A Read-Write Aware DRAM Scheduling for Power Reduction in Multi-Core Systems

Authors:

Chih-Yen Lai*, Gung-Yu Pan*, Hsien-Kai Guo*, Jing-Yang Jou*^

Affiliations: * National Chiao Tung University, Taiwan

^ National Central University, Taiwan

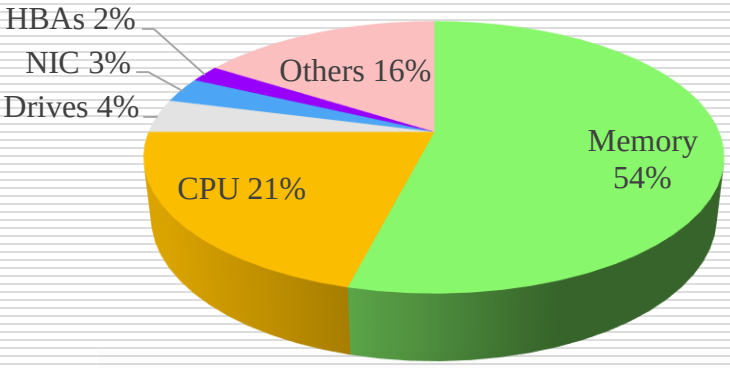
2014/01/23

Outline

- Introduction
  - DRAM power management
  - DRAM architecture
- Related Works and Our Motivation
- Proposed Techniques
- Experimental Results
- Conclusions and Future Works

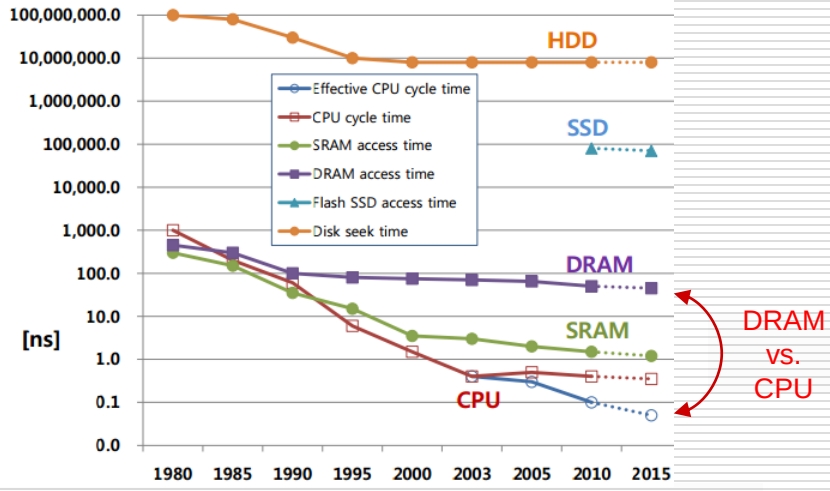
DRAM Power Management

- Demand of low power systems
  - DRAM consumes 25%~60% of the system power [2][3][4]
  - DRAM power management is important
- The memory wall
  - Limits the system performance
  - Makes DRAM power management harder



Power consumption of FBDIMM server

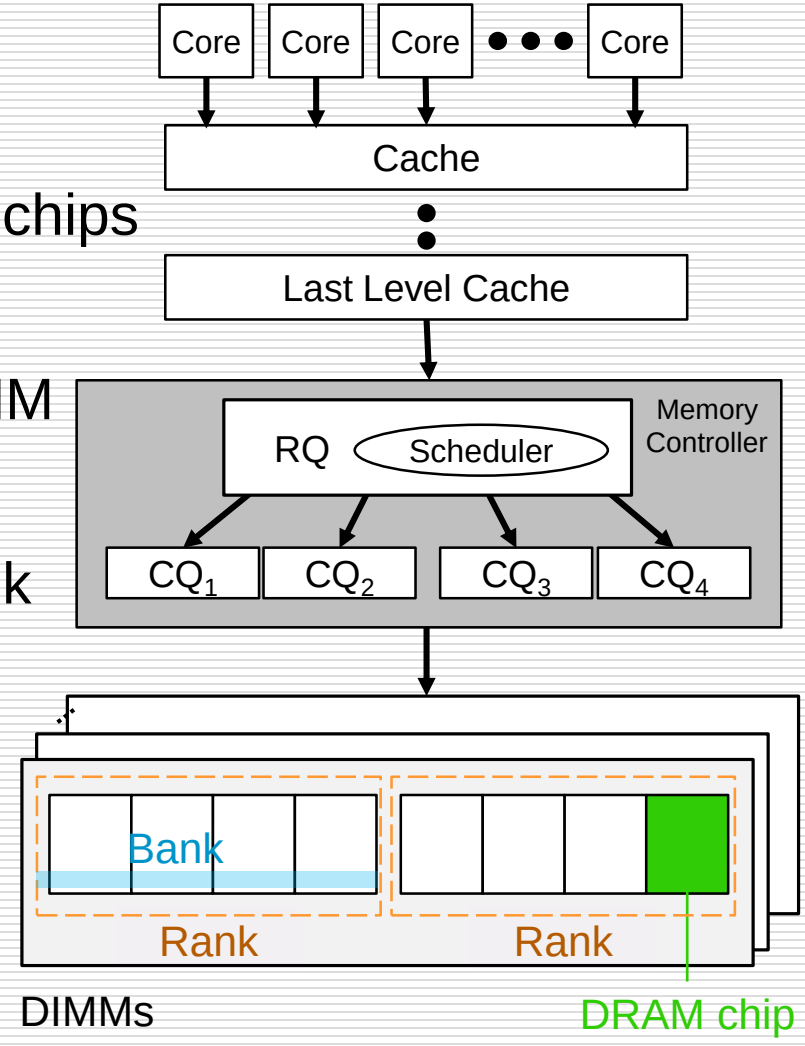
*Source: A speech by Samsung's S. Kadivar at Denali MemCon, 2009



*Source: A speech by Samsung's K. Han, 2012

DRAM Architecture

- Dual Inline Memory Module (DIMM)
  - Composed of multiple DRAM chips
- Rank
  - A set of DRAM chips in a DIMM
- Bank
  - Spreads across chips in a rank
- Memory controller
  - Reorder Queue (RQ)
  - Command Queue (CQ)
  - Scheduling
  - Power mode switching



Outline

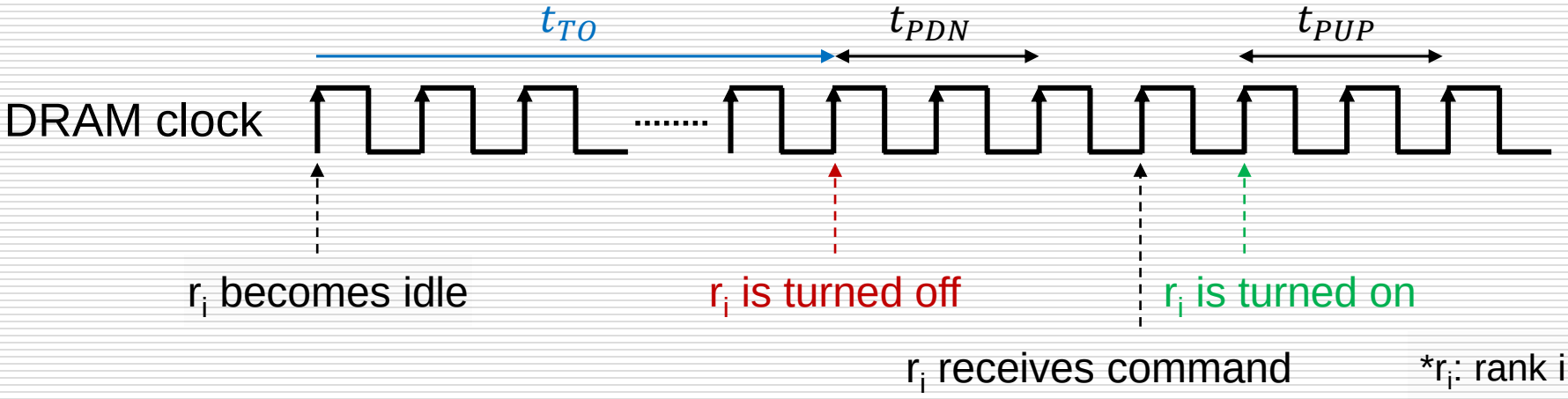
- Introduction
- Related Works and Our Motivation
  - Related works
  - Motivation
- Proposed Techniques
- Experimental Results
- Conclusions and Future Works

Related Works

- Hybrid main memory
  - PDRAM [13][14], cached DRAM [15]
- Re-designing the physical structure of the DRAM
  - Array rearrangement [2], Mini-rank [16]
- Adding hardware component to the DRAM
  - Intelligent refresh [17], automatic data migration [18]
- Power management policies on the memory controller
  - Power-down policy
  - Scheduler
  - Throttling mechanism

Power Management Policies

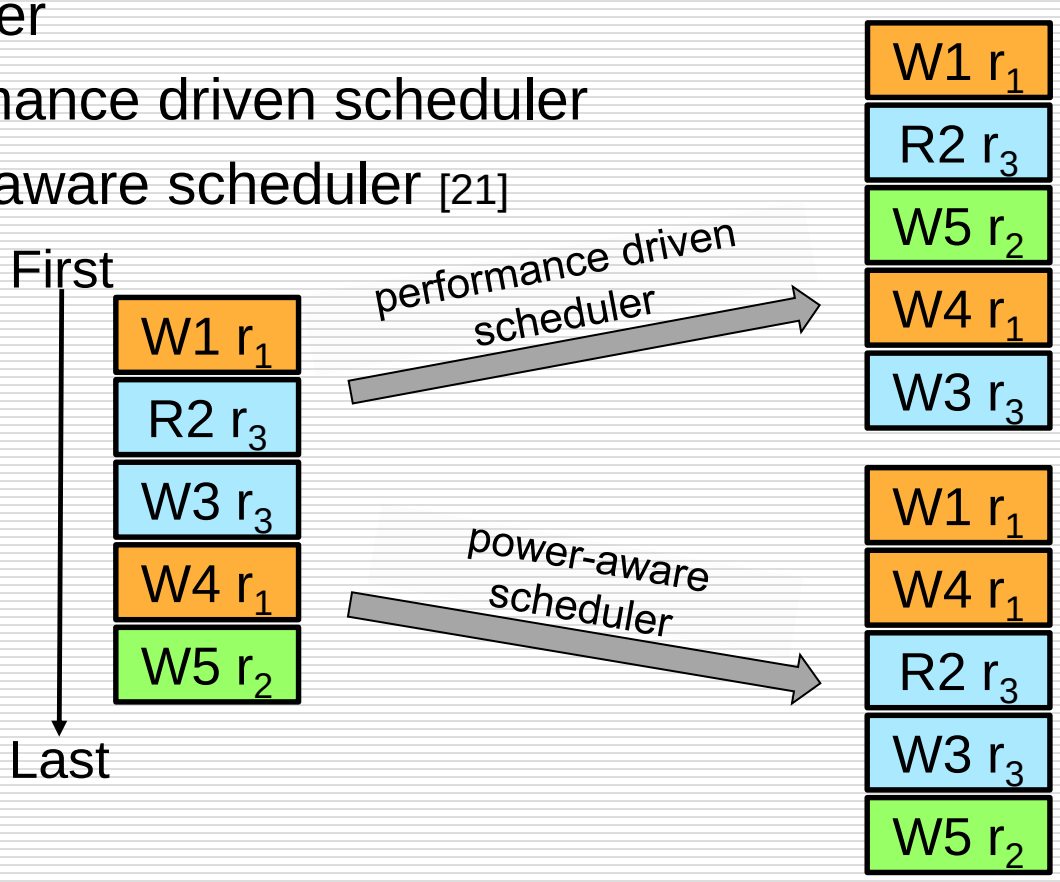
- Power-down policy
  - Determine when to turn off idle ranks
  - Granularity: minimum number of chips to be switched
  - Overhead: transition delays
  - Time-out power-down [19]



  - Queue-aware power-down [21]

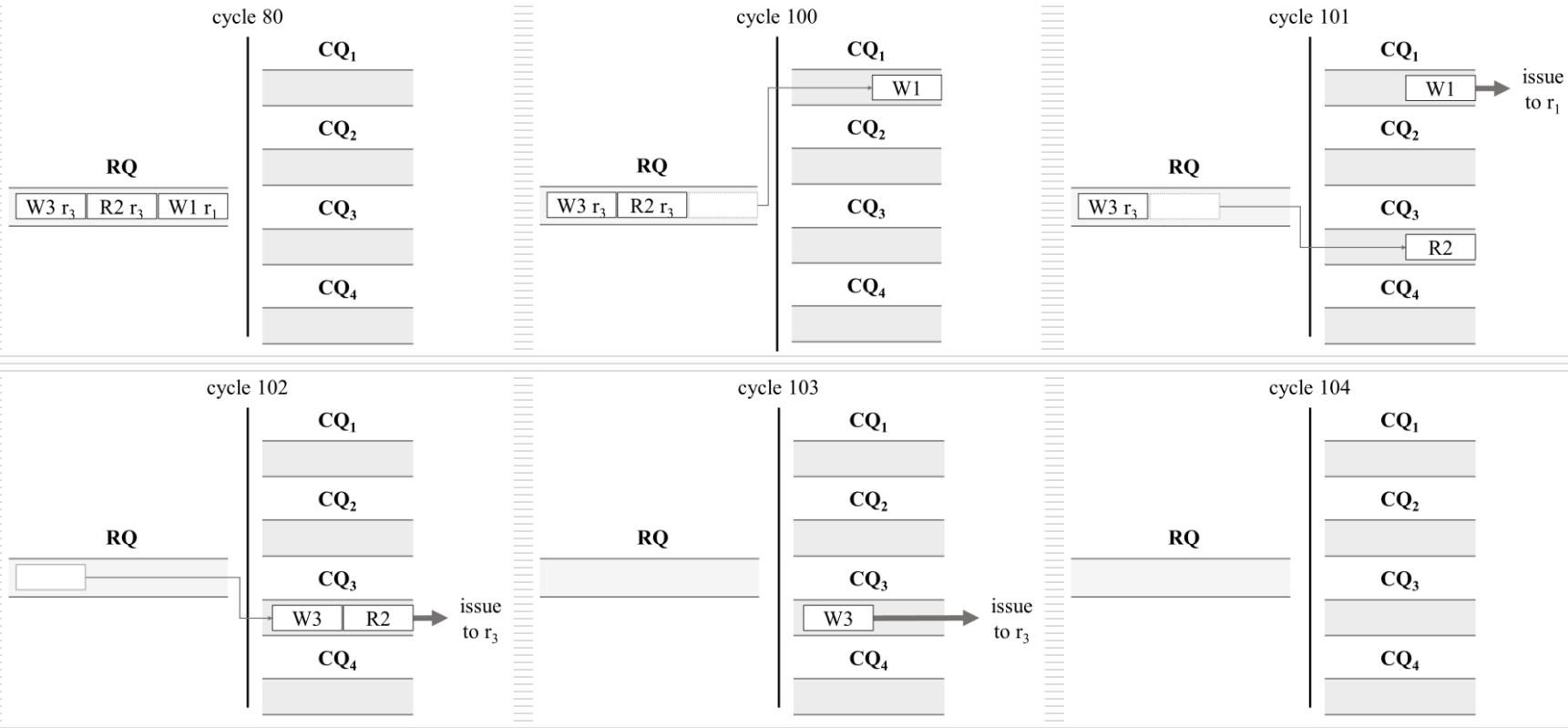
Power Management Policies

- Scheduler
  - Schedules the order of request commands in the memory controller
  - Performance driven scheduler
  - Power-aware scheduler [21]



Power Management Policies

- Throttling mechanism
  - Blocks commands until throttle delay (t_{TD}) is reached
  - Ranks stay in the low power mode for longer periods



Our Motivation

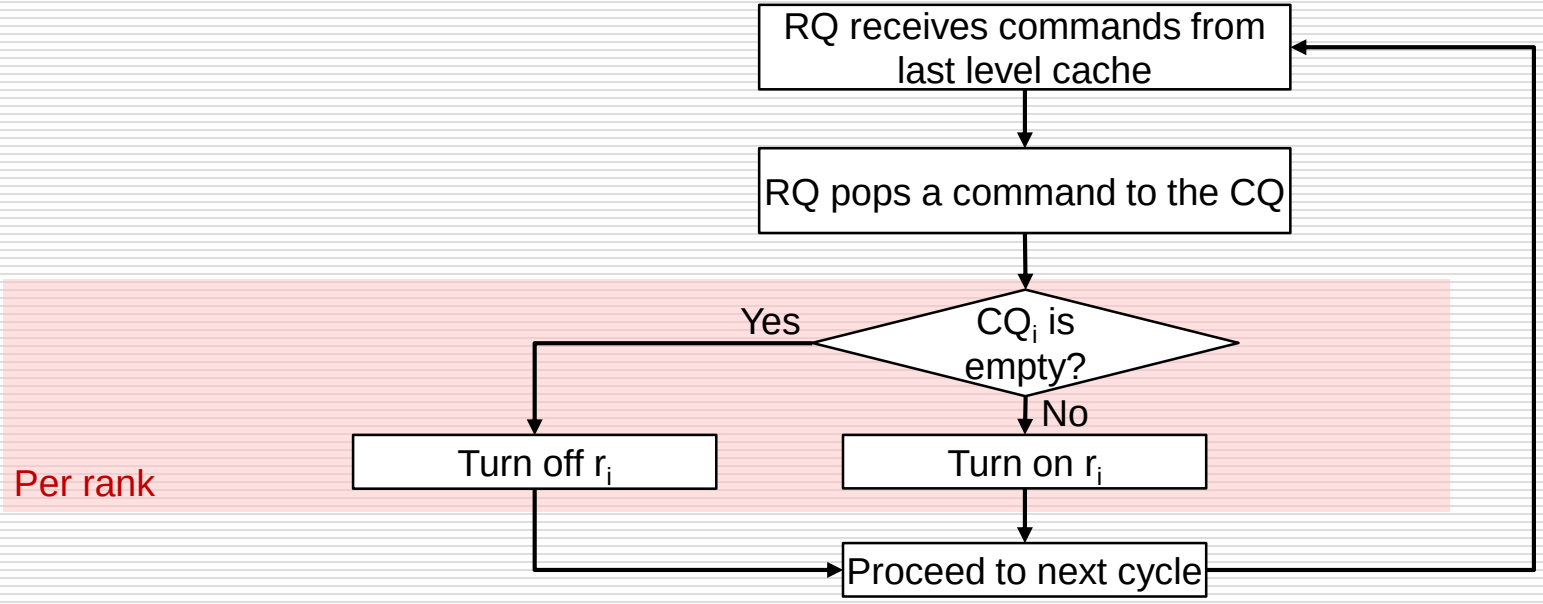
- Reduce power with minimal hardware overhead
- Criticality difference between reads and writes
- Knobs
  - Throttling mechanism
  - Reorder request commands
  - Power mode control
- We propose two techniques
  - Read-write aware throttling
  - Rank level read-write reordering

Outline

- Introduction
- Related Works and Our Motivation
- Proposed Techniques
  - Overview
  - Read-write aware throttling
  - Rank level read-write reordering
  - A complete example
- Experimental results
- Conclusions and Future Works

Overview

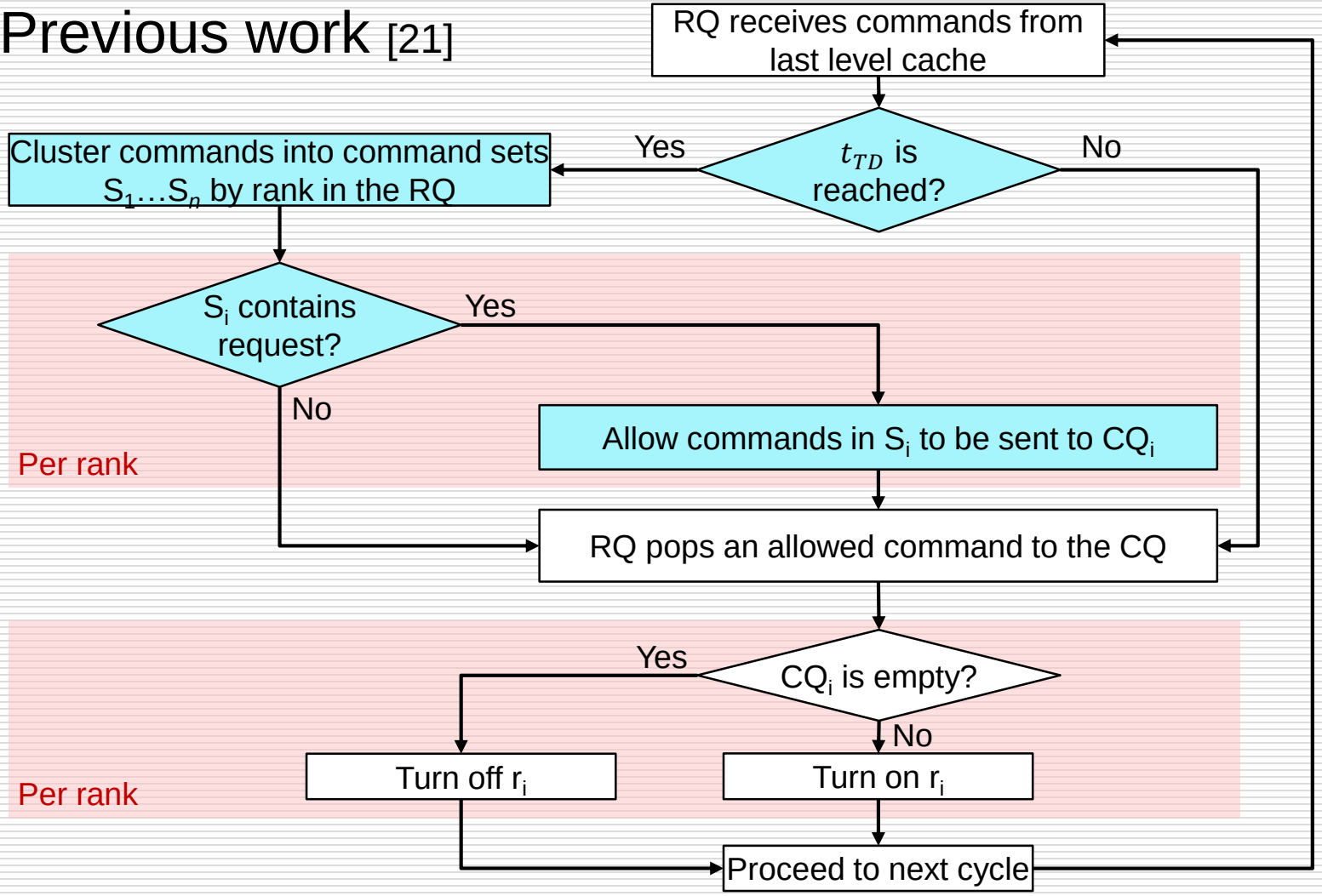
- Greedy power-down memory controller



- Greedy power-down memory controller
- Previous work [21]

Overview

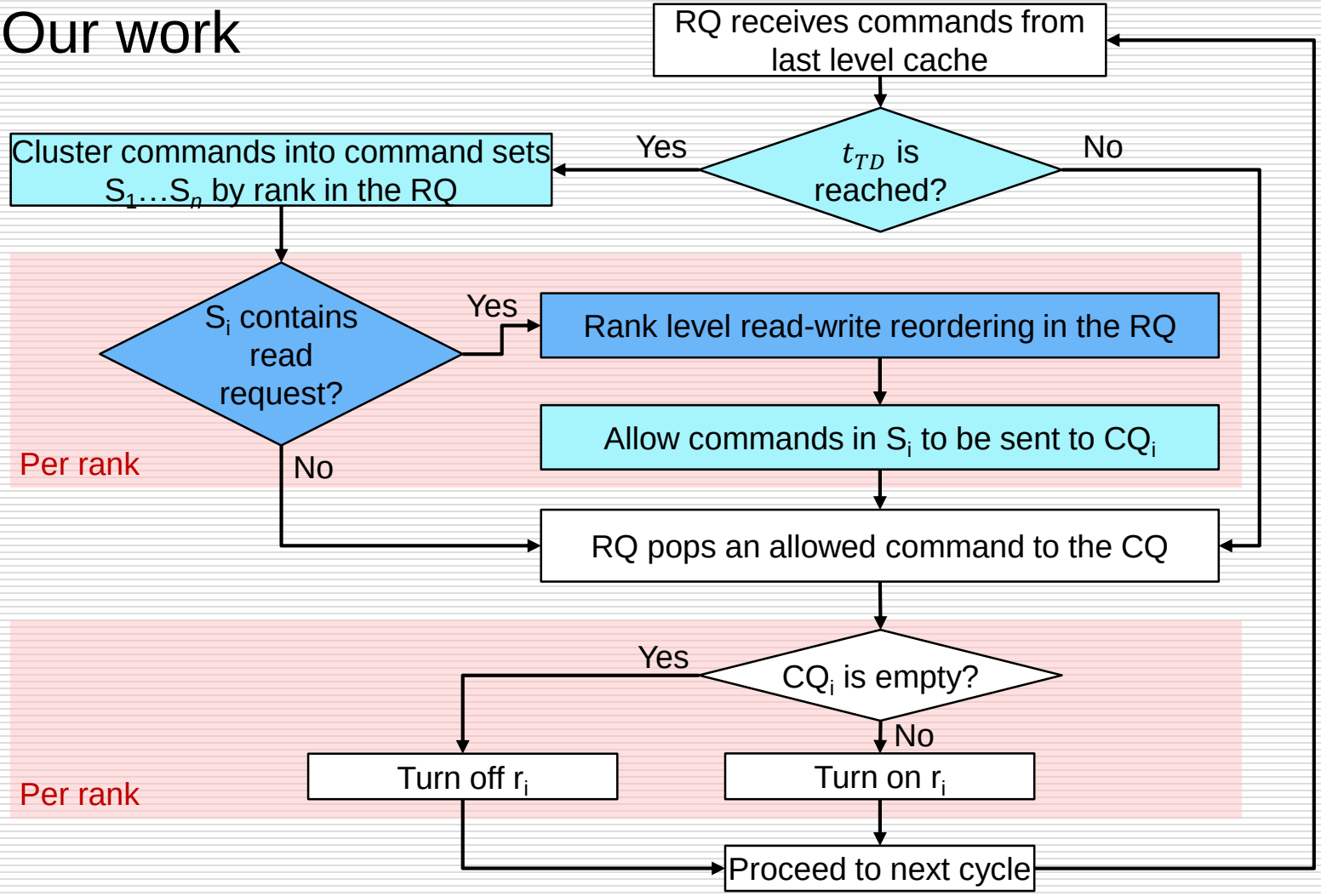
Previous work [21]



- Greedy power-down memory controller
- Previous work [21]
- Our work

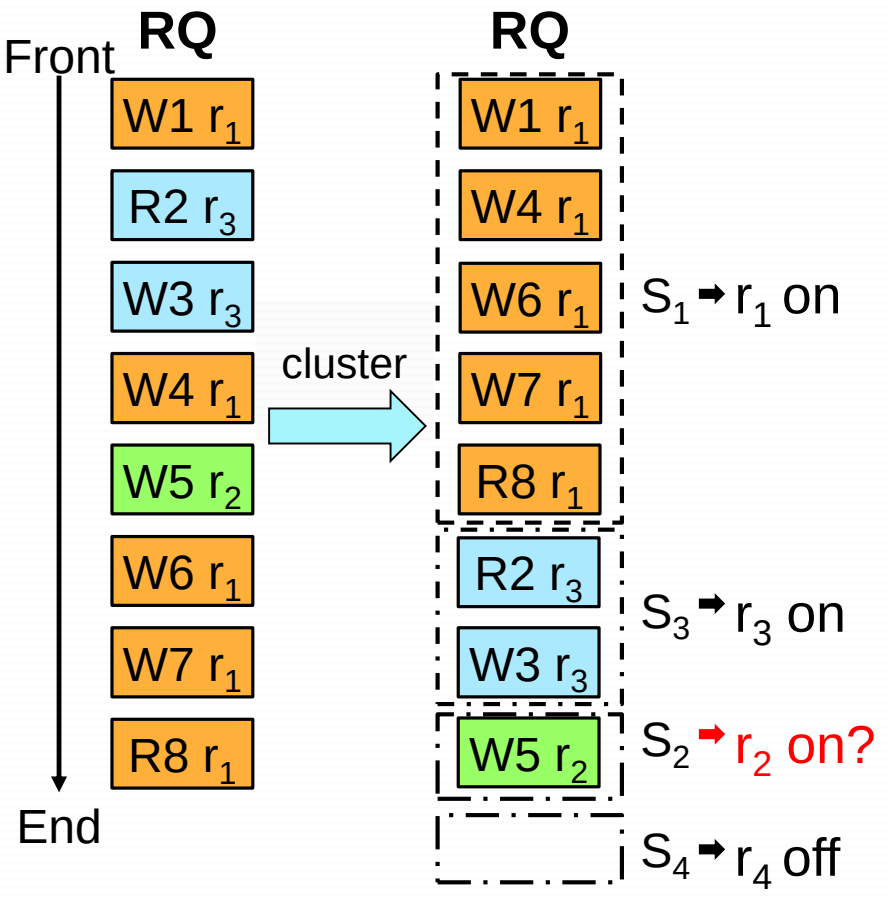
Overview

Our work



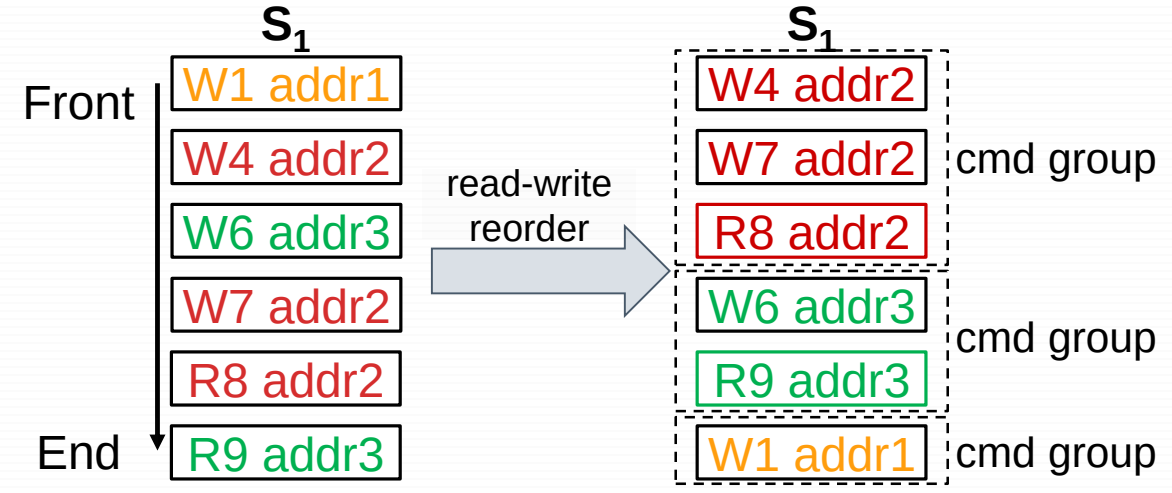
Read-Write Aware Throttling

- Based on basic throttling [21]
  - Blocks commands in the RQ Clusters commands into command sets $S_1, S_2, \ldots, S_n$
- Read-write aware throttling
  - Read requests in each command set?
    - Urgent rank vs Trivial rank
  - Only allows commands targeting urgent ranks to be sent to the corresponding CQs

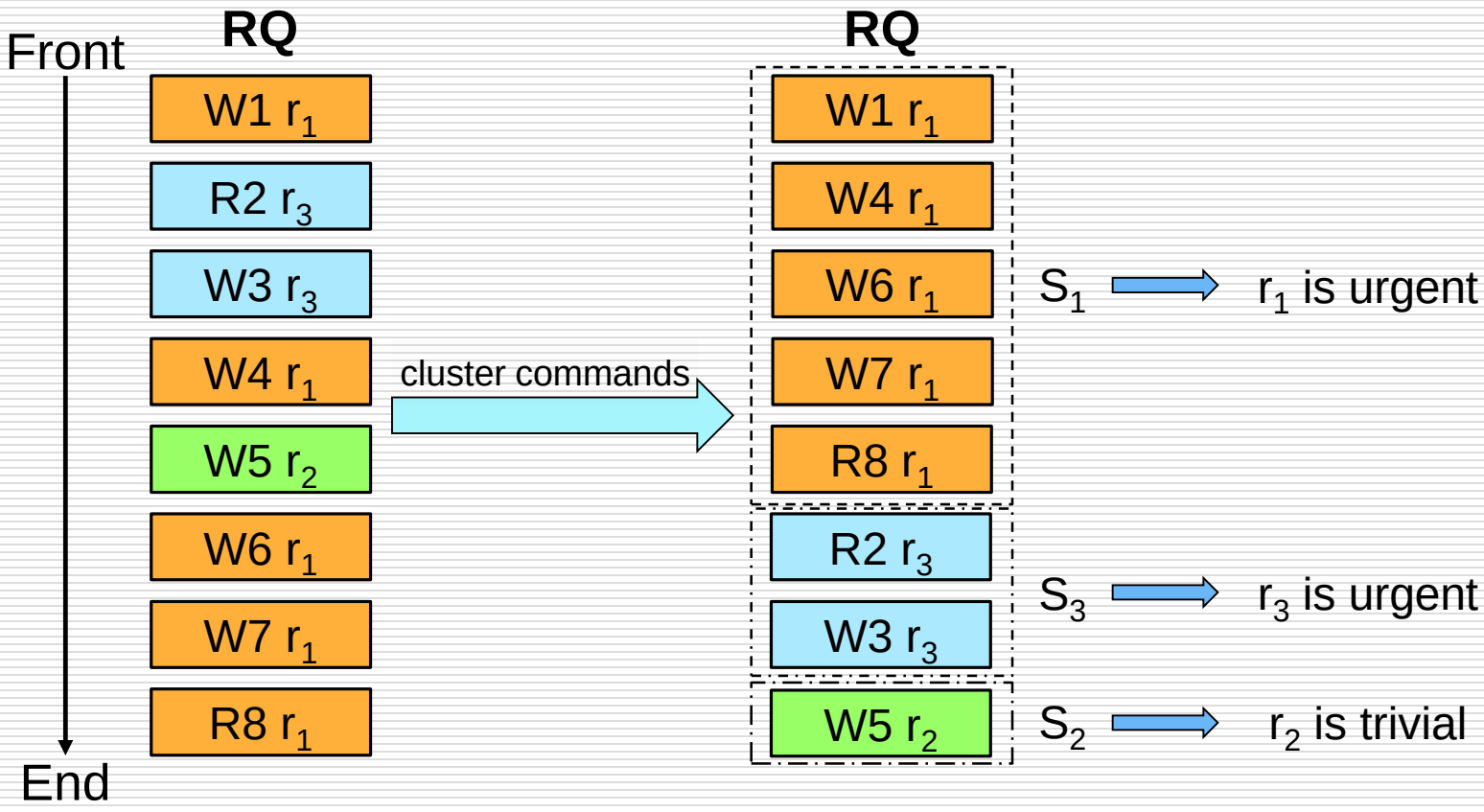


Rank Level Read-Write Reordering

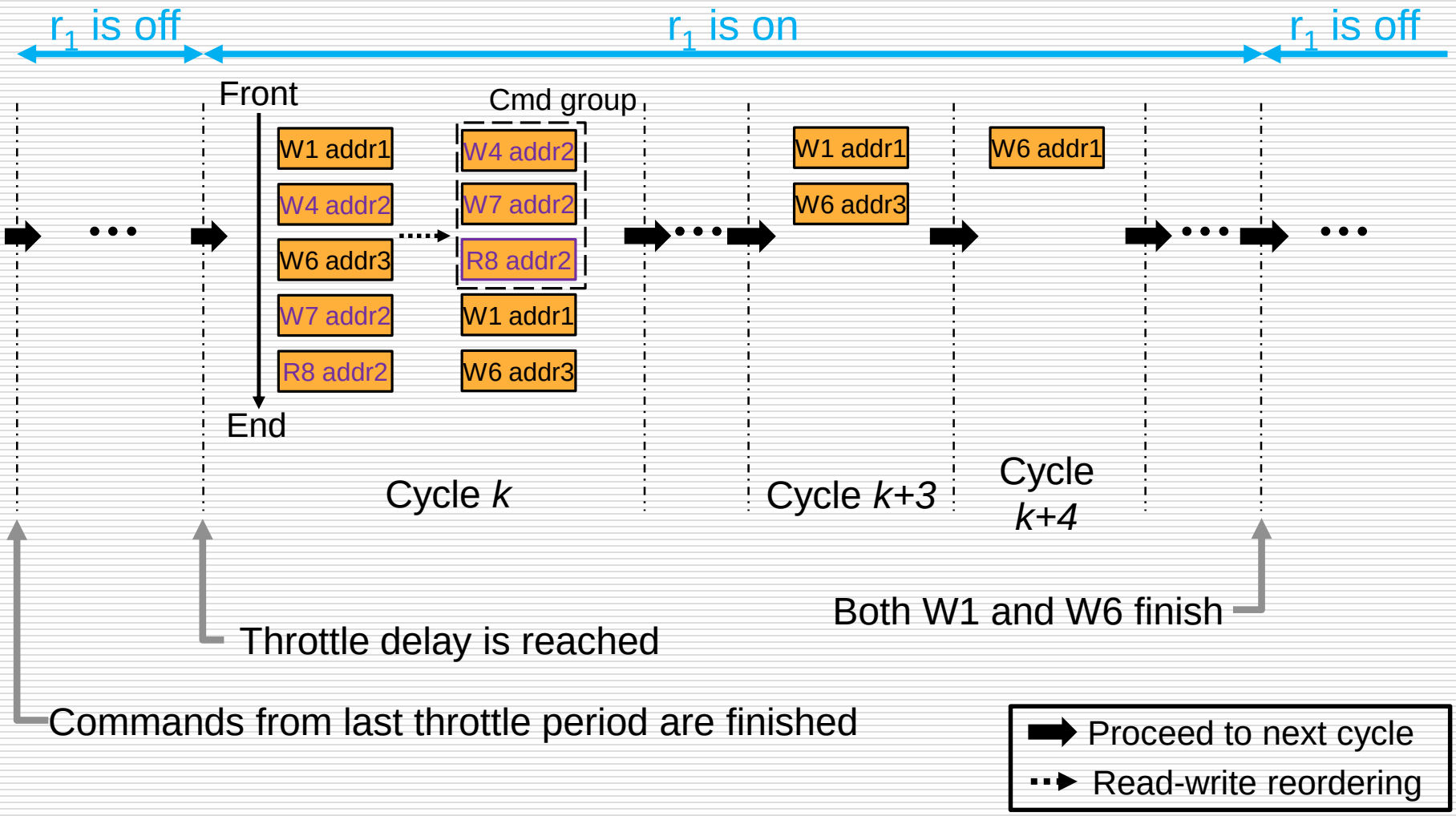
- Reorder commands within a command set
  - Lets the read requests be processed as early as possible
  - Groups commands based on target addresses
    - At most one read request in a command group
  - FIFO order for commands in a command group
  - FIFO order for read requests across command groups



A Complete Example



A Complete Example



Outline

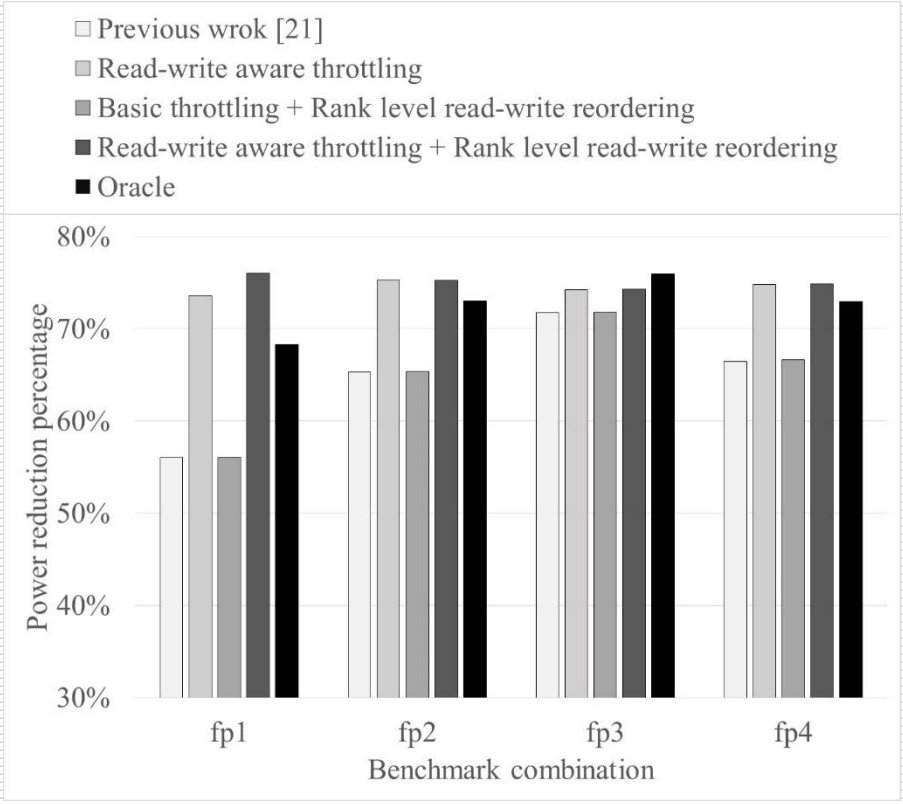
- Introduction
- Related Works and Our Motivation
- Proposed Techniques
- Experimental Results
  - Analysis on different techniques
  - Power and performance trade-off
- Conclusions and Future Works

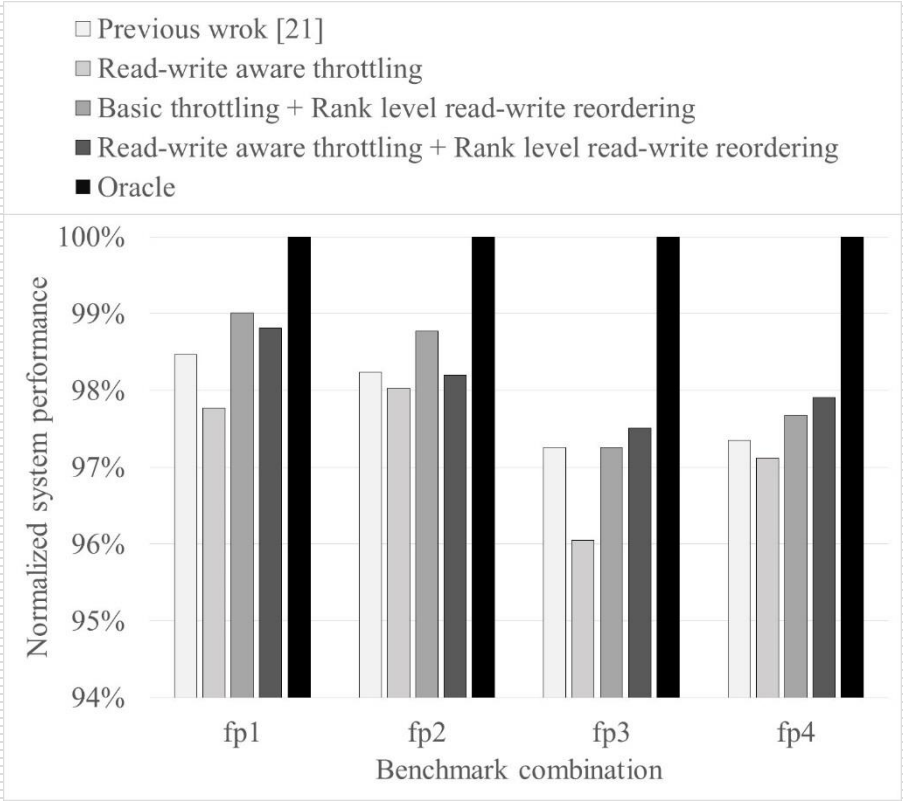
Simulation Environment

- Simulators: Multi2Sim + DRAMSim2
- System: ARM Cortex A9 + DDR2 SDRAM
- Workloads: SPEC CPU2006 (multi-program), SPLASH-2 (multi-threaded)
- Policies
  - Previous work [21]
    - Basic throttling, Power-aware scheduling, Queue-aware power-down
  - Oracle policy
    - Maximum power reduction at zero performance degradation
  - Our work
    - Read-write aware throttling, Rank level read-write reordering

Analysis on Different Techniques

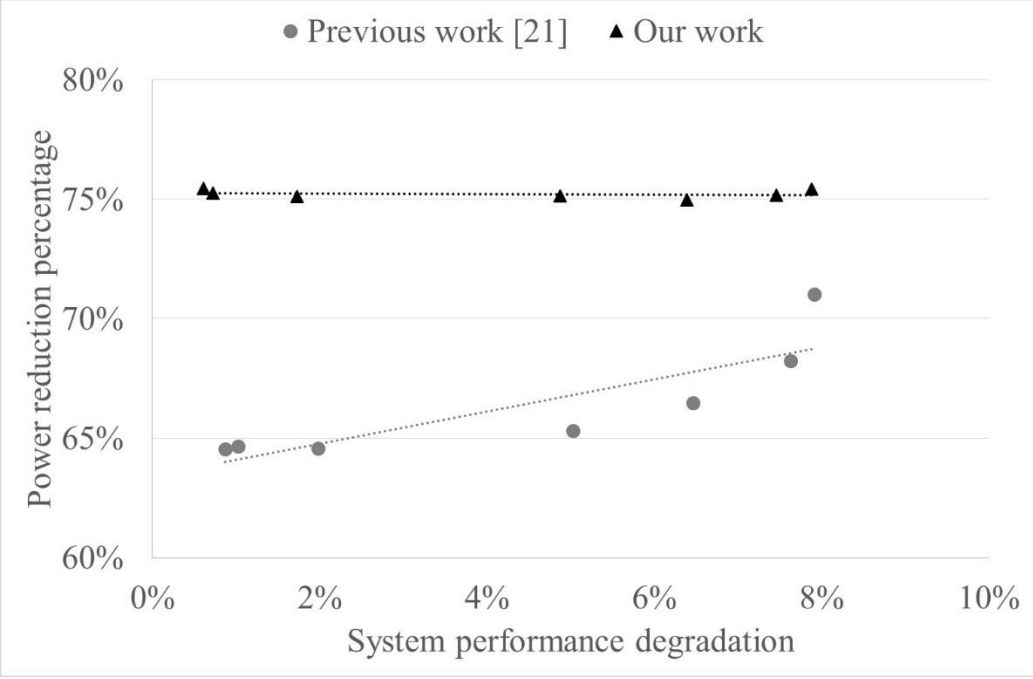
- At $t_{TD} = 400$ CPU cycles





Power and Performance Trade-Off

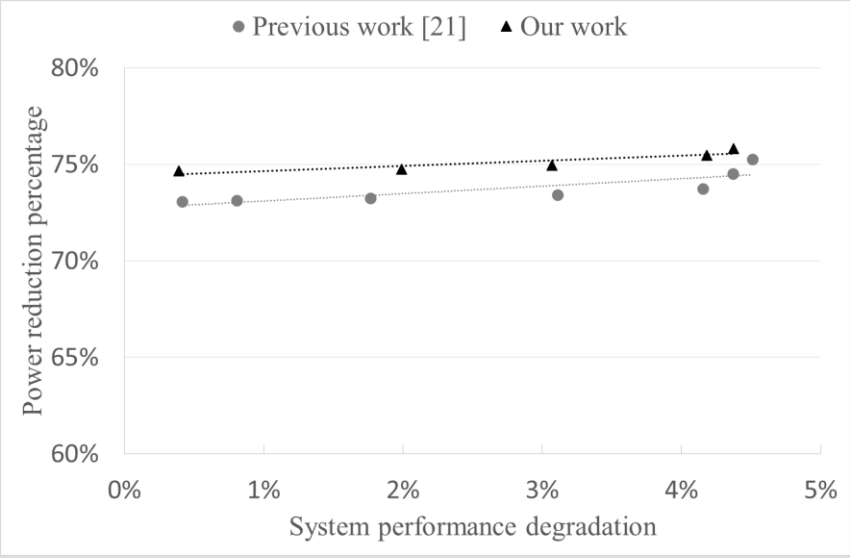
- Trade-off characteristic for SPEC CPU2006



- ~75% of power reduction is close to the upper bound
- Short throttle delays work well for our work

Power and Performance Trade-Off

- Trade-off characteristic for SPLASH-2



	Benchmark (combination)	Main memory requests per million cycles
SPEC CPU2006	fp1	511391.80
	fp2	818090.00
	fp3	31697.60
	fp4	442664.80
SPLASH-2	cholesky	894.36
	fft	2196.41
	fmm	271.93
	radix	801.45
	barnes	38.60

- Small difference in power reduction
- SPLASH-2 benchmarks are less memory intensive

Outline

- Introduction
- Related Works and Our Motivation
- Proposed Techniques
- Experimental Results
- Conclusions and Future Works

Conclusions and Future Works

- Conclusions
  - We propose two techniques
    - Read-write aware throttling
    - Rank level read-write reordering
  - Our work improves the power reduction by 10%~15% on average comparing to the previous work [21]
  - Our work achieves ~75% power reduction at 1%~3% system performance degradation
- Future works
  - Run-time throttle delay controller
  - Associate with automatic data migration, write combining…
  - Utilizing deeper power mode

Thank you