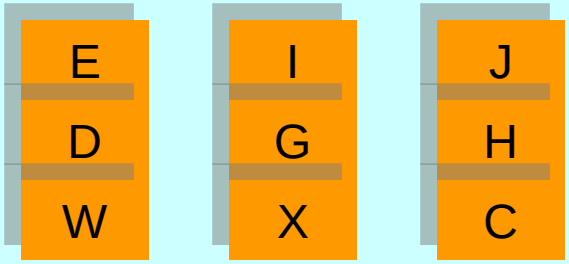
**LRU stack**

*top*

| X |
|---|
| W |
| J |
| I |
| H |
| G |
| E |
| D |
| C |

*bottom*

**Memory Physical Layout**




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | B | I |
| Y | C | 2 |
| | | |

## Bottom Most Algorithm (BMA)

**LRU stack**

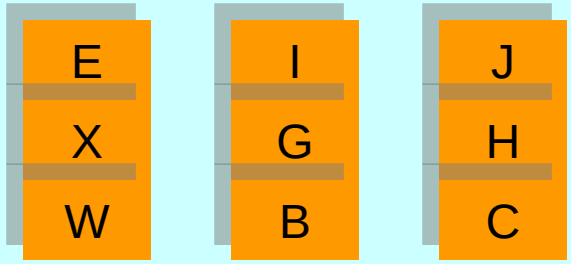
*top*

| X |
|---|
| W |
| J |
| I |
| H |
| G |
| E |
| C |
| B |

Window = 5

*bottom*

**Memory Physical Layout**




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | D | 0 |
| Y | E | 0 |
| | | |

*last_victim_cnt = 1*

*victim_rank = 0*

# After Y swapped in

## LRU

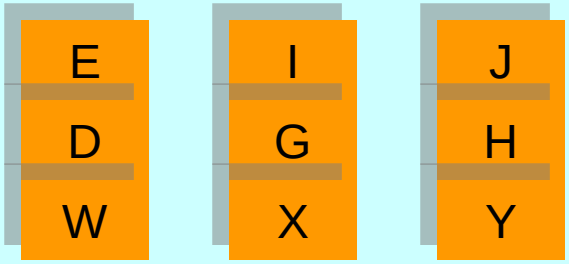
LRU stack

top

| Y |
|---|
| X |
| W |
| J |
| I |
| H |
| G |
| E |
| D |

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | B | 1 |
| Y | C | 2 |
| | | |

## Bottom Most Algorithm (BMA)

LRU stack

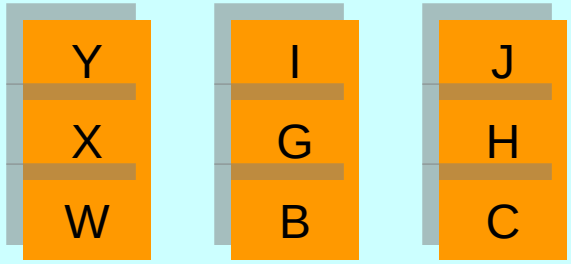
top

| Y |
|---|
| X |
| W |
| J |
| I |
| H |
| G |
| C |
| B |

Window = 5

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | D | 0 |
| Y | E | 0 |
| | | |

*last_victim_cnt = 0*

*victim_rank = 0*

# Choose New Victim Rank

## LRU

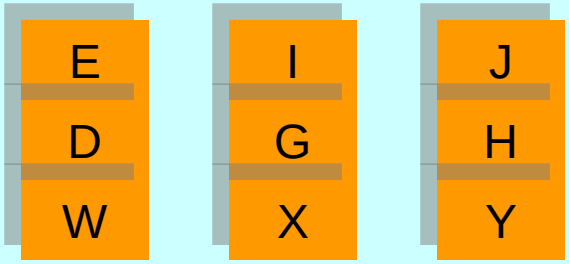
**LRU stack**

*top*

| Y |
|---|
| X |
| W |
| J |
| I |
| H |
| G |
| E |
| D |

*bottom*

**Memory Physical Layout**




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | B | 1 |
| Y | C | 2 |
| | | |

## Bottom Most Algorithm (BMA)

**LRU stack**

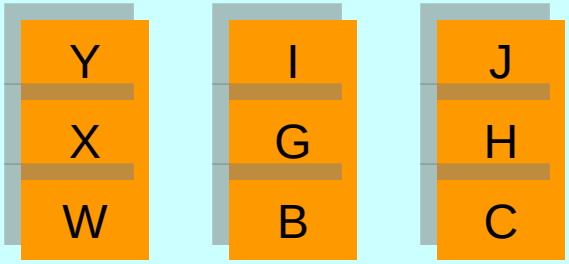
*top*

| Y |
|---|
| X |
| W |
| J |
| I |
| H |
| G |
| C |
| B |

Window = 5

*bottom*

**Memory Physical Layout**




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | D | 0 |
| Y | E | 0 |
| | | |

*last_victim_cnt = 3*

*victim_rank = 1*

# Access Sequence: W, X, Y, Z

## LRU

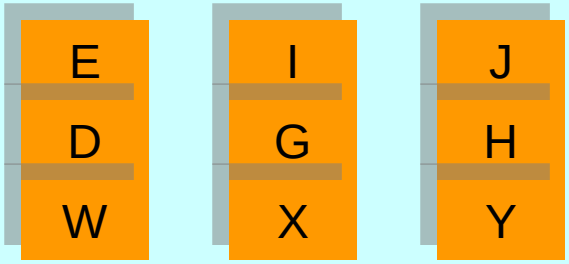
LRU stack

top

| Y |
|---|
| X |
| W |
| J |
| I |
| H |
| G |
| E |
| D |

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | B | 1 |
| Y | C | 2 |
| Z | D | 0 |

## Bottom Most Algorithm (BMA)

LRU stack

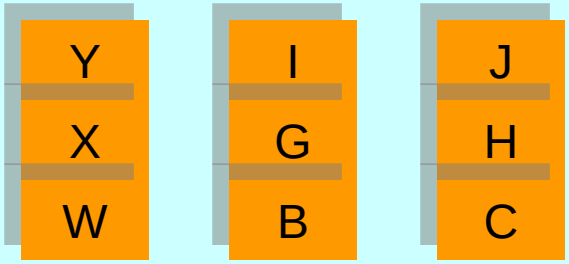
top

| Y |
|---|
| X |
| W |
| J |
| I |
| H |
| G |
| C |
| B |

Window = 5

bottom

Memory Physical Layout




| Request block | Victim block | Victim rank |
|---|---|---|
| W | A | 0 |
| X | D | 0 |
| Y | E | 0 |
| Z | B | 1 |

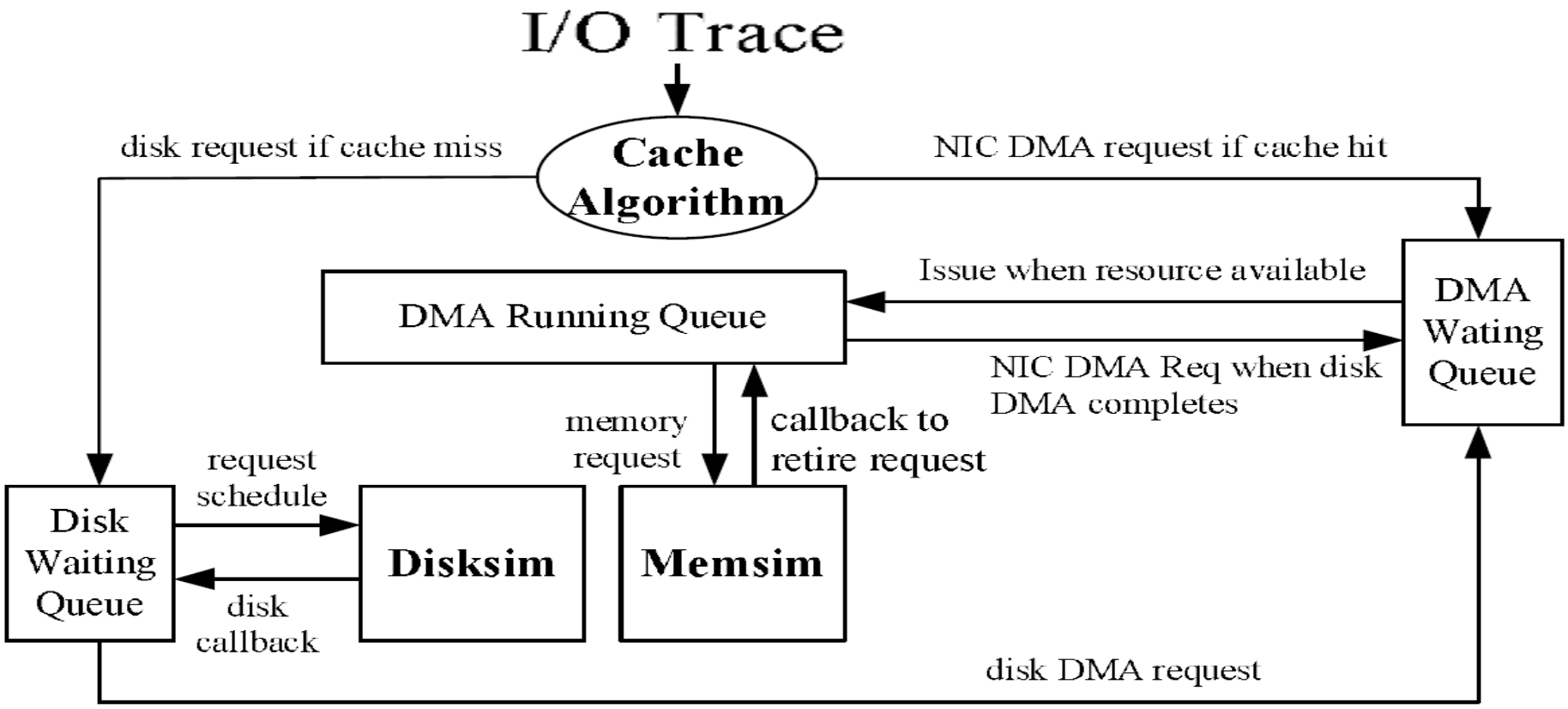
*last_victim_cnt = 3*

*victim_rank = 1*

# Outline

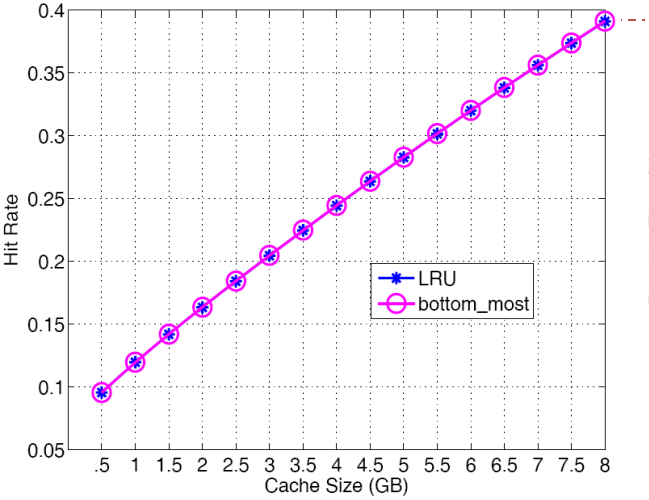
- Background
- New buffer cache replacement algorithm
- ***Evaluation***
- Conclusion and future work

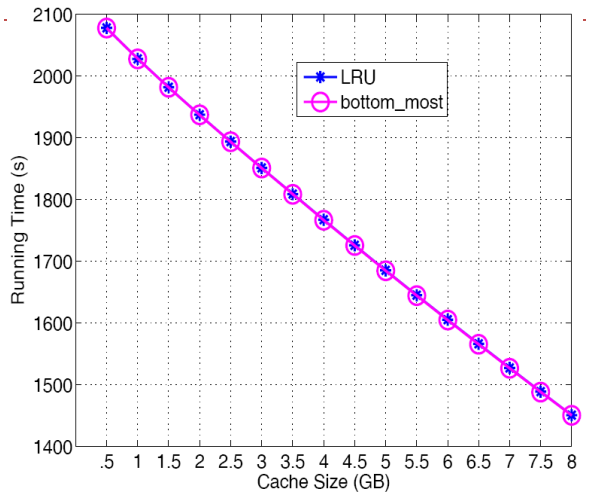
# Simulator and I/O Trace Characteristics

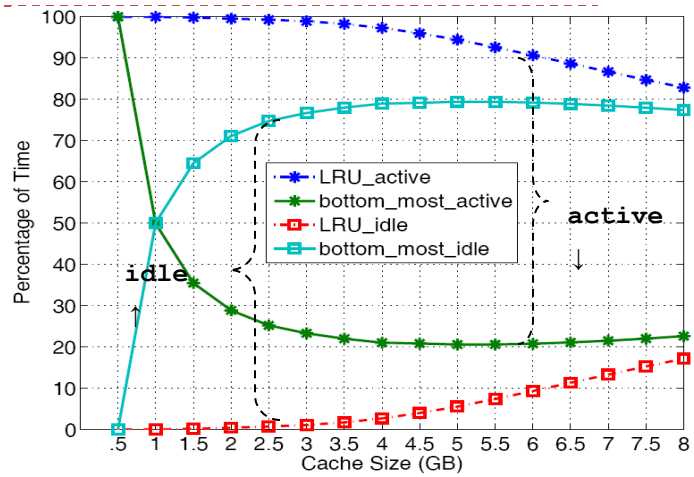


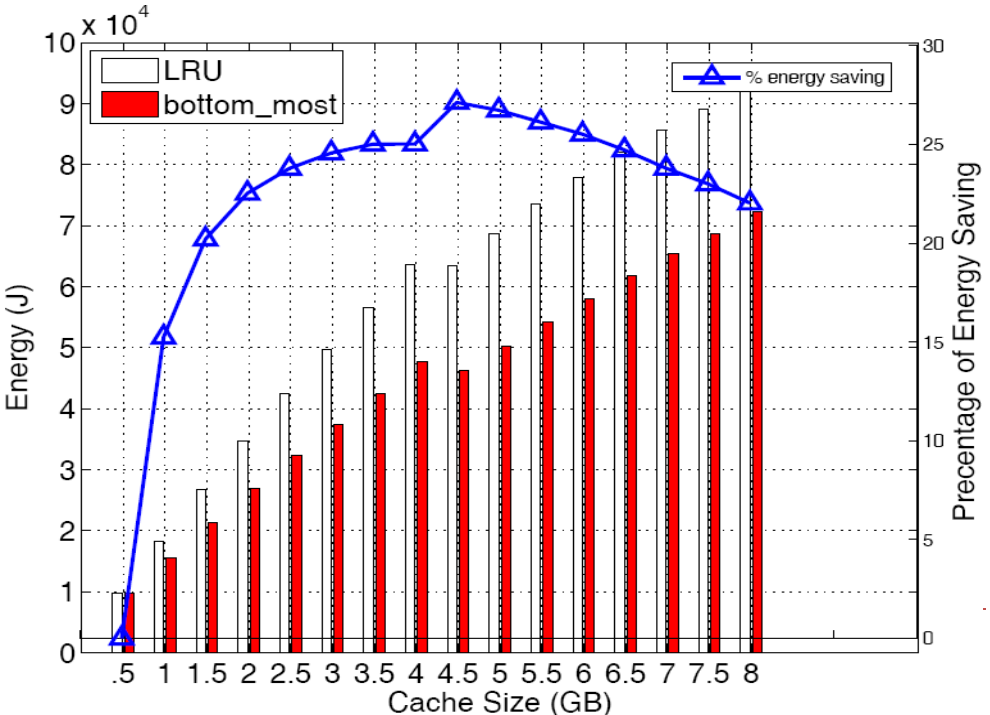

| Trace | Working Set(GB) | Traffic(GB) | Duration(Sec) |
|---|---|---|---|
| TPC-C | 66.53 | 465.3 | 360 |
| LM-TBE | 143.9 | 205.5 | 3604 |
| MSN-BEFS | 10.58 | 18.05 | 603 |

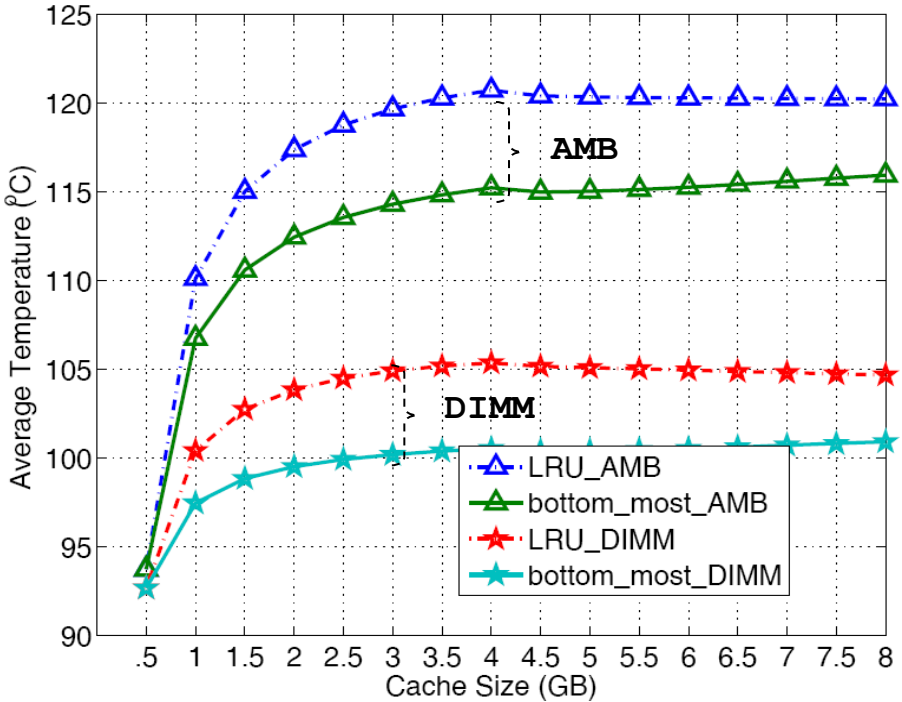
# TPC-C Results



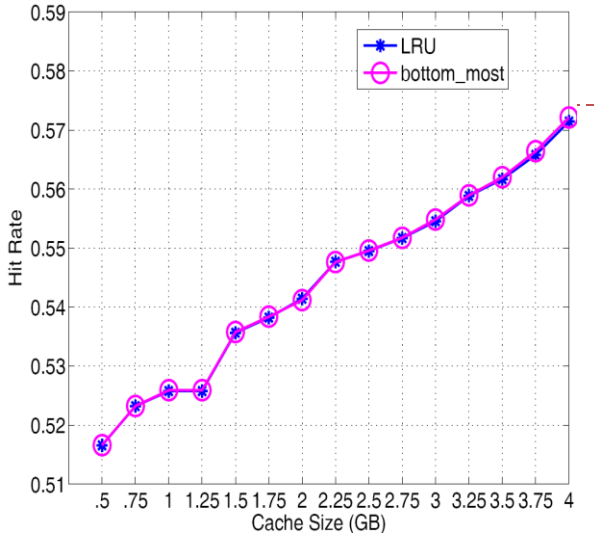



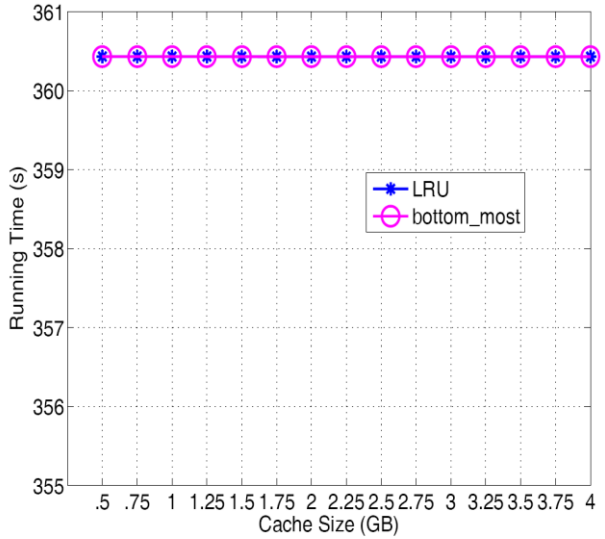



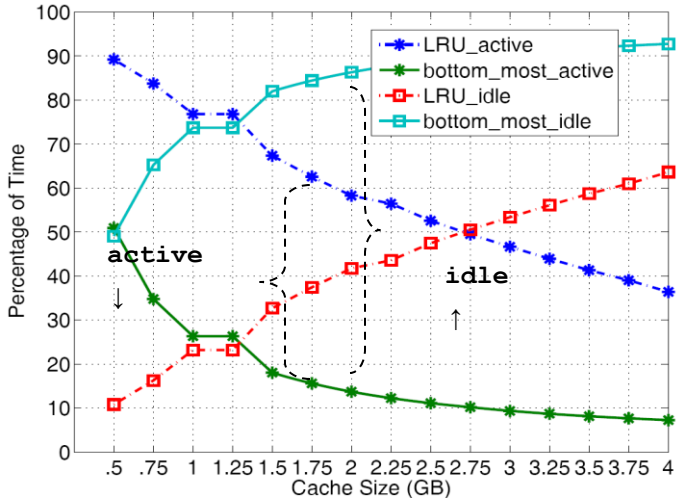



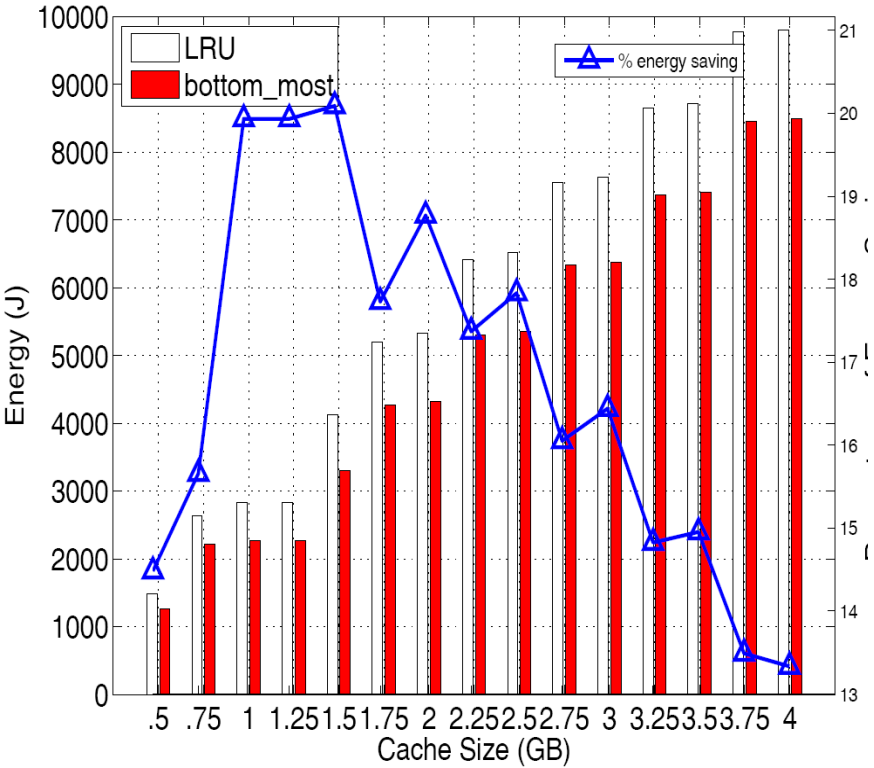



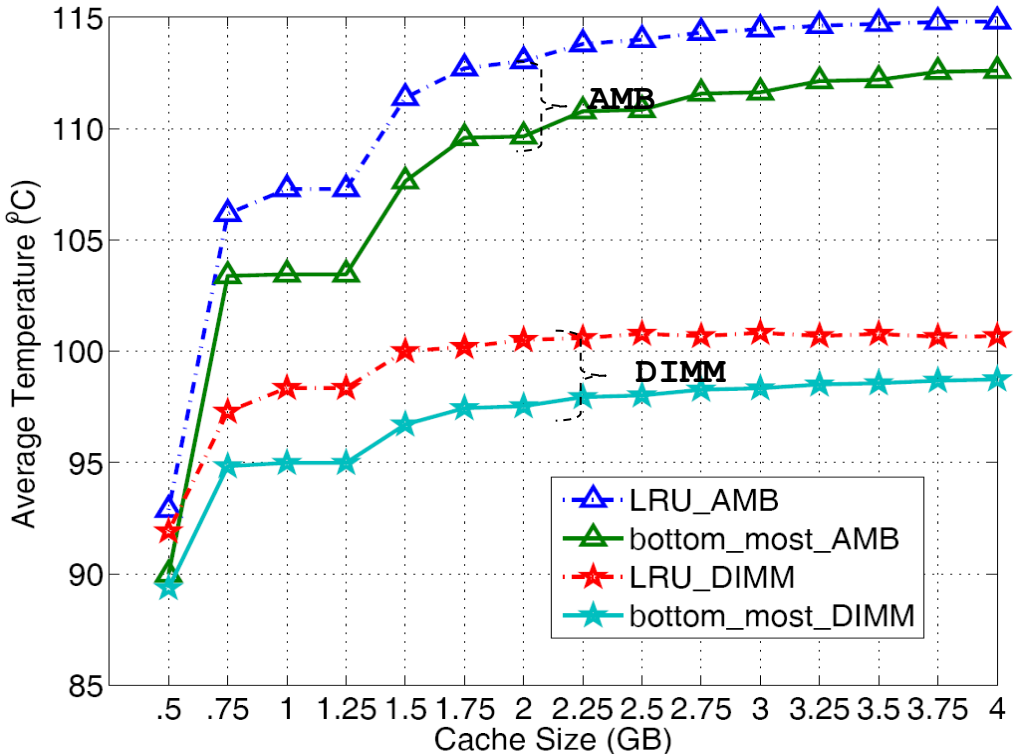
# LM-TBE Results



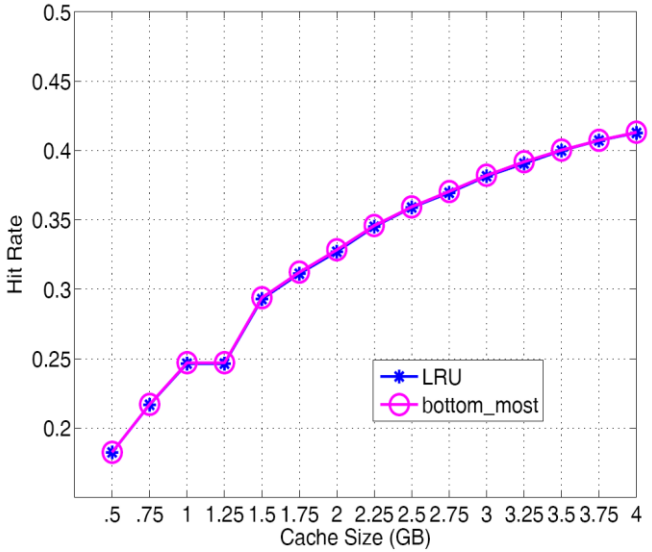



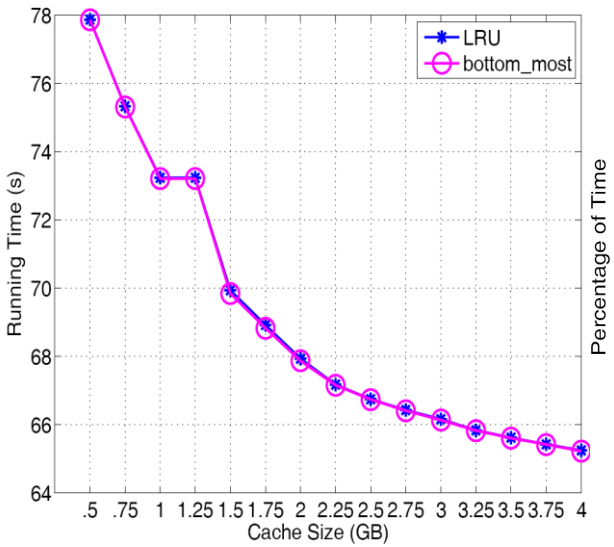



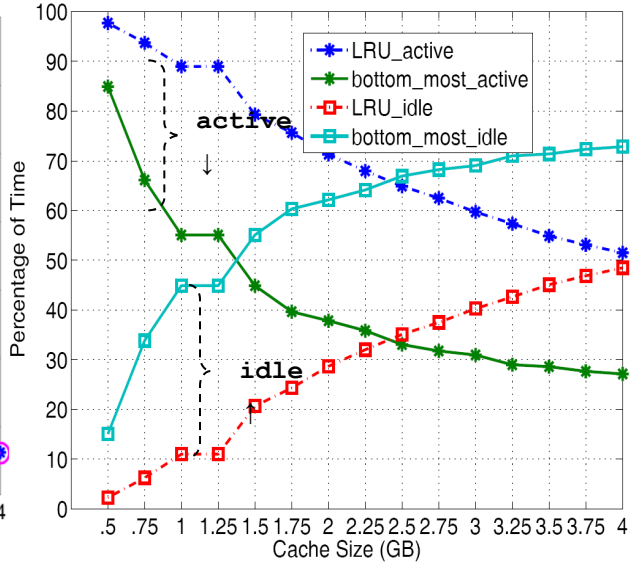



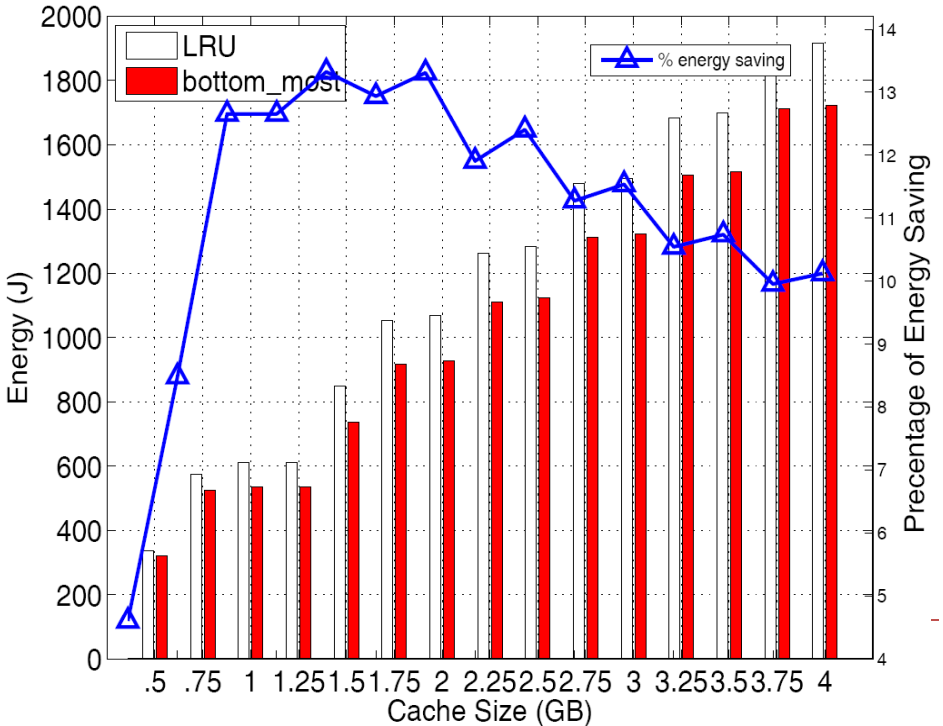



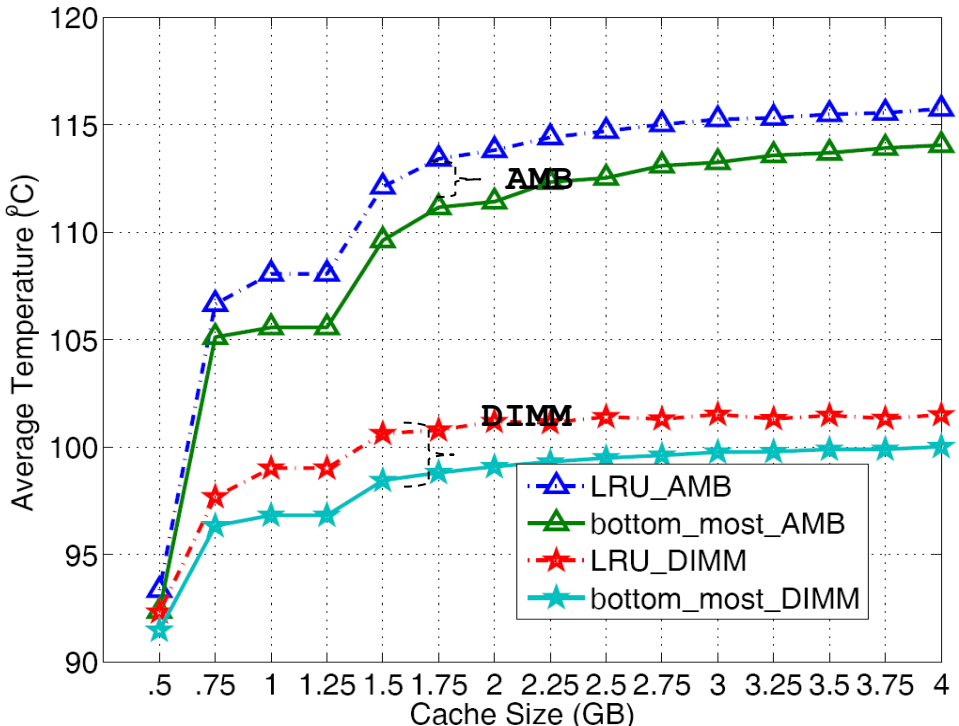
# MSN-BEFS Results



## Outline

- Background
- New buffer cache replacement algorithm
- *Evaluation*
- Conclusion and future work

# Conclusion and Future Work

- The first method to save buffer cache memory energy without data migration
- Limit replacement to smaller set of memory rank without sacrificing hit rate for memory
- Saves up to 27% energy  and reduces temperature 5.45 ℃ than LRU
- Implement this algorithm at Linux Kernel
- Evaluate algorithm at real physical machine with I/O intensive workload

# Thanks !

# DRAM Thermal

$$P_{AMB} = P_{static} + P_{localTraffc} + P_{forwardingTraffic}$$

$$\begin{aligned}
T_{AMB} &= T_A + P_{AMB} * \psi_{AMB} \\
&\quad + P_{DRAM} * \psi_{DRAM_AMB} \\
T_{DRAM} &= T_A + P_{AMB} * \psi_{AMB_DRAM} \\
&\quad + P_{DRAM} * \psi_{DRAM} \\
T(t + \Delta t) &= T(t) + (T_{stable} - T(t)) * (1 - e^{-\frac{\Delta t}{\tau}})
\end{aligned}$$

# Simulation Parameters

| Parameter | value |
|---|---|
| rank powerdown delay | 3 memory cycles |
| rank powerup delay | 10 memory cycles |
| rank powerdown threshold | 30 memory cycles |
| tRP: Row Precharge time | 15ns |
| tRCD: Row active to row active delay | 15ns |
| tRAS : Row Activation time | 45ns |
| tCAS: Delay to access a certain column | 15ns |
| IDD0: Active precharge current | 80mA |
| IDD2P: Precharge powerdown current | 7mA |
| IDD2N: Precharge standby current | 45mA |
| IDD3N: Active standby current | 55mA |
| IDD3P: Active powerdown current | 240mA |
| IDD4R: Burst read current | 145mA |
| IDD4W:Burst write current | 140mA |
| IDD5: Burst refesh current | 170mA |
| IDD6: Self refresh current | 7mA |
| AMB_Power1: the last AMB static power | 2W |
| AMB_Power: other AMB static power | 2.55W |
| AMB dynamic power for local traffic | 3.55W |
| AMB dynamic power for forwarding traffic | 2.8W |
| AMB read bandwidth | 6.4GB/s |
| AMB write bandwidth | 3.2GB/s |
| # of ranks per DIMM | 2 |
| # of DIMMS per channel | 8 |
| Rank capacity | 256MB |
| Vdd: Supply voltage | 1.8v |



Search window 1024 Thresh window 1024 Block size 64KB