Hiding DRAM Refresh Overhead in Real-Time Cyclic Executives

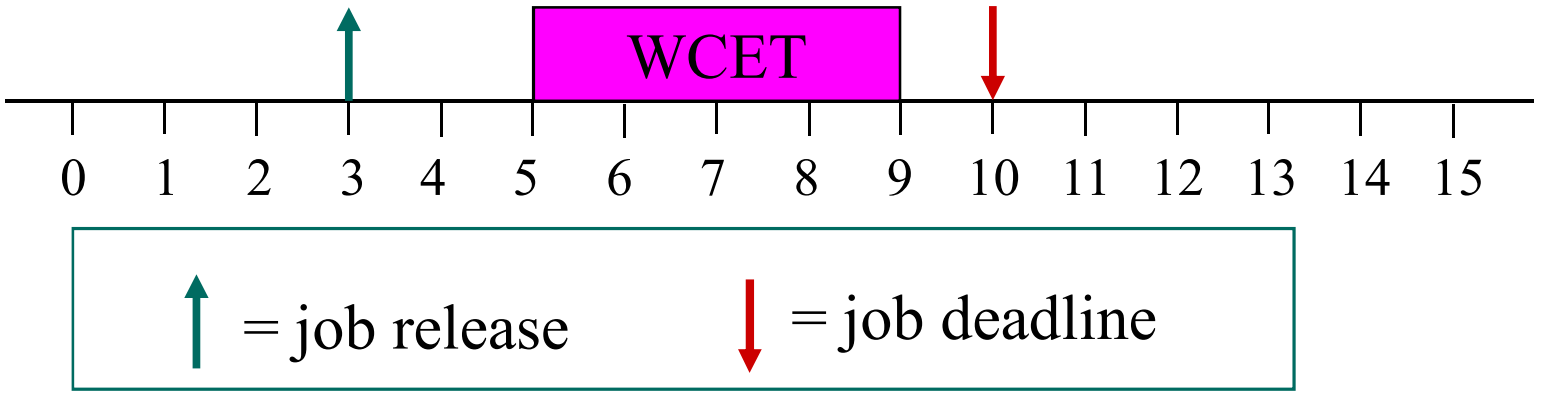
Xing Pan, Frank Mueller

North Carolina State University

NC STATE UNIVERSITY
Department of Computer Science

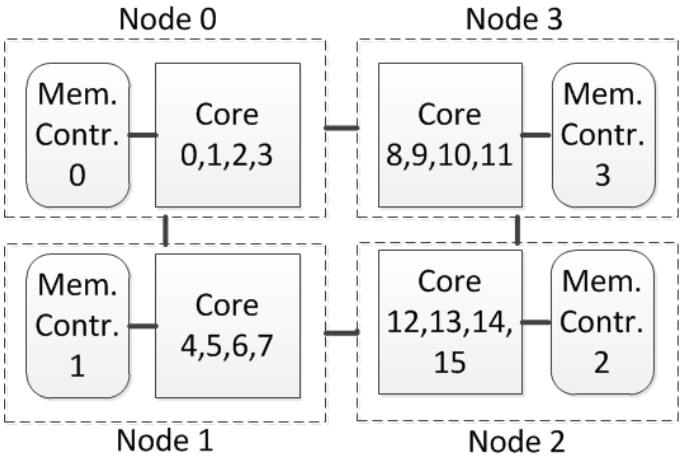
Real-time system

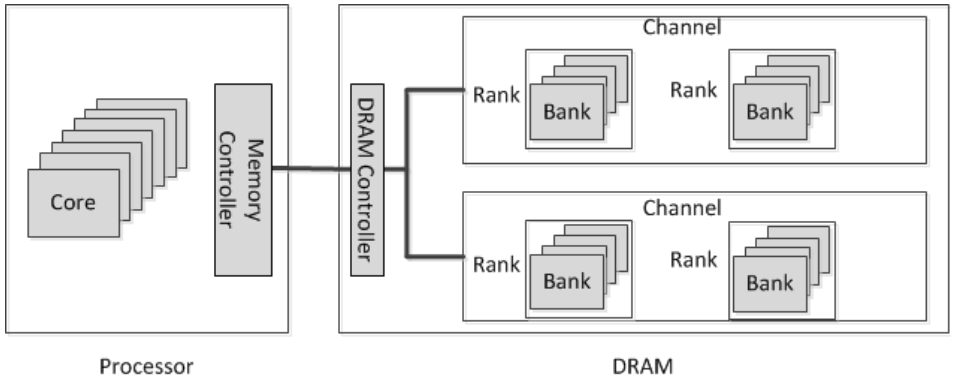
- Real-Time System requires:
  - Logical Correctness: Produces correct outputs.
  - Temporal Correctness: Produces outputs at the right time.
- Real-time task
  - predict its worst-case execution time
  - schedule it to meet its deadline



NUMA Architecture

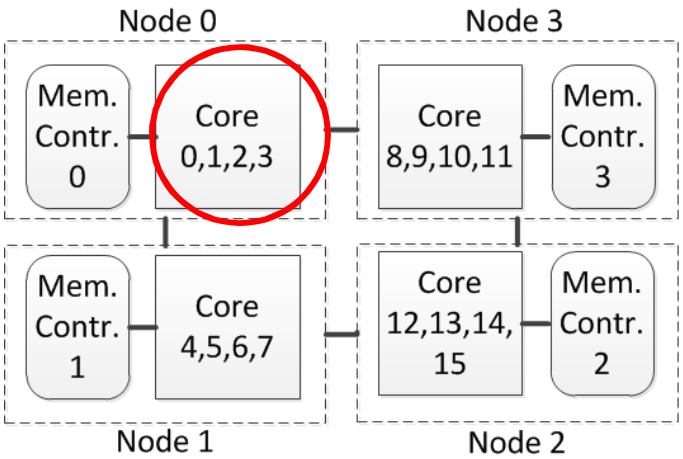
- Modern NUMA (non-uniform memory access) architectures:
  - CPU partitions sets of cores into "node":
    1 local + several remote controllers
  - Each memory controller (node) consists of multilevel resources (channel, rank and bank)





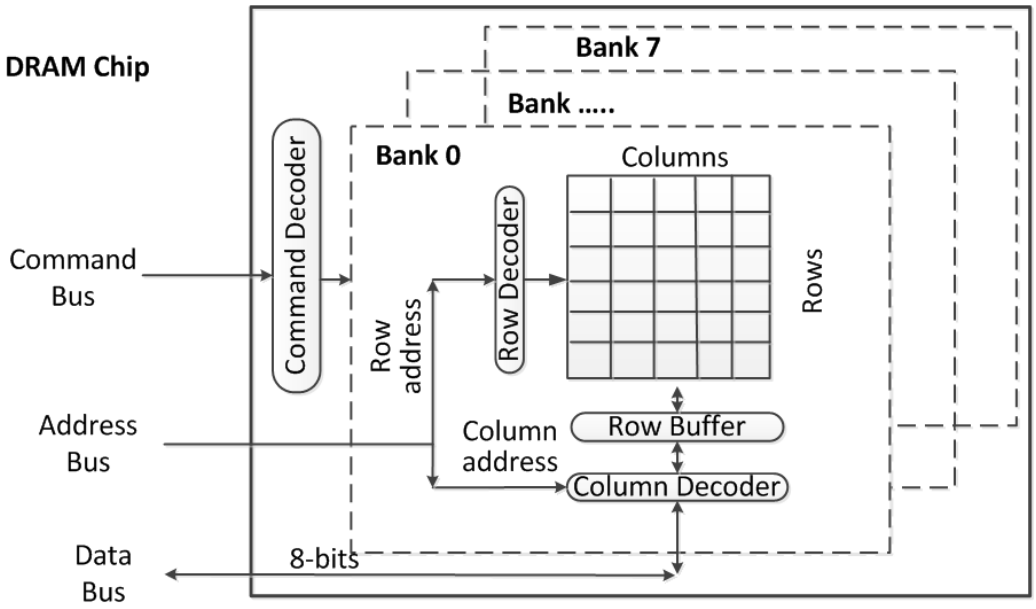
Core Isolation → Hard Real-Time Composition

- Challenge: shared resources
  - One core execution affects other cores
- Objective: Isolate cores
  - Allows compositional timing analysis
- Application: mission critical hard real-time
  - Automated driving…



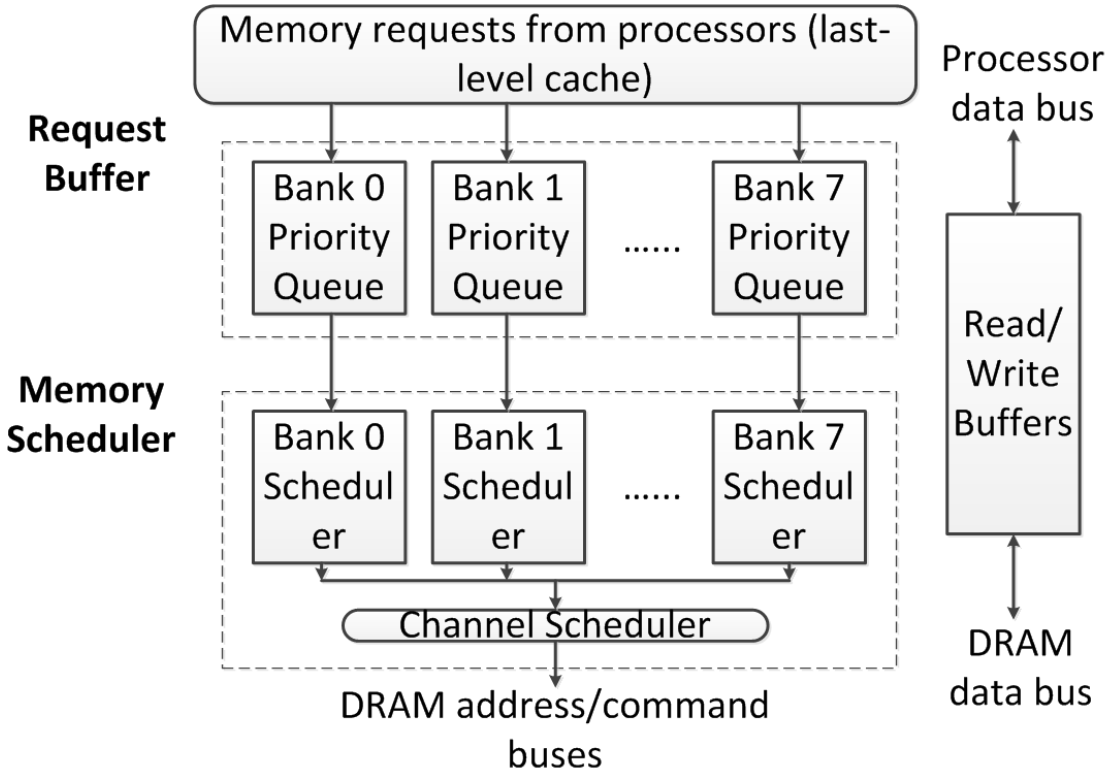
DRAM Organization

- DRAM bank array has: rows+columns of data cells
- Load the row which contains requested data into Row Buffer
  - Row Buffer hit vs. Row Buffer miss



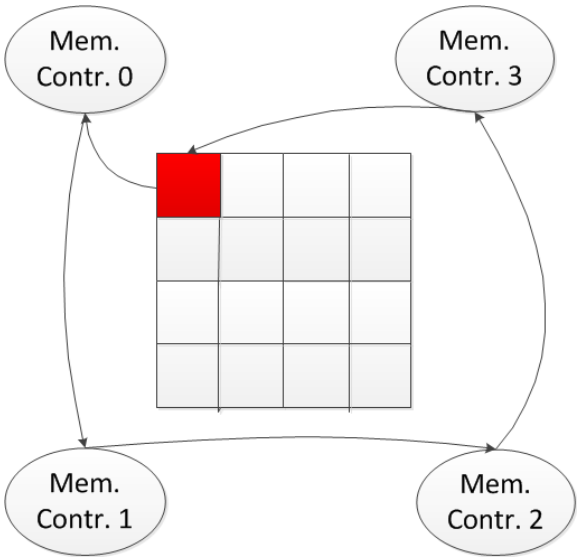
Memory Controller

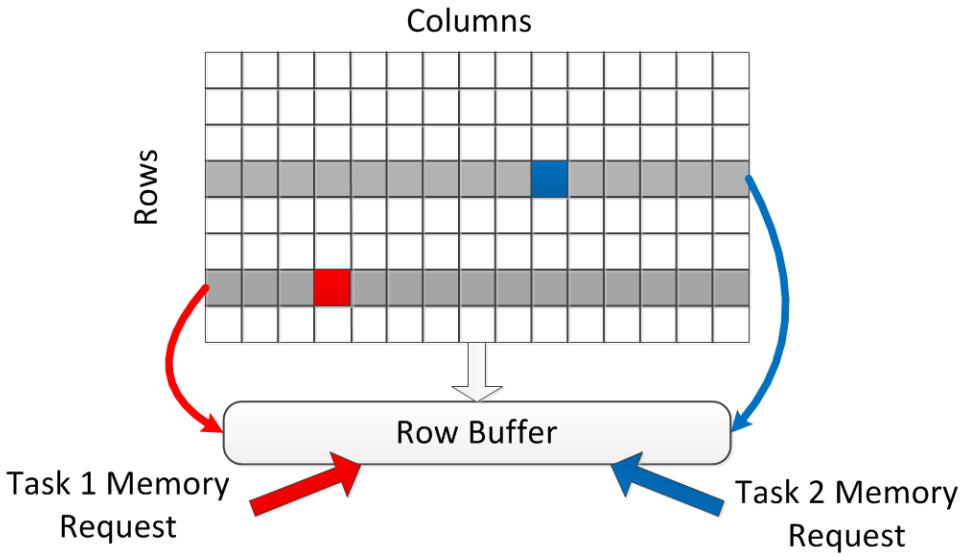
- DRAM banks can be accessed in parallel



Motivation

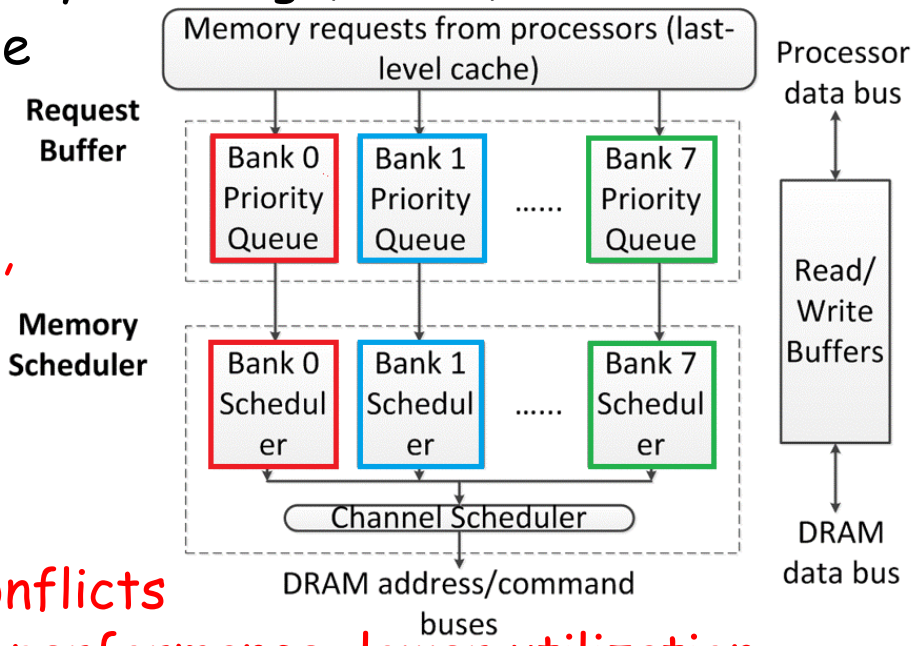
- Apps on NUMA arch. experience varying execution times due to
  - Remote memory node accesses
  - Conflict in memory banks/controllers



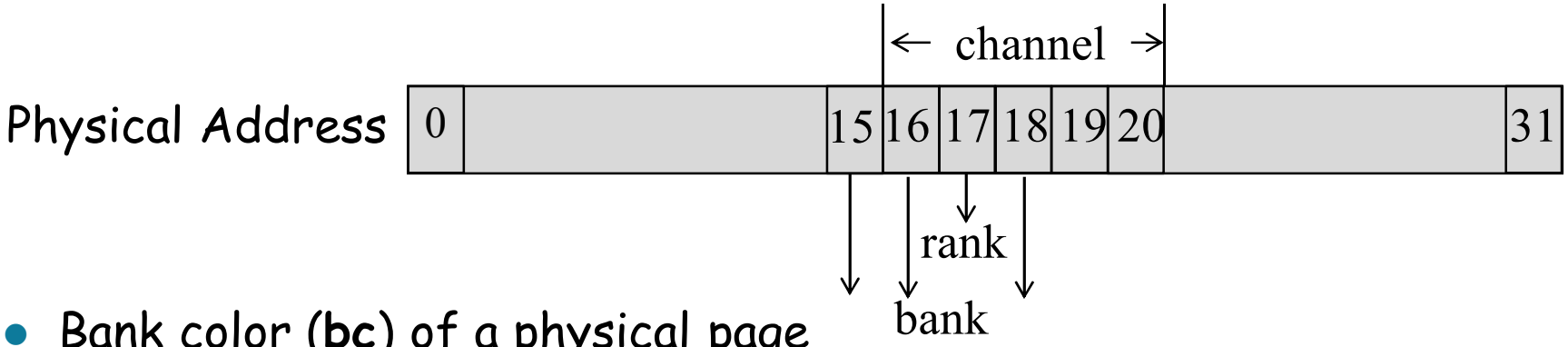


Past: Memory Predictability by Coloring

- Local node policy under standard buddy allocation / numa library
  - Not bank aware
  - numa library only works on heap memory
- Previous Work
  - Our Controller-Aware Memory Coloring (CAMC) @ SAC'18
  - NUMA causes unpredictable execution time
  - New memory allocator in kernel via mmap() syscall, no hardware modifications
  - Each task gets private memory (coloring) on local NUMA node
  - Avoid remote refs, bank conflicts → predictable exec., lower performance, lower utilization



Memory Frame Color Selection



- Bank color (**bc**) of a physical page

  bc = ((node∗NN ∗NC+channel)∗NR+rank)∗NB+bank

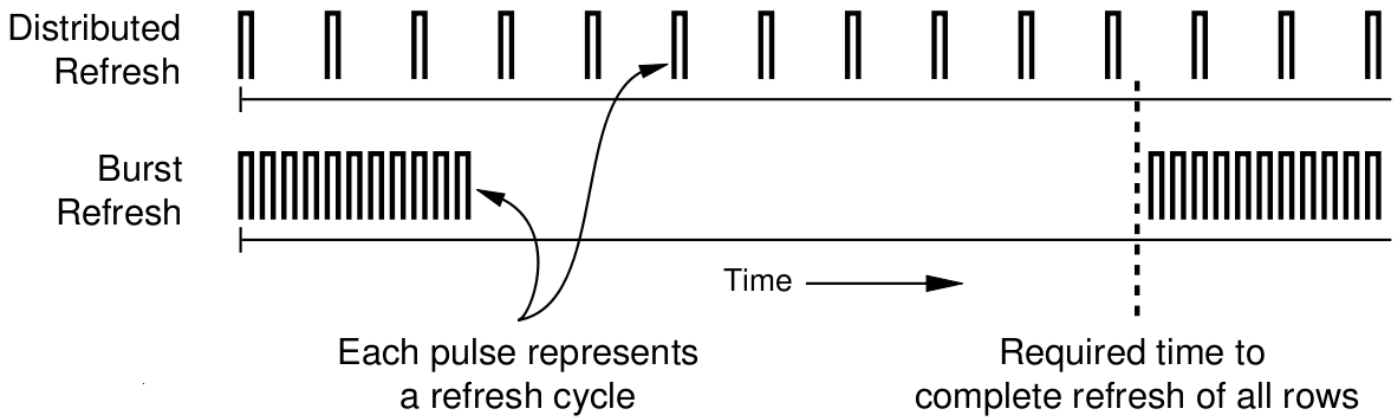
  - NN: # nodes (mem controllers) of a system
  - NC: # channels per controller
  - NR: # ranks per channel
  - NB: # banks per rank
- Opteron 6128: NN=4, NC=2, NR=2, NB=8, Total of 128 colors
- Example: page in node 0, channel 1, rank 1 and bank 2
  → color is ((0∗4∗2+1)∗2+1)*8+2=26

Focus in this Paper: DRAM Refresh

- Dynamic Random Access Memory (DRAM)
  - data is stored in the capacitor as 1 or 0 (electrically charged/discharged)
  - capacitors slowly leak their charge over time
  - requires cells to be refreshed, otherwise data would be lost.

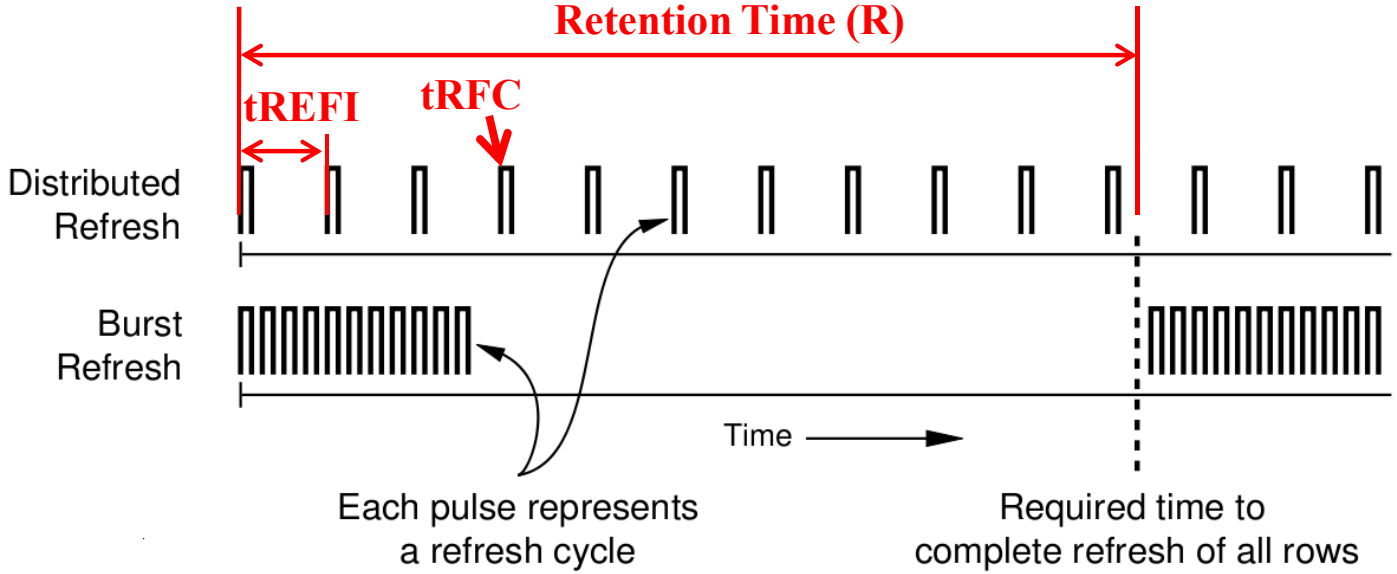
Unpredictability due to DRAM Refresh

- Refresh commands to all DRAM cells periodically issued by DRAM controller to maintain data validity.
  - row-buffer is closed
  - any memory access deferred until refresh completes
- Distributed Refresh vs. Burst refresh



Unpredictability due to DRAM Refresh

- Refresh commands to all DRAM cells periodically issued by DRAM controller to maintain data validity
  - row-buffer is closed
  - any memory access deferred until refresh completes
- Distributed Refresh vs. Burst refresh



DRAM Refresh Trends: It's getting worse

- R: 64 ms / 32 ms. determined by temperature (85 C)
- tRFC increases quickly with growing DRAM densities

Chip Density	# banks	#rows/bank	#rows/bin	tRFC
1Gb	8	16K	16	110 ns [1]
2Gb	8	32K	32	160 ns [1]
4Gb	8	64K	64	260 ns [1]
8Gb	8	128K	128	350 ns [1]
16Gb	8	256K	256	550 ns [2]
32Gb	8	512K	512	> 1 us [3]
64Gb	8	1M	1K	> 2 us [3]

- [1] Standard, JEDEC, DDR3 SDRAM
- [2] Standard, JEDEC, DDR4 SDRAM
- [3] Jamie Liu, Onur Mutlu et al. "RAIDR: Retention-aware intelligent DRAM refresh." *ACM SIGARCH Computer Architecture News*. 2012.

Challenge: Refresh Delay

- Auto-refresh : recharges all the memory cells within the "retention time"
  - a rank during refresh becomes unavailable to memory requests until the refresh completes (tRFC).
  - all bank row buffers of this rank closed (tRP) and need to be re-opened (tRAS)
  - More bank row buffer misses around refreshes.

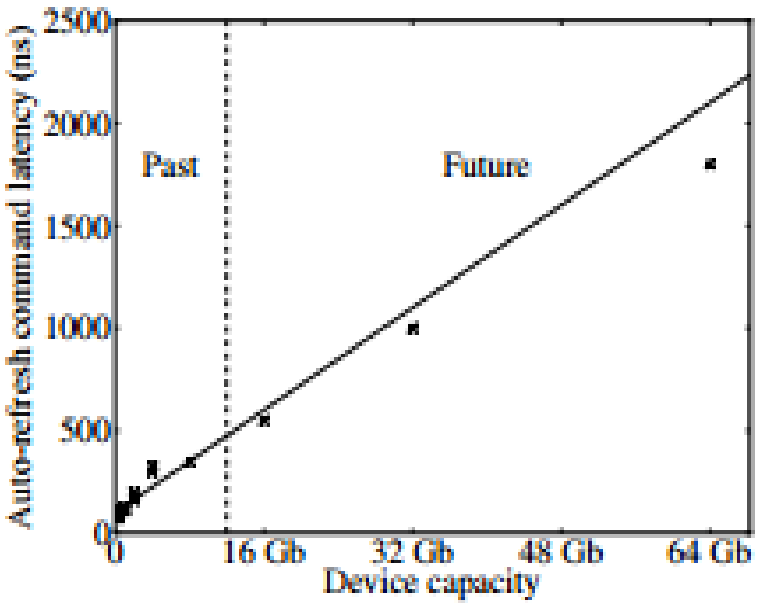
Challenge: Refresh Delay

- Auto-refresh : recharges all the memory cells within the "retention time"
  - a rank during refresh becomes unavailable to memory requests until the refresh completes (tRFC).
  - all bank row buffers of this rank closed (tRP) and need to be re-opened (tRAS)
  - More bank row buffer misses around refreshes.

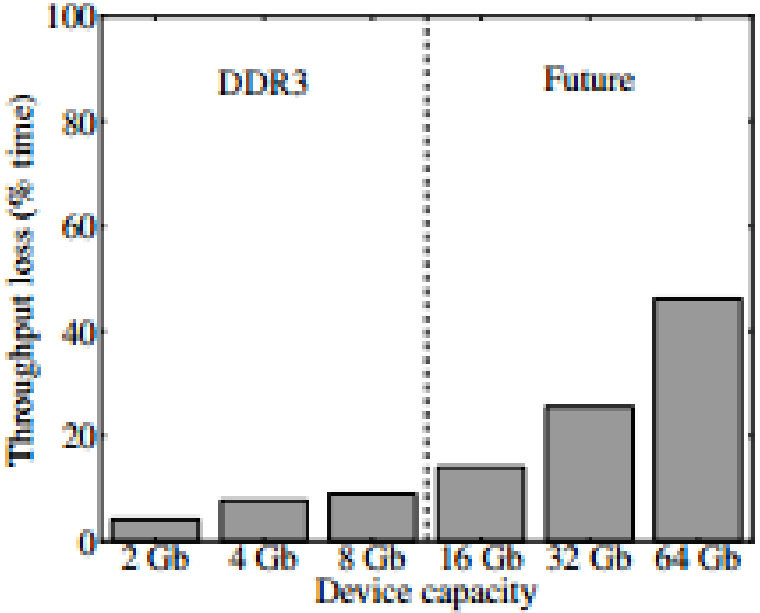
1. Increase in memory latency
2. Significant fluctuation of memory reference latency.

Challenge: Refresh Delay

- As density and size of DRAM grow:
  - more rows required per DRAM chip
  - longer tRFC
  - higher probability for refresh interference



(a) Refresh latency



(b) Throughput loss

Challenge: Refresh Delay

- As density and size of DRAM grow:
  - more rows required per DRAM chip
  - longer tRFC
  - higher probability for refresh interference

1. Increases length a refresh operation
2. Reduces memory throughput

Colored Refresh for Cyclic Executive (CE)

- Partitions DRAM memory at rank granularity
  - Refresh rotate round-robin from rank to rank.
  - Assign Real-time tasks to different ranks via colored memory allocation.
  - Schedule with Cyclic Executive
- By cooperatively scheduling real-time tasks and refresh operations, memory requests no longer suffer from refresh interference

Colored Refresh Design for CE

- For a task set $T = T_1 \ldots T_n$, where T_i is specified by (p_i, D_i, e_i),
- R = DRAM retention time
- K is the number of DRAM ranks, **f** denotes the **frame size**.
- Select an appropriate f based on the following 5 rules:

1. $f \leq \min_{1 \leq i \leq n} (p_i/2)$ → burst refreshes can alternate once tasks are assigned disjoint memory colors
2. $\lfloor R/f \rfloor - R/f = 0$, i.e., f should divide R → refresh synch. to T
3. $2f - \gcd(p_i, f) \leq D_i$ for all I → exec in next frame (std CE)
4. $\lfloor f/(R/K) \rfloor - f/(R/K) = 0$, i.e., R/K divides f → sync & color frames
5. $f \geq \max_{1 \leq i \leq n} (e_i)$ → job can exec. non-preempt. in frame (std CE)

Colored Refresh Design

- Theorem (high level): For any task set
  - If schedulable as a cyclic executive under auto-refresh
  - Also schedulable under Colored Refresh.

- With Colored Refresh, a memory rank is either accessed by a processor or refreshed by the DRAM controller at any time, but not both

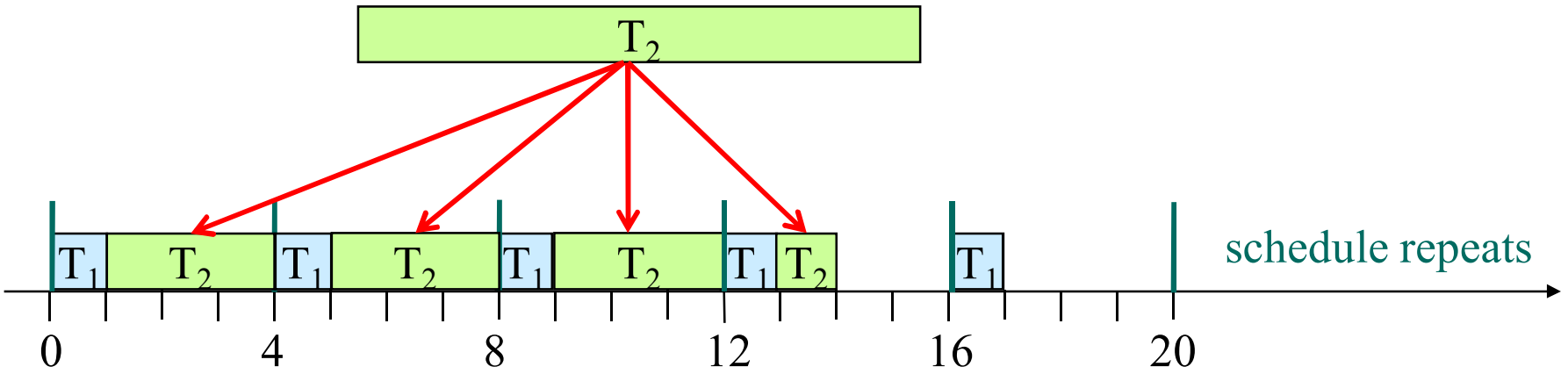
Making CE Refresh Synchronous

- Theorem: Tasks of any CE can be schedule in $(0, D_{min}/2]$, where $D_{min}=\min(D_1, \ldots, D_n)$
- Proof: → paper… in a nut shell:
  - Constructive algorithm to transform CE → refresh sync CE
  - Trivial if CE already meets constraints, exceptions:
  - Split tasks into subtasks to meet (5)
  - Task t w/ short period or period does not divide $f_{min}=R/K$
    → create copy tasks, t and t', of different color, "alternate"
  - Very small f causes problems with (1)-(4)
    → fuse frames into larger "virtual frame" $f' \geq R/K$, then color
  - Huge task (atypical) taking more than half of total memory
    → cannot hide refresh complete, but one 1 task to consider for refresh blocking, still benefits from bound in paper
- Detailed discussion of cases → paper

Colored Refresh Design: Task Splitting

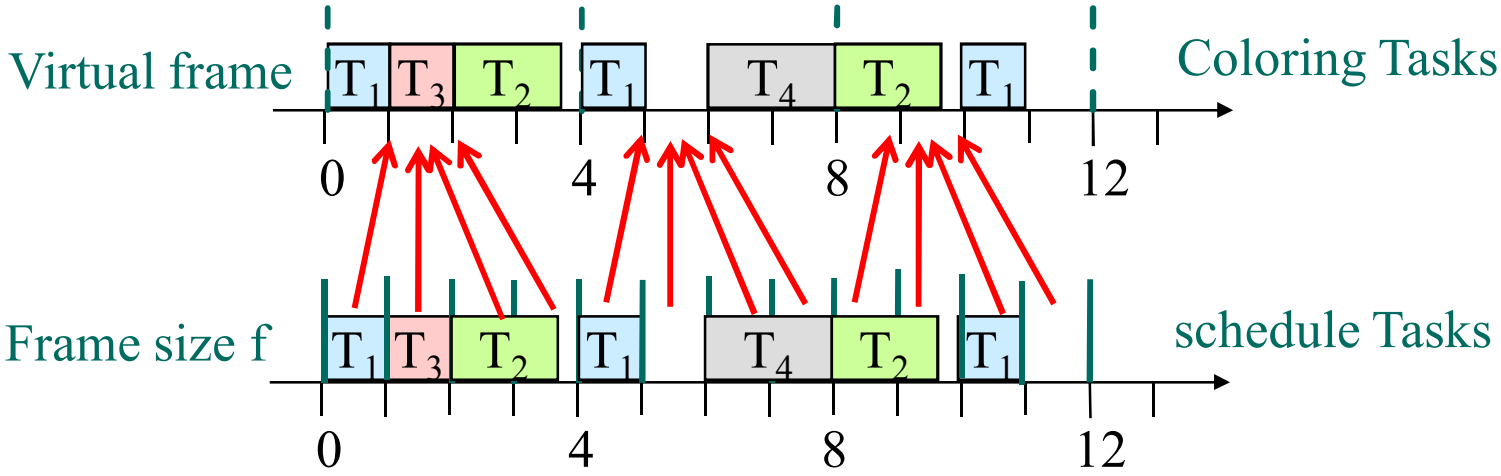
- Task Splitting: split such tasks into job slices
  - Reduce $\max_{1 \le i \le n} (e_i)$ to satisfy rule (5)

- Example:
- T1 (4, 1), T2 (20, 10)



Colored Refresh Design: Frame Fusion

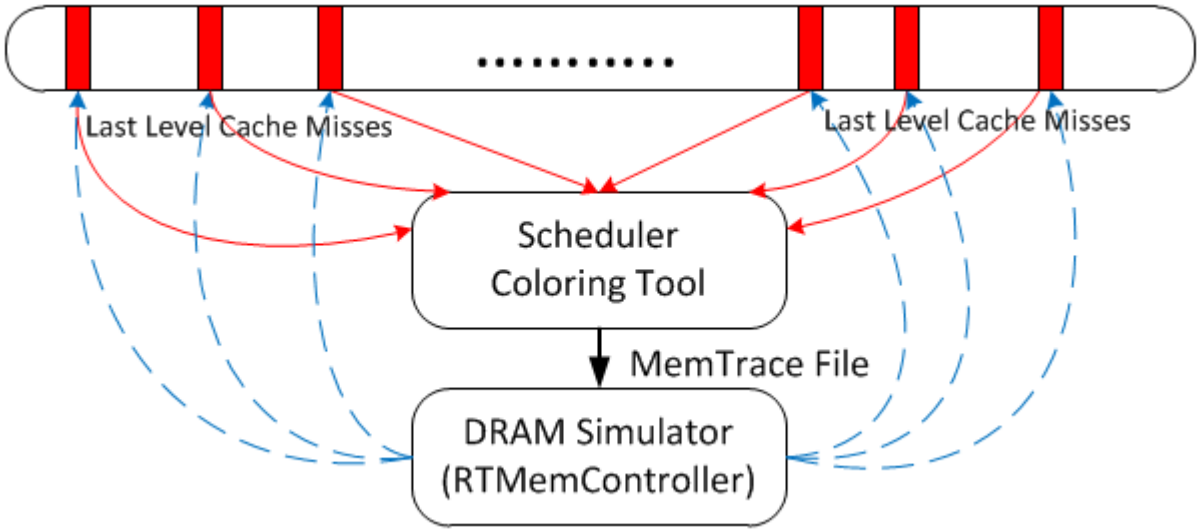
- Task Fusing: fuse multiple frames repeatedly into a "virtual frame",
  - Make a small f derived from rule (3) can satisfy rule (4)
  - For example, $f \leq 1$ from rule (3), while R/K=4



- Task copying: create duplicate instances
  - Increase p_i to satisfy rule (1)

Colored Refresh Implementation

- SimpleScalar
  - simulates execution of application
  - generates memory tracefile
- Scheduler & Coloring Tool
- RTMemController
  - schedule memory transcations and determine access latency



Experimental Setup

- Single core processor
  - split 16KB data and instruction caches,
  - unified 128KB L2 cache
  - cache line size is 64B.
- JEDEC-compliant DDR3/DDR4 SDRAM
  - varied memory density (1Gb, 2Gb, 4Gb, 8Gb, 16Gb, 32Gb and 64Gb).
- The DRAM retention time (tRET) is 64 ms.
  - 8 ranks (K =8) and one memory controller.
  - Issue refresh at by memory controllers at rank granularity.

Real-Time Tasks

- Malardalen benchmark task set
- Cyclic Executive scheduling policy

Real-time tasks set

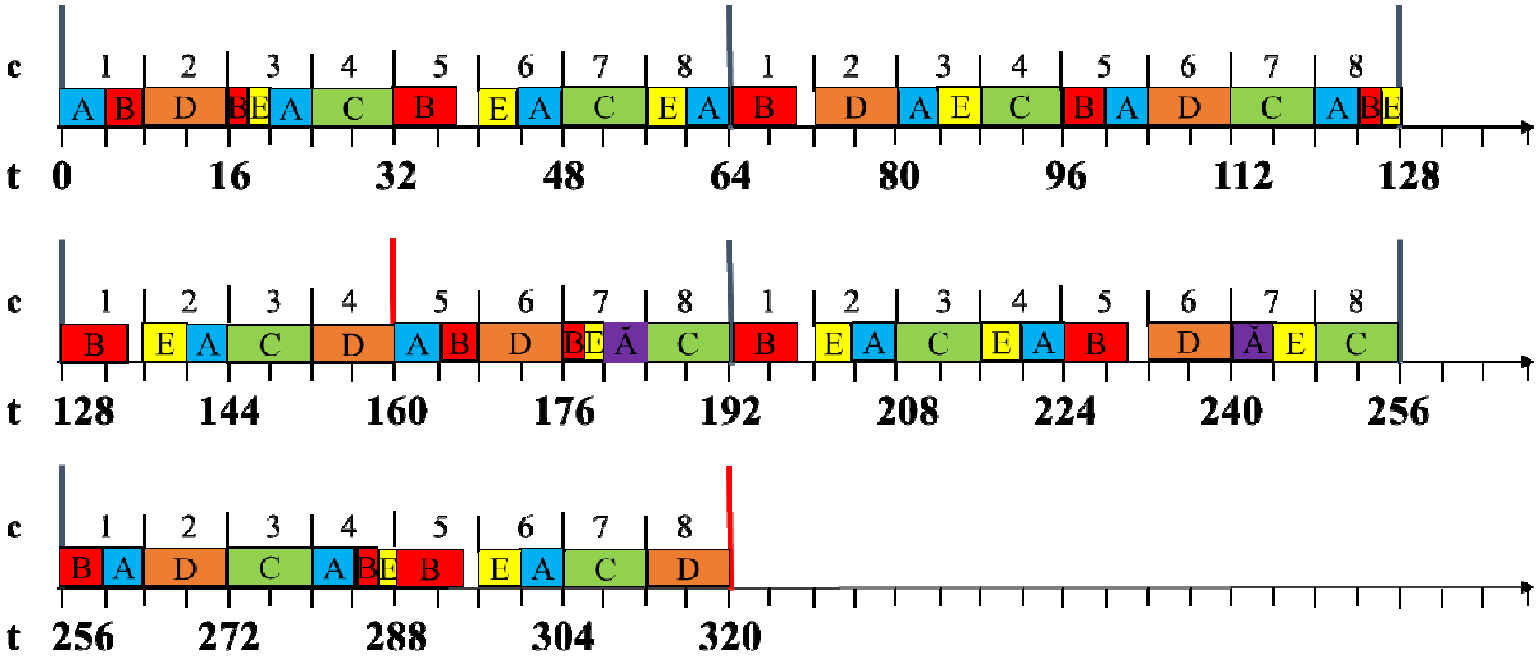
	Execution Time	Period
lms	4 ms	20 ms
compress	6 ms	32 ms
cnt	8 ms	32 ms
st	8 ms	40 ms
matmult	10 ms	80 ms

Real-Time Tasks

- By rules (1)-(5) in Colored Refresh, f =8ms

Program	Occupied frames	available colors (c_i)
lms	$f_1 - f_8$	coloring by task copying
compress	$f_1, f_3, f_4, f_5, f_7, f_8$	c_2, c_6
cnt	f_3, f_4, f_7, f_8	c_1, c_2, c_5, c_6
st	f_2, f_4, f_6, f_8	c_1, c_3, c_5, c_7
matmult	$f_2, f_3, f_4, f_6, f_7, f_8$	c_1, c_5

T	program	color
A	lms	c_7
$\hat{A}$	lms copying	c_8
B	compress	c_2
C	cnt	c_1
D	st	c_3
E	matmult	c_5

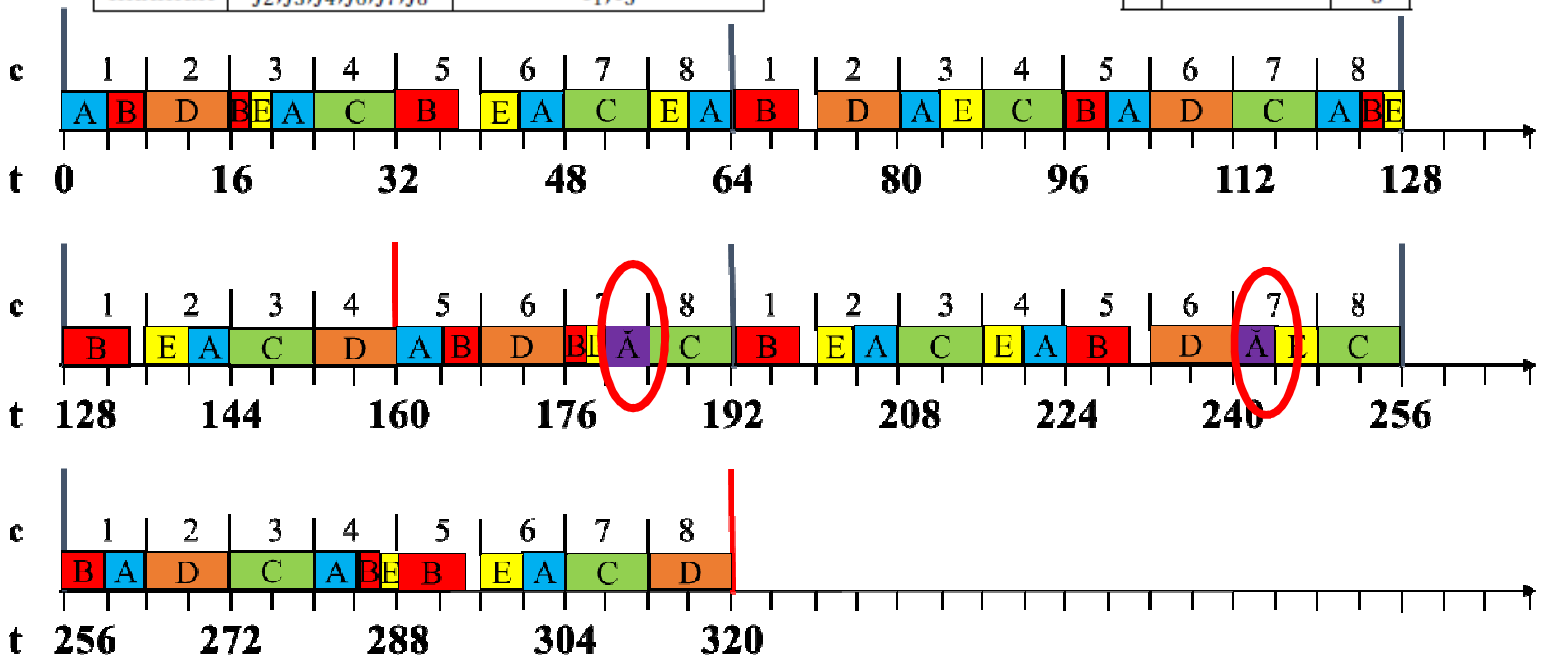


Real-Time Tasks

- By rules (1)-(5) in Colored Refresh, f =8ms

Program	Occupied frames	available colors (c_i)
lms	$f_1 - f_8$	coloring by task copying
compress	$f_1, f_3, f_4, f_5, f_7, f_8$	c_2, c_6
cnt	f_3, f_4, f_7, f_8	c_1, c_2, c_5, c_6
st	f_2, f_4, f_6, f_8	c_1, c_3, c_5, c_7
matmult	$f_2, f_3, f_4, f_6, f_7, f_8$	c_1, c_5

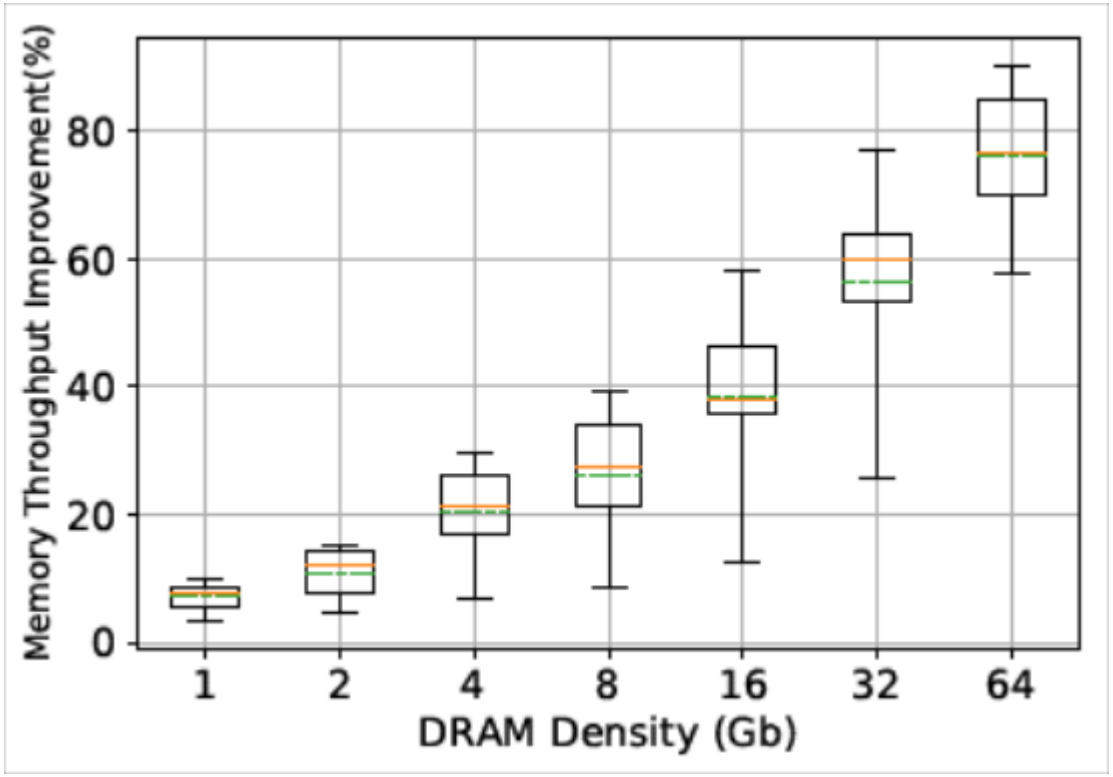
T	program	color
A	lms	c_7
Â	lms copying	c_8
B	compress	c_2
C	cnt	c_1
D	st	c_3
E	matmult	c_5



- Create a copy for lms to hide refresh overhead

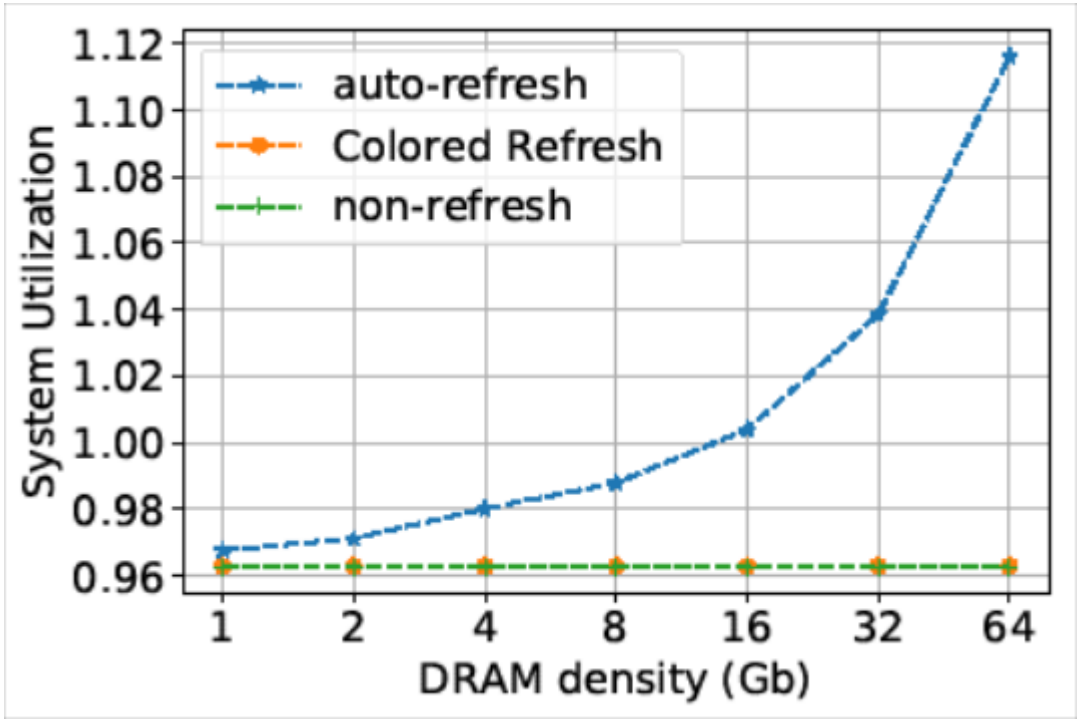
Colored Refresh Results

- Memory Throughput is increased by Colored Refresh



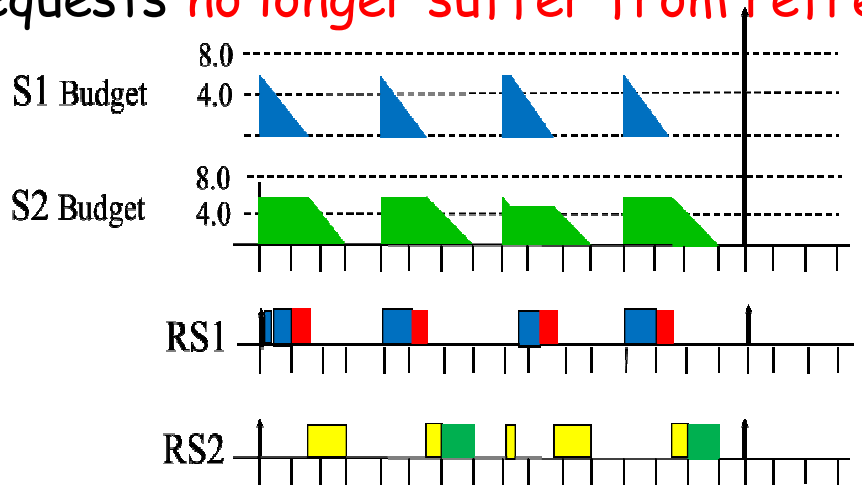
Colored Refresh Results

- Colored Refresh
  - reduce tasks set execution time
  - keeps execution time stable with DRAM density grows



For EDF: Colored Refresh Server [ISORC'19]

- Partition DRAM memory at rank granularity
  - Refreshes rotate round-robin from rank to rank
  - Assign real-time tasks to different ranks via colored memory allocation (say: green,blue)
  - Schedule 2 server tasks to refresh green/blue memory
  - Ensure that no blue task runs when green server active and vice versa: no green task runs when blue server active
- Cooperative scheduling real-time tasks and refresh operations → memory requests no longer suffer from refresh interference



Conclusion

- Make memory references more predictable w/ coloring
  - Controller-Aware Memory Coloring (CAMC) [SAC'18]
    - reduce varied memory access latency
    - provide single core equivalence **but** subject to refresh delay
- ***Colored Refresh for Cyclic Executives [RTSS'19]***: ← this paper
  - hide refresh delays & reduce DRAM access latencies
  - better performance & predictability than auto-refresh [BM'10]
  - Theorem: can hide refresh for any cyclic executive (constructive)
- *Colored Refresh Server for EDF [ISORC'19]*:
  - hierarchical server task scheduling, apps inside servers
  - supports any real-time scheduling policy in server (EDF, RM)
- realized in software, for commercial off-the-shelf (COTS) systems.
- **Supports Core Isolation → real-time composability**
- supported in part by NSF grants 1239246,1329780,1525609 and 1813004.