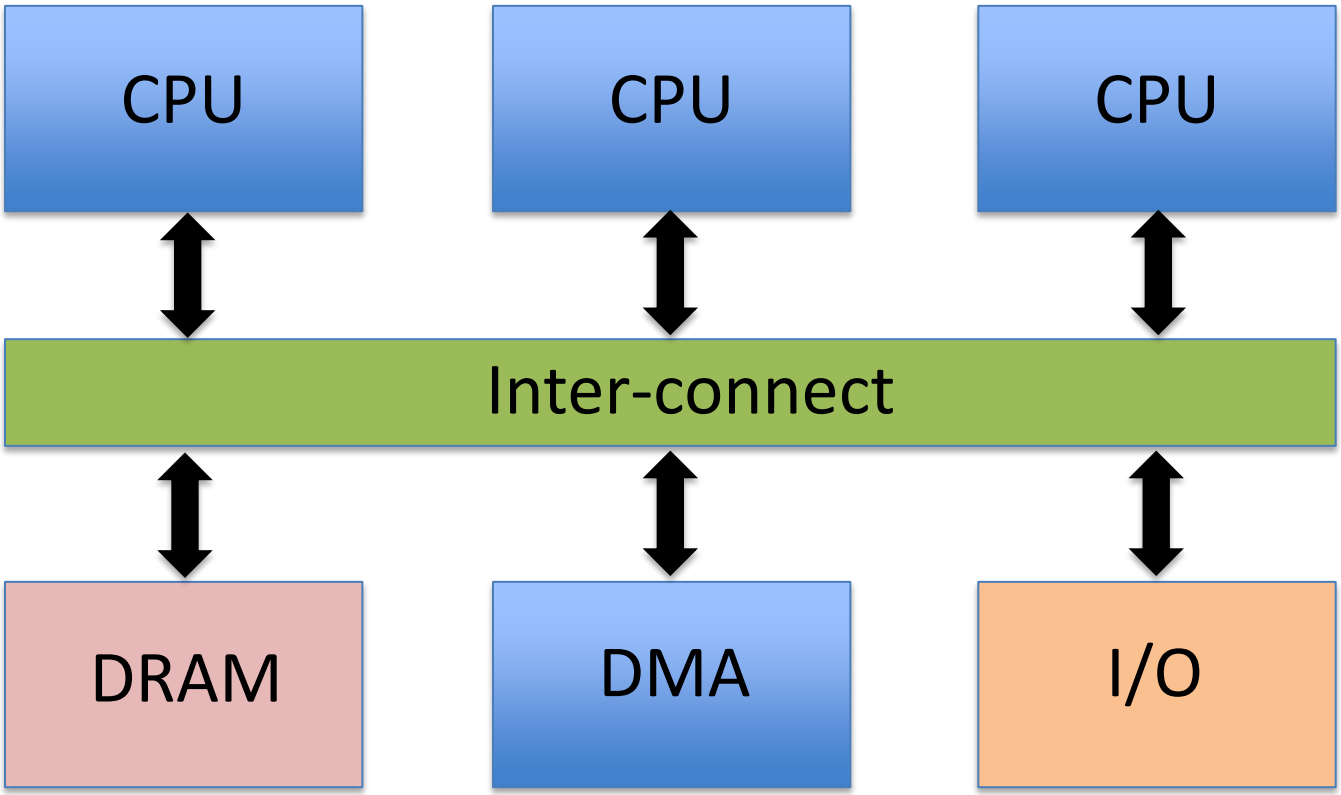
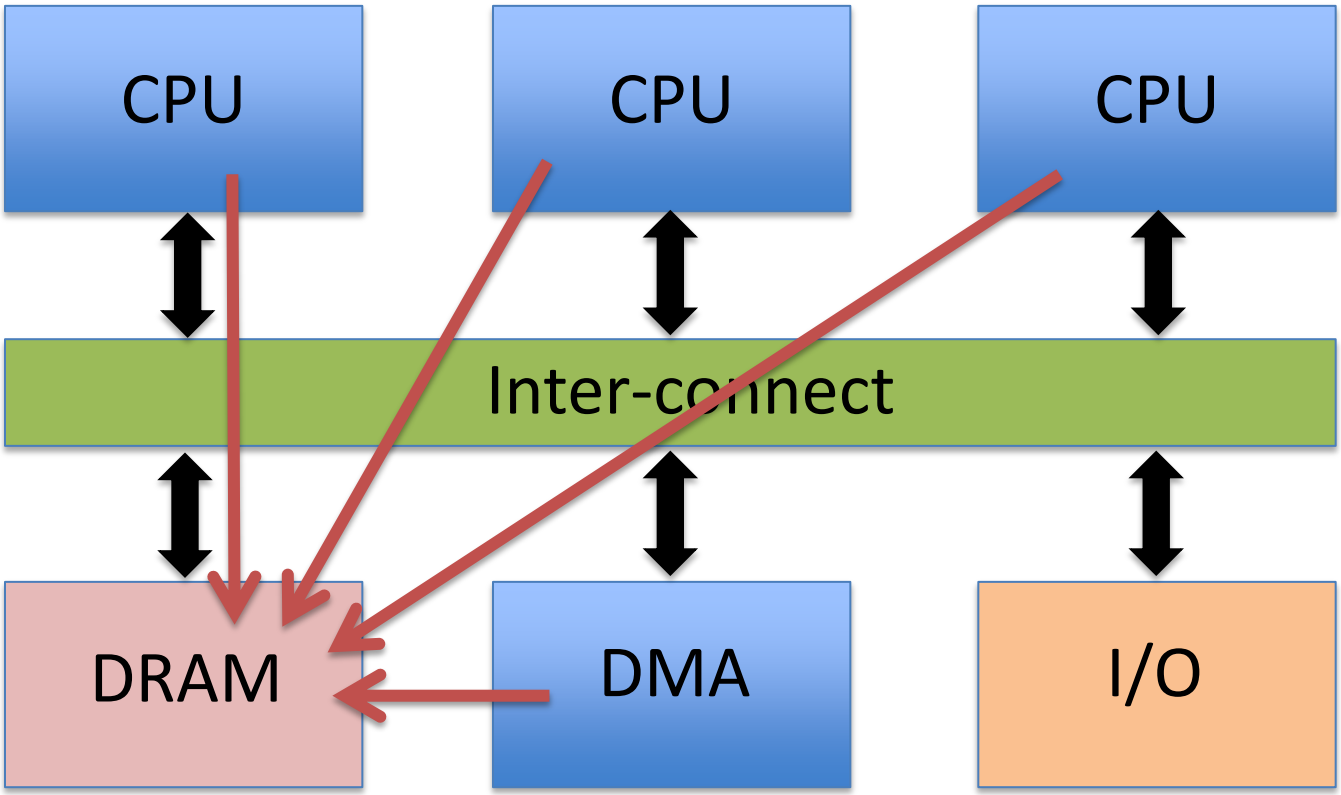
ROC: A Rank-switching, Open-row DRAM Controller for Time-predictable Systems

Yogen Krishnapillai
Zheng Pei Wu
Rodolfo Pellizzoni

Multi-Requestor Systems



Multi-Requestor Systems



INTERFERENCE!!!

Multi-Requestor Systems

- Schedula ut
- WCET d
- WCET: w d resources (e.g. cac
- Existing approaches can bound the interference but they assume the latency for DRAM access is constant

Problem:
DRAM latency is variable and changes depending on its state

DRAM Latency - Solutions

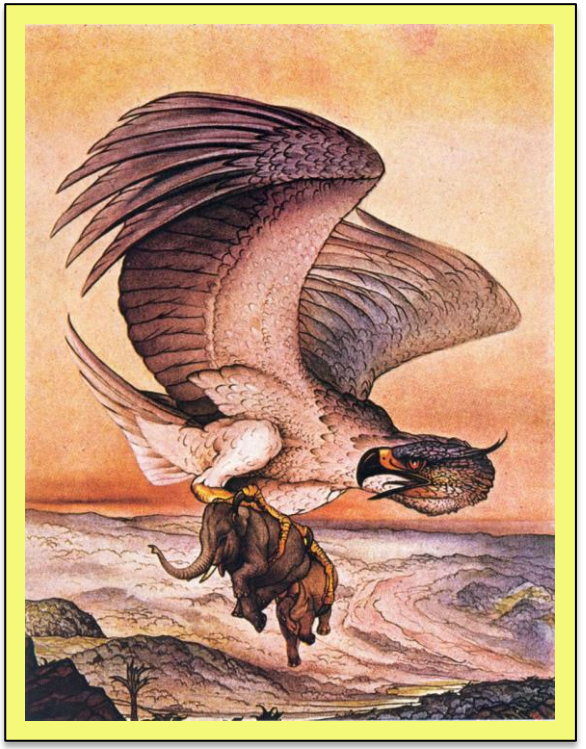
- **Solution 1**: use (complex) analysis to determine upper latency bound
  - Ex: “Bounding Memory Interference Delay in COTS-based Multi-Core Systems”, RTAS’14
  - Problem: COTS DRAM controllers optimized for average case latency; worst case bound can be very pessimistic
- **Solution 2**: predictable DRAM controller
  - Various solutions available
  - Typically simplifies analysis by making latency constant
  - Problem: without architectural optimizations, latency can be high for modern DRAM devices (more on this later)

Our Solution

- Key idea: design new architectural optimizations targeted at reducing worst case, not average case latency
- In this paper: DRAM rank-switching with open-row policy

- **ROC**: **R**ank-switching,
  Open-row **C**ontroller

- We discuss:
  - Design
  - Latency Analysis
  - Implementation
  - Results



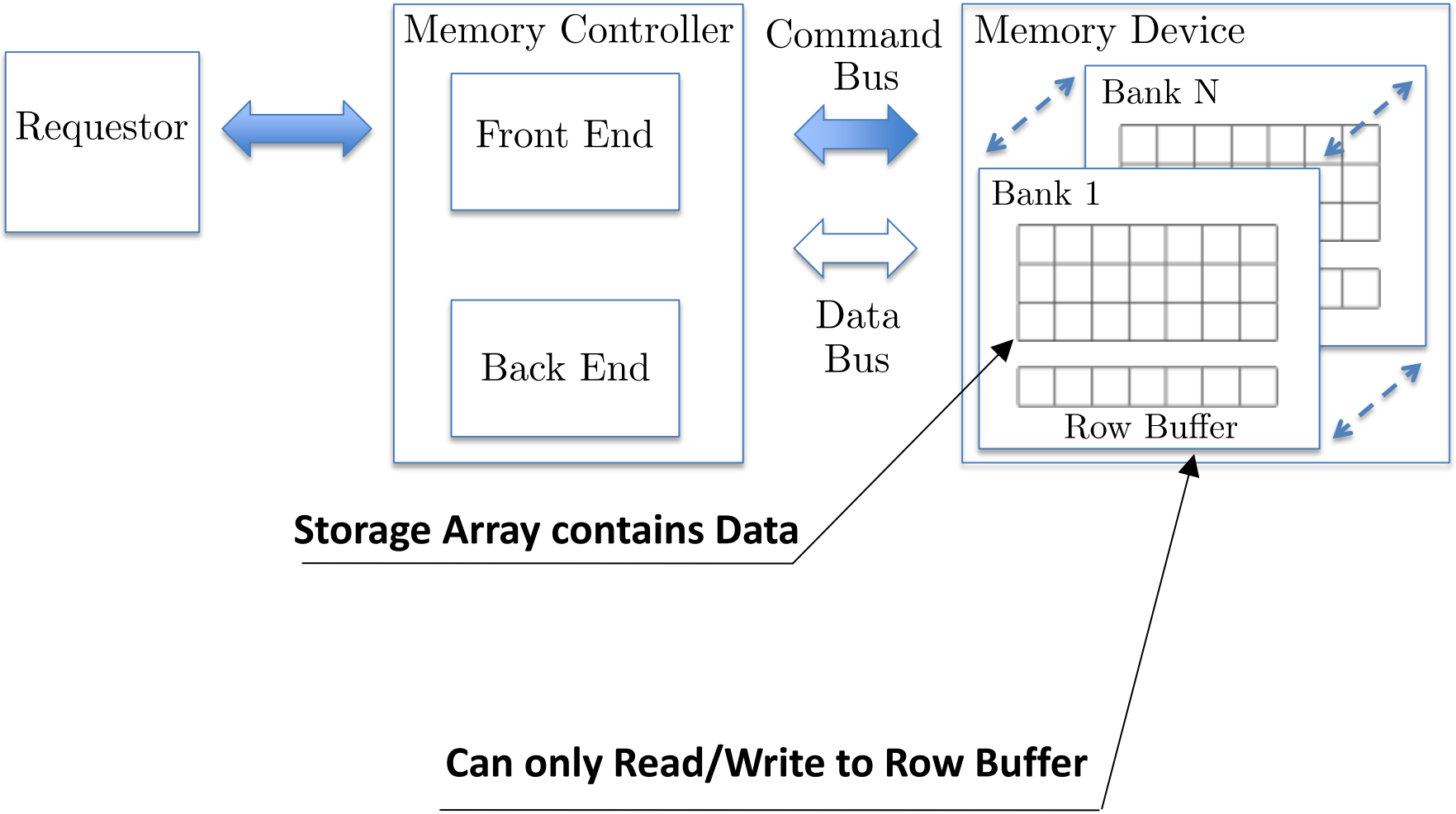
Outline

1. Background & Related Work
2. Rank-Switching Mechanism
3. Worst Case Latency Analysis
4. Memory Controller Model
5. Results & Conclusion

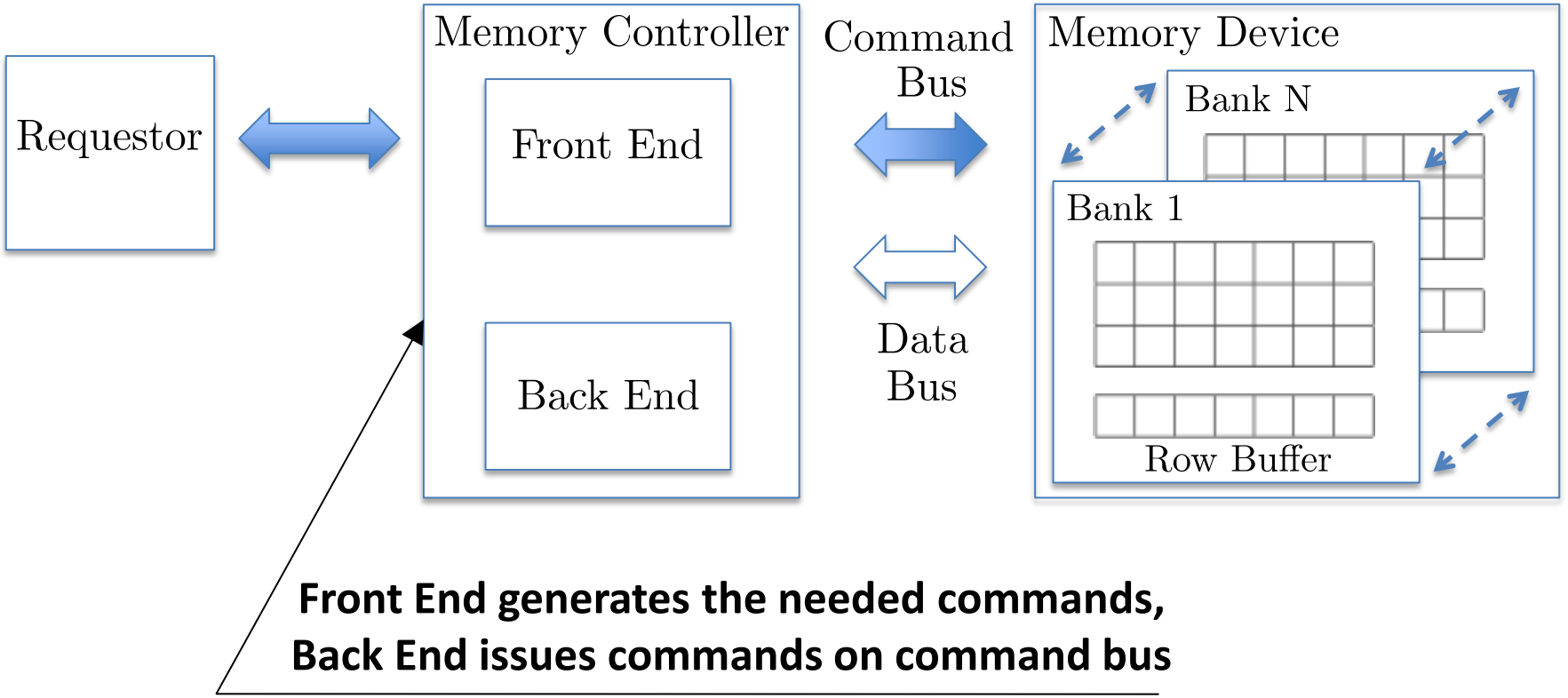
Outline

1. Background & Related Work
2. Rank-Switching Mechanism
3. Worst Case Latency Analysis
4. Memory Controller Model
5. Results & Conclusion

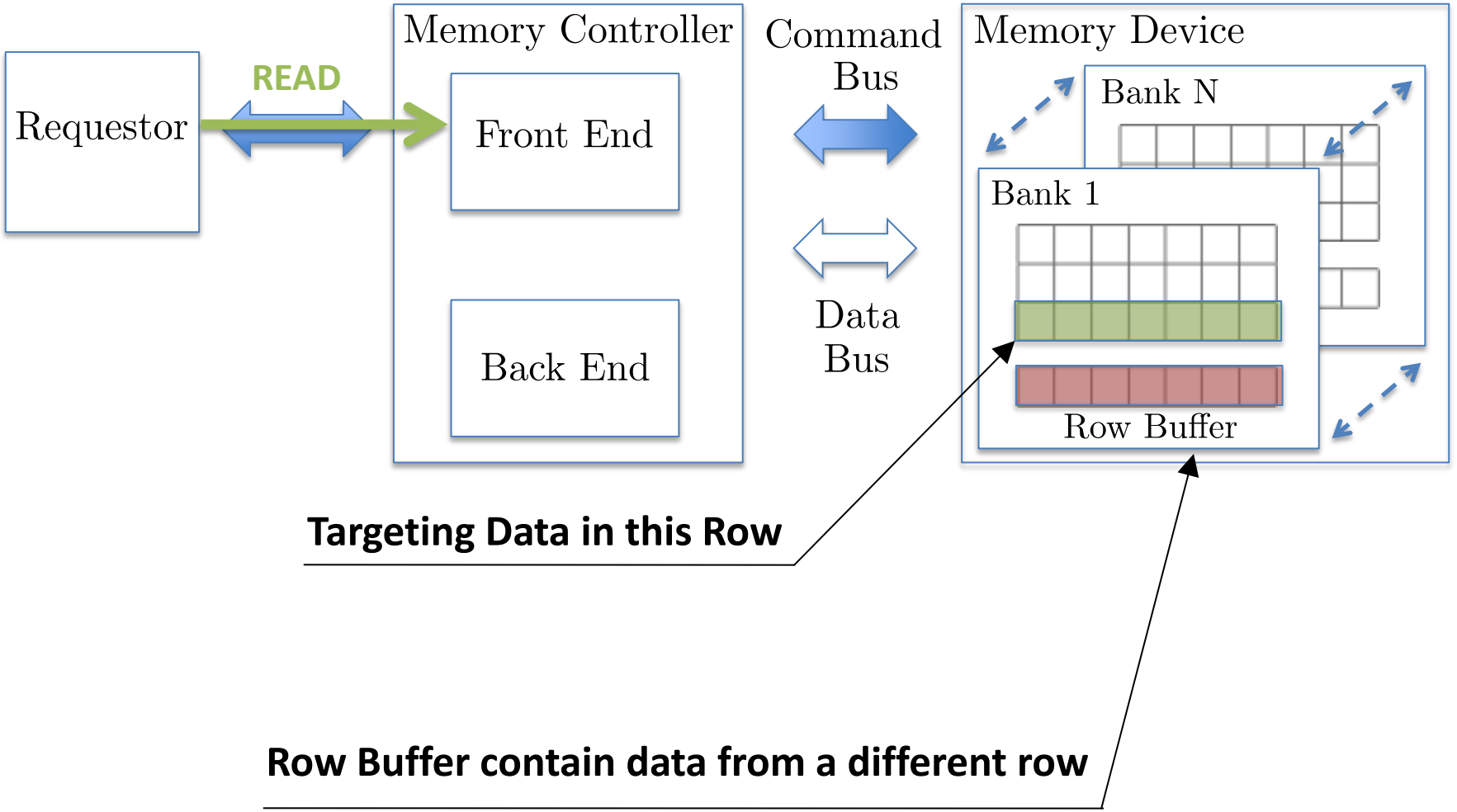
Background


Requestor
Memory Controller
Front End
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer
Storage Array contains Data
Can only Read/Write to Row Buffer

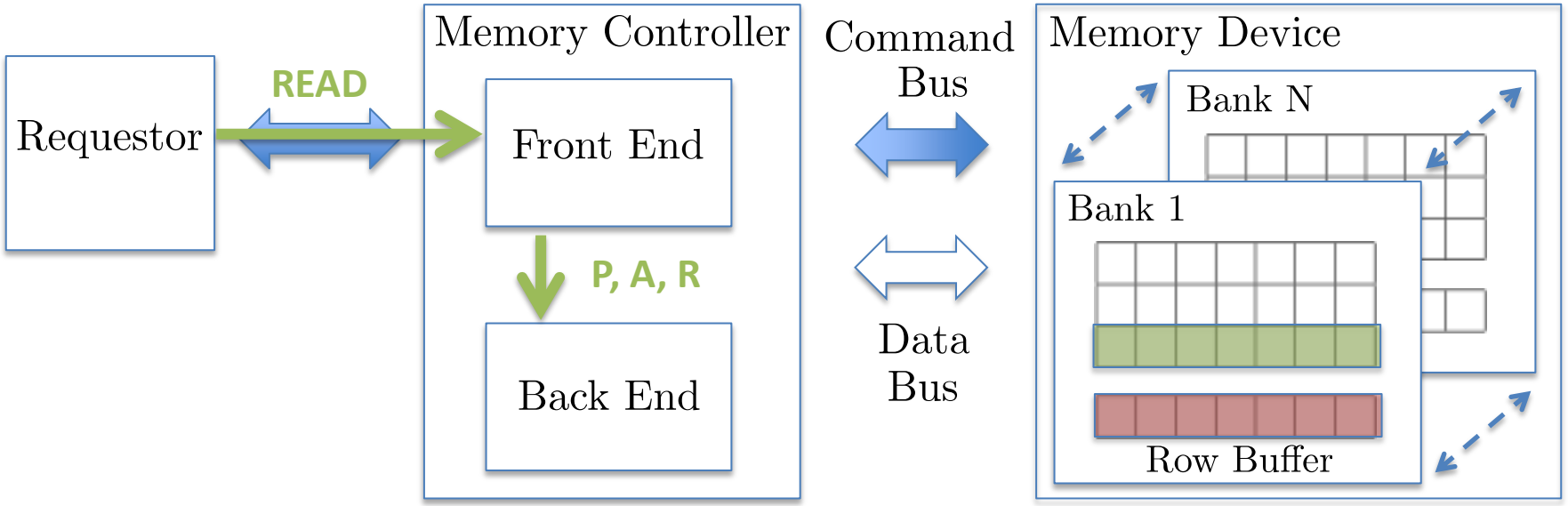
Background


Requestor
Memory Controller
Front End
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer
Front End generates the needed commands,
Back End issues commands on command bus

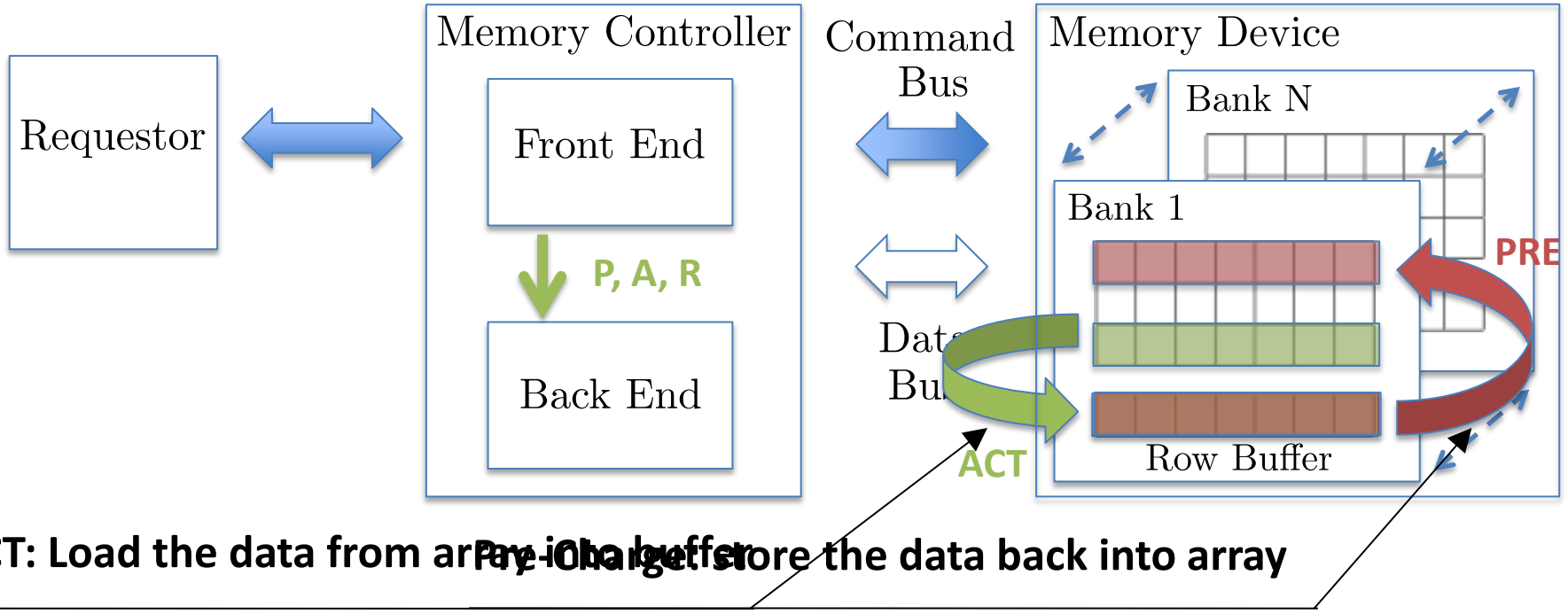
Background


Requestor
READ
Memory Controller
Front End
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer
Targeting Data in this Row
Row Buffer contain data from a different row

Background

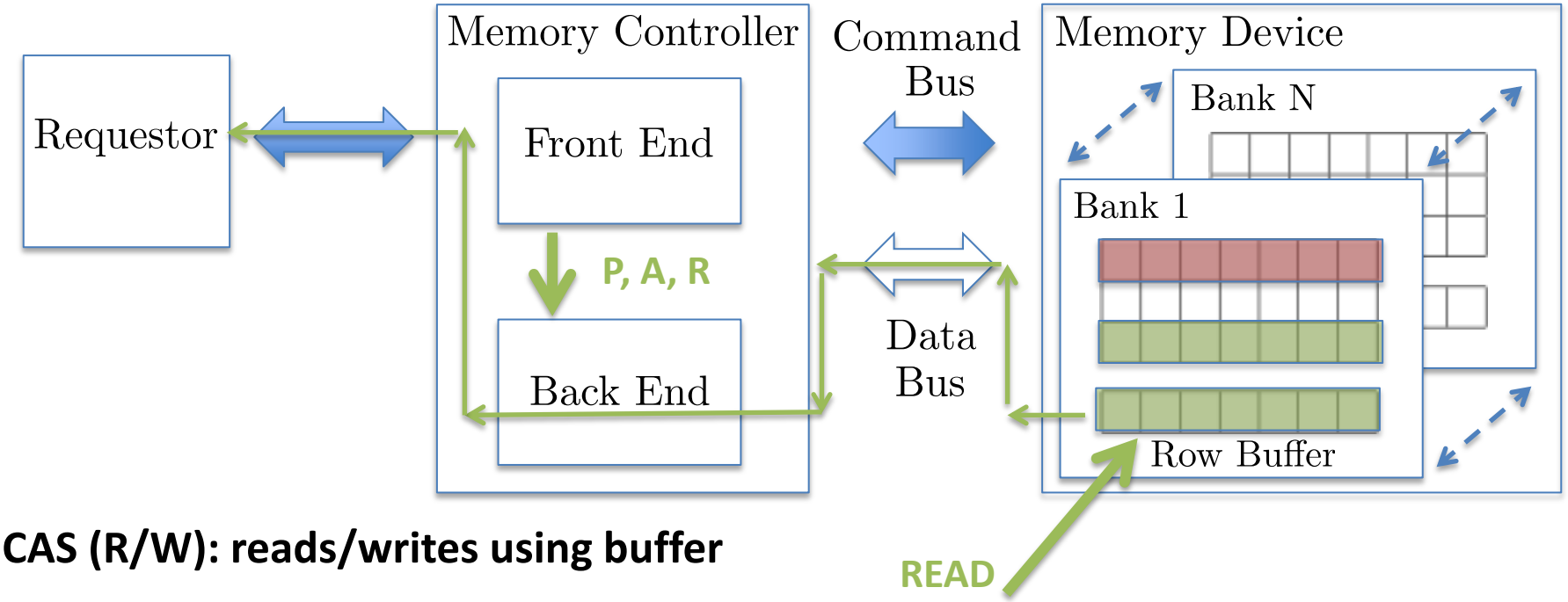

Requestor
READ
Memory Controller
Front End
P, A, R
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer

Background

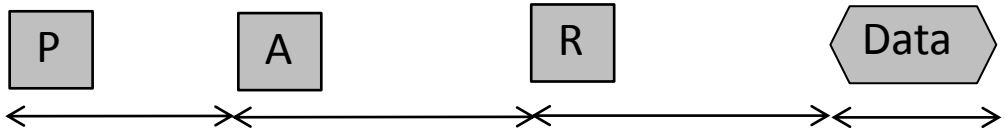

Requestor
Memory Controller
Front End
P, A, R
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
PRE
ACT
Row Buffer
ACT: Load the data from array into buffer
Pre-Charge: store the data back into array


P
A
Pre-charge command issued on command bus
Timing Constraint

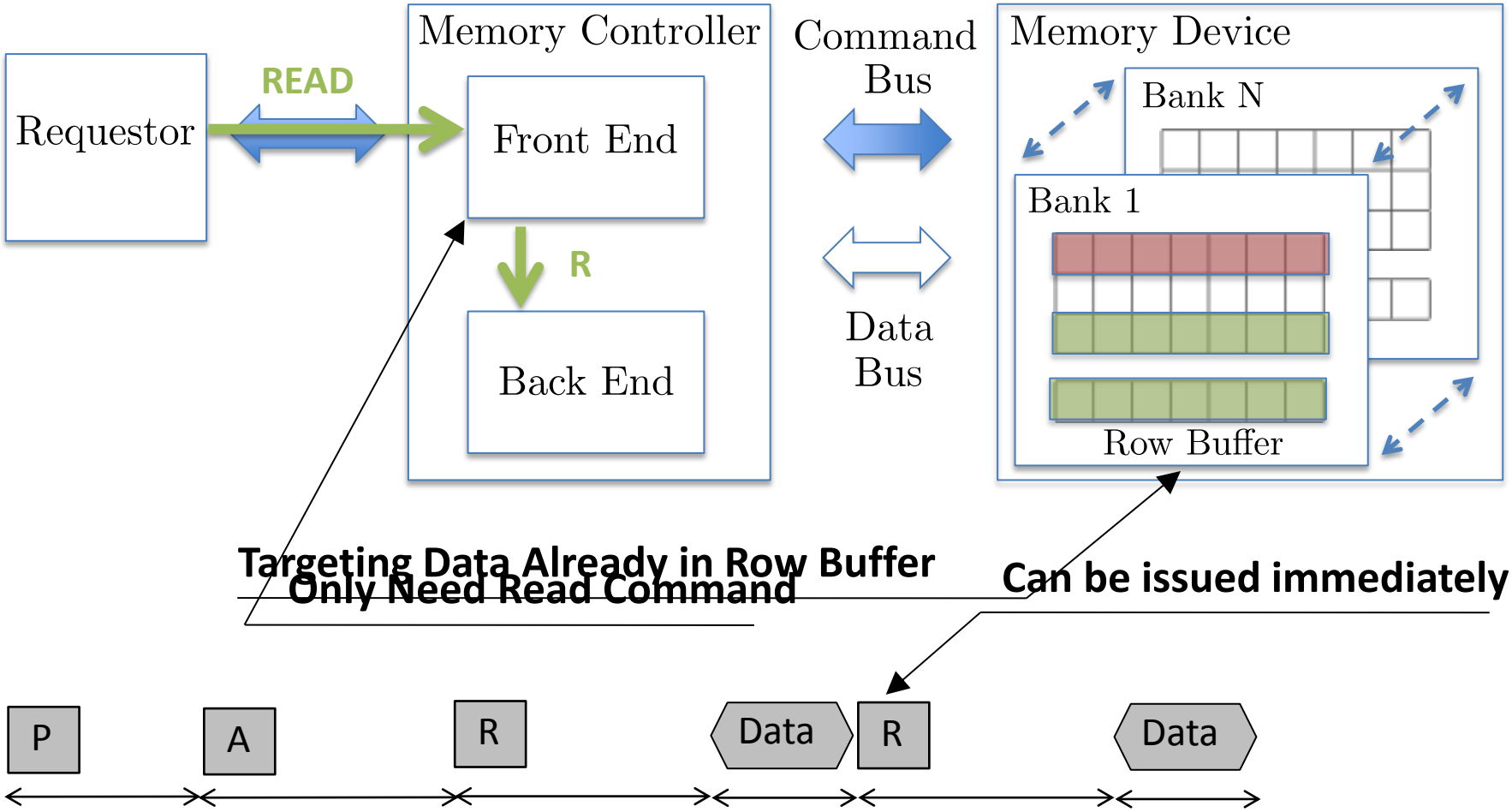
Background



CAS (R/W): reads/writes using buffer

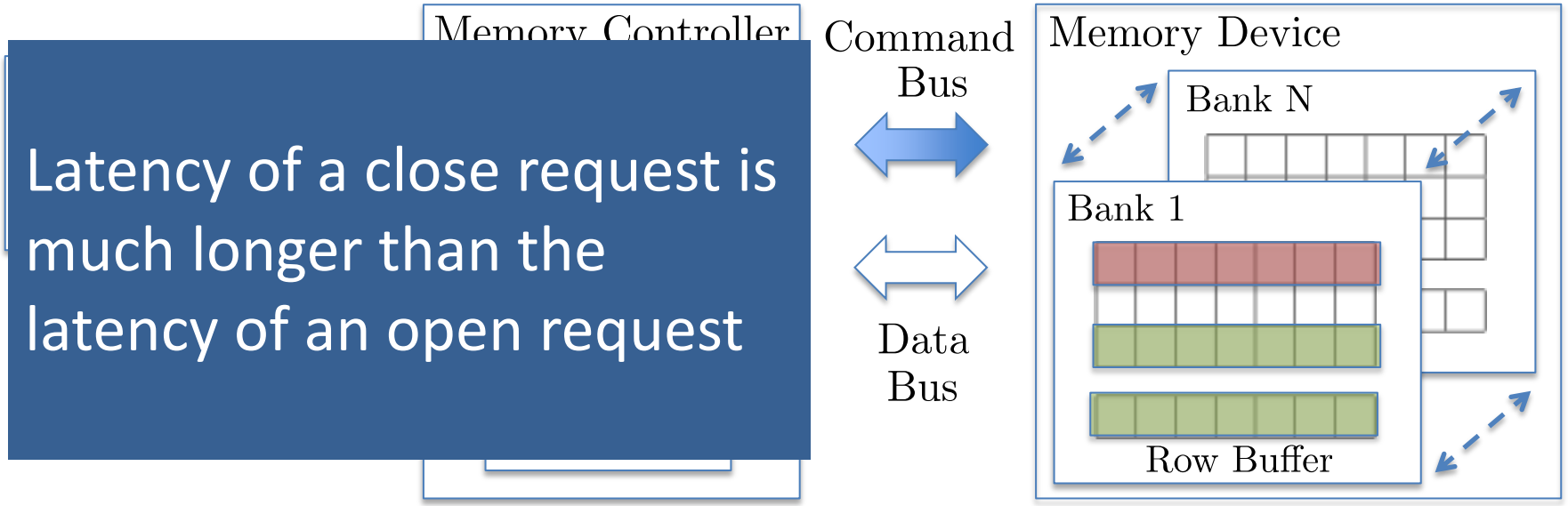


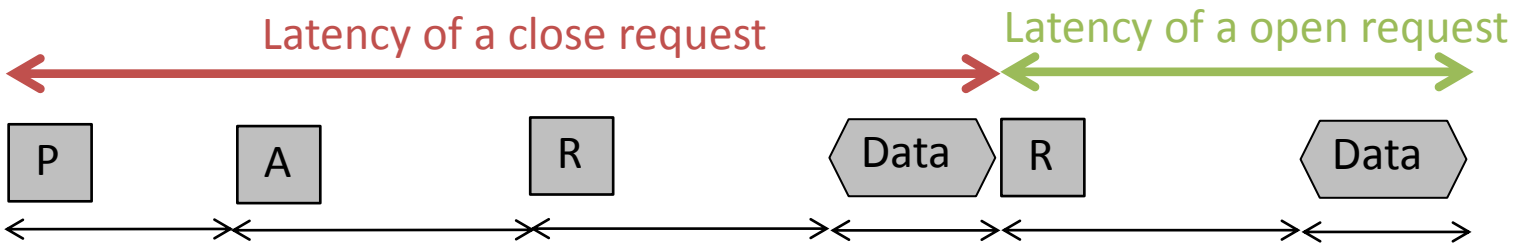
Background


Requestor
READ
Memory Controller
Front End
R
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer
Targeting Data Already in Row Buffer
Only Need Read Command
Can be issued immediately
P
A
R
Data
R
Data

Background

Latency of a close request is much longer than the latency of an open request





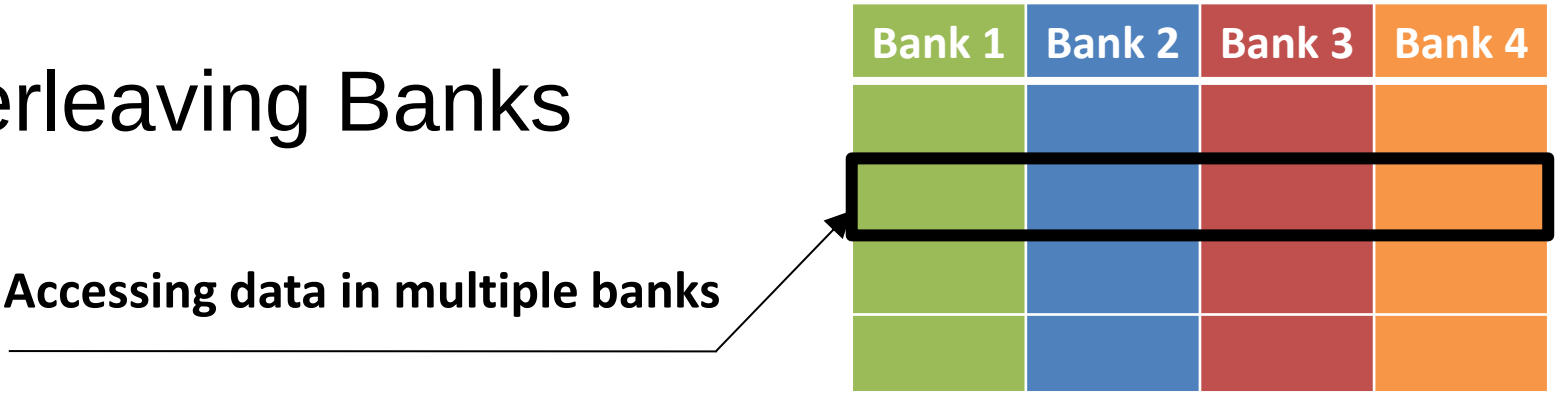
Row Policy

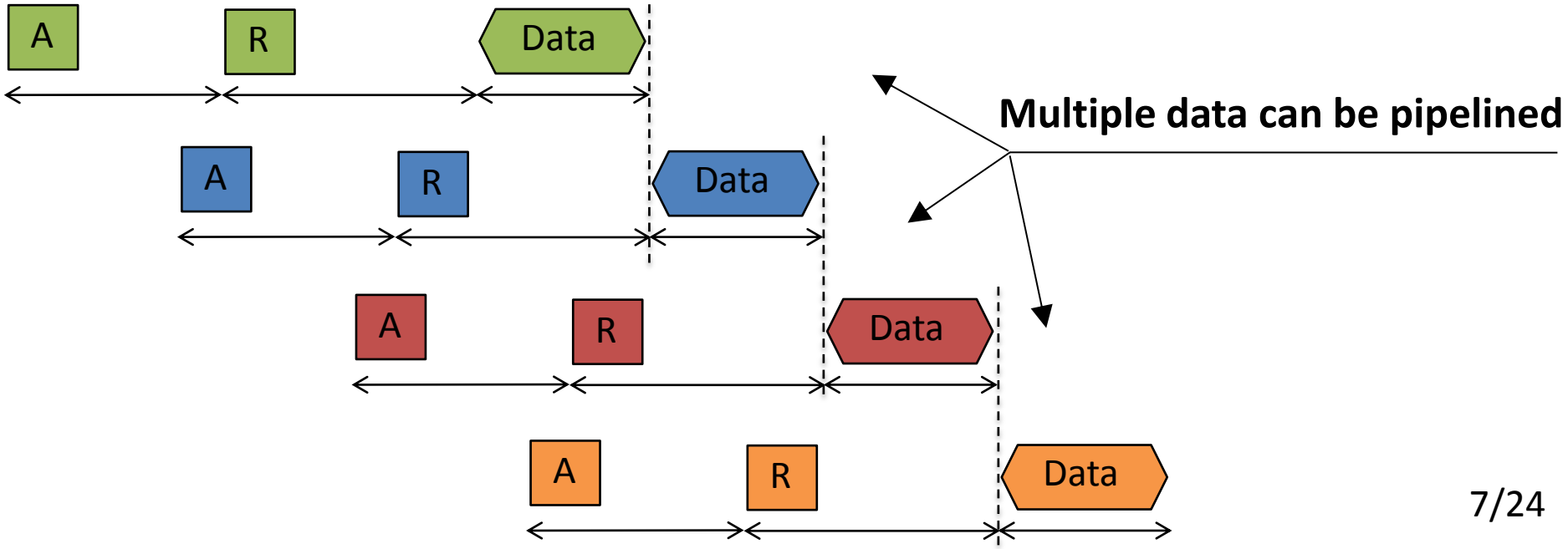
- Close Row Policy:
  - Used by most predictable memory controllers
  - After each access, the row buffer is automatically pre-charged
  - Constant request latency
  - Cannot take advantage of locality (row hits)

- Open Row Policy
  - Used in our approach
  - Keep the row open to exploit locality
  - Different latency for open/close requests

Address Mapping

- Interleaving Banks

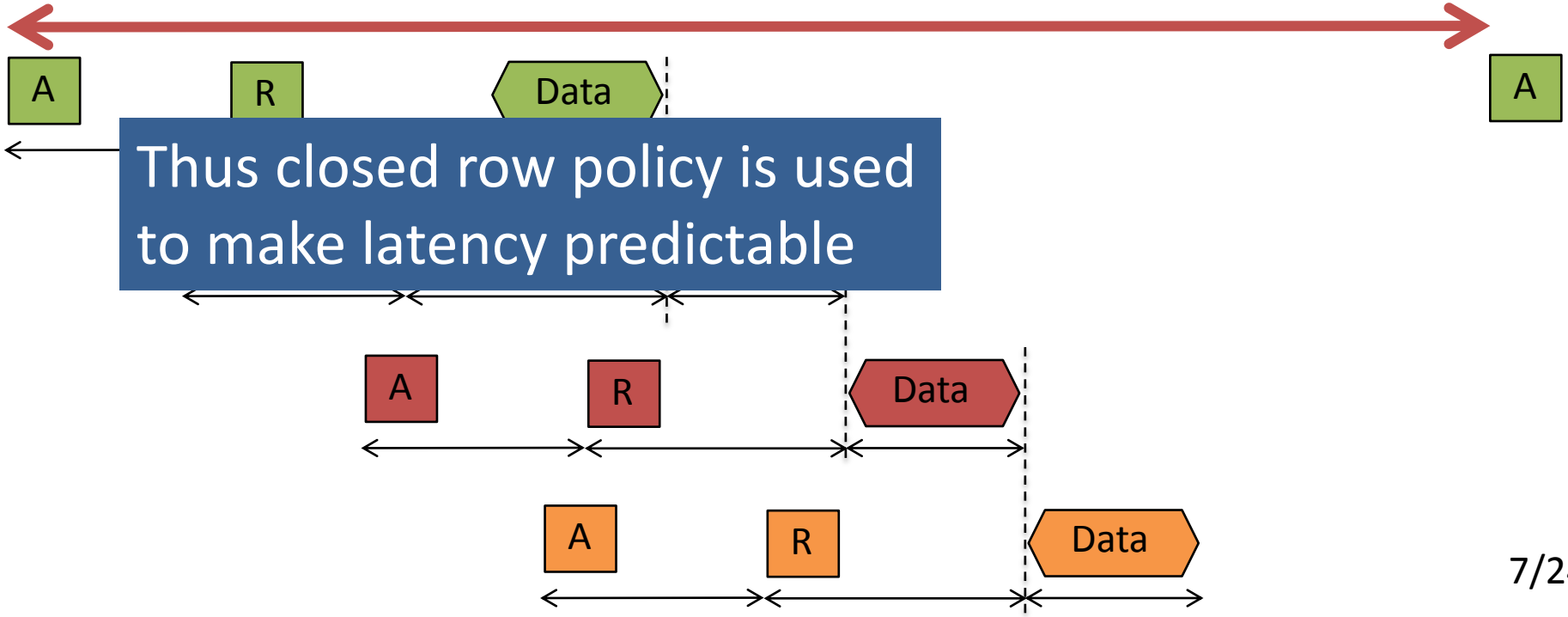




Address Mapping

- Interleaving Banks

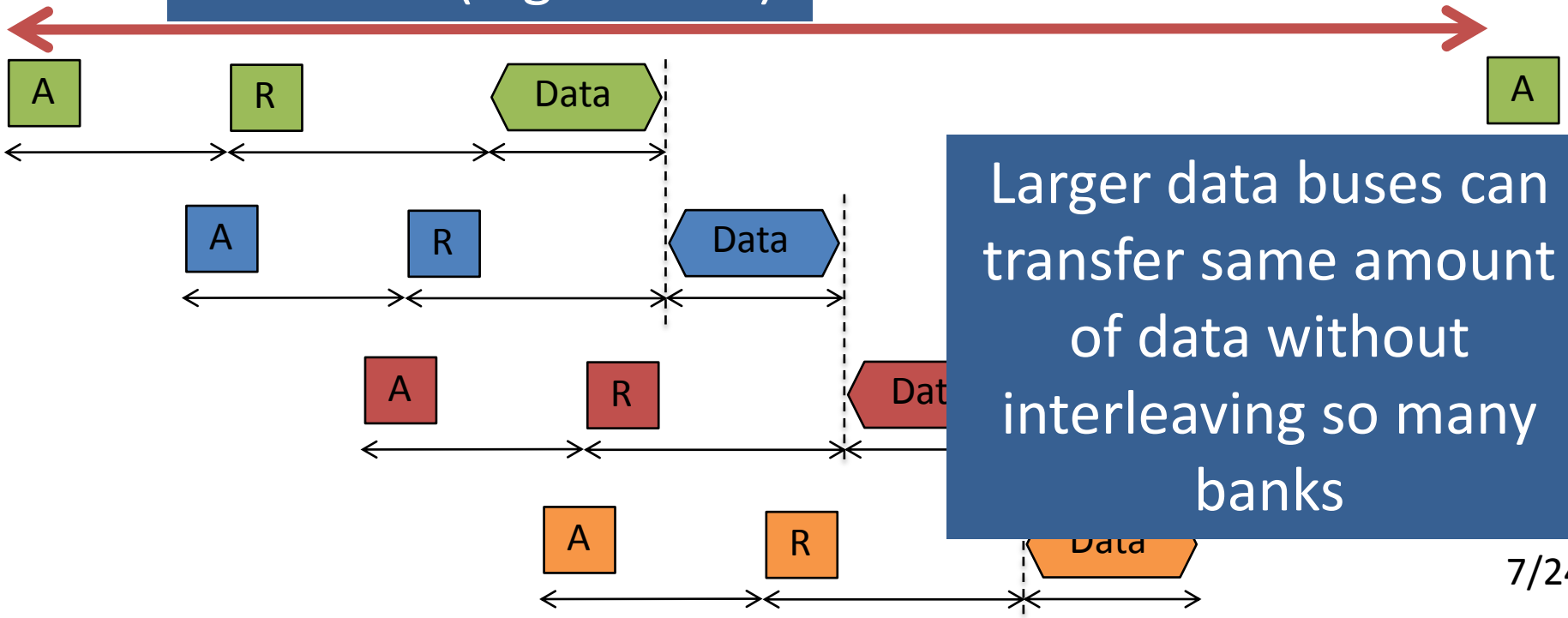
Problem: requestors can close each other's row buffer since they can access all banks



Address Mapping

- Interleaving Banks

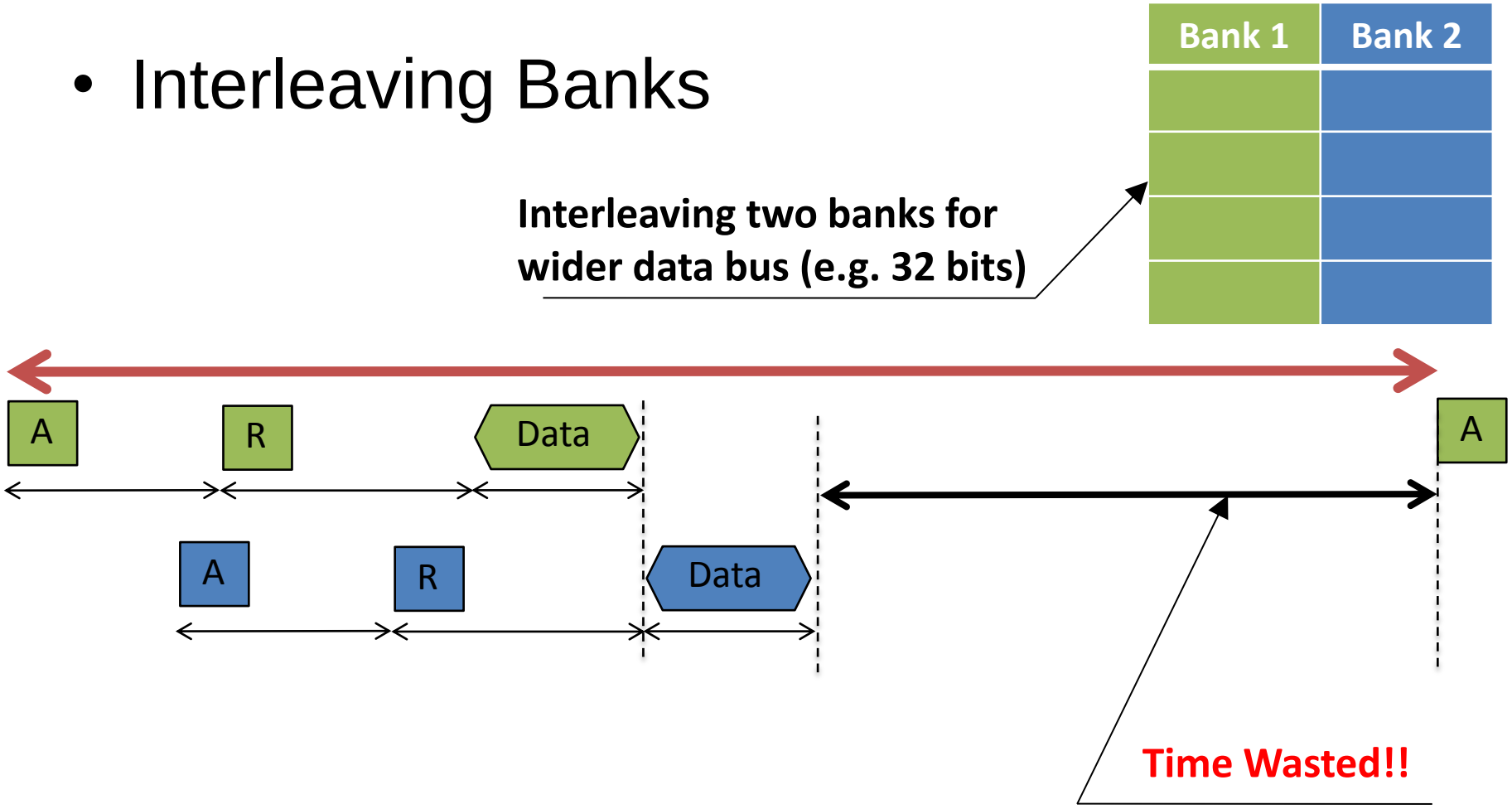
This is good for system with small DRAM data bus width (e.g. 16 bits)



Larger data buses can transfer same amount of data without interleaving so many banks

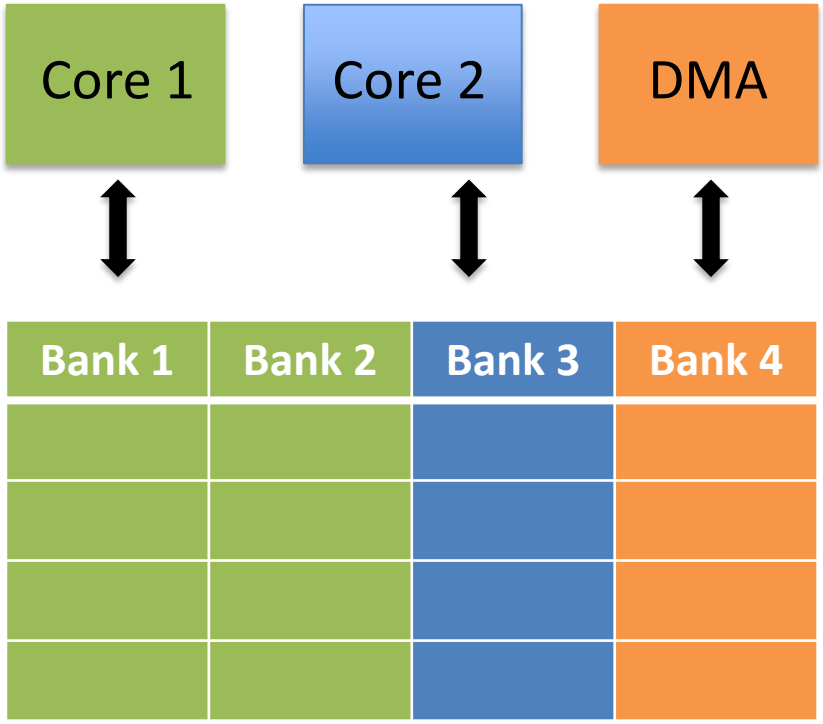
Address Mapping

- Interleaving Banks



Address Mapping

- Private Banks



- Used in our approach
- Partition banks among requestors. How:
  - Hardware if memory controller supports
  - By compiler
  - In OS, using virtual memory
- No row conflicts

Related Work

- AMC
  - Int
  - Cl
- Cons
  - Int
  - Le
- PRE
  - Pri
  - Cl

In RTSS'13 [5], we presented the first predictable controller with:

- Open row policy
- Private Banks

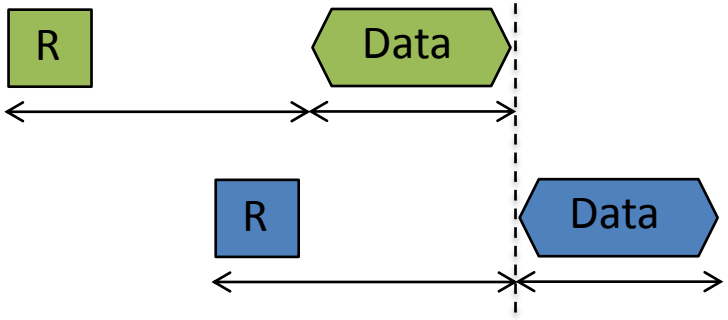
Improved latency but…

Problem

DRAM is very inefficient when switching between write and read

Write to Read Switching

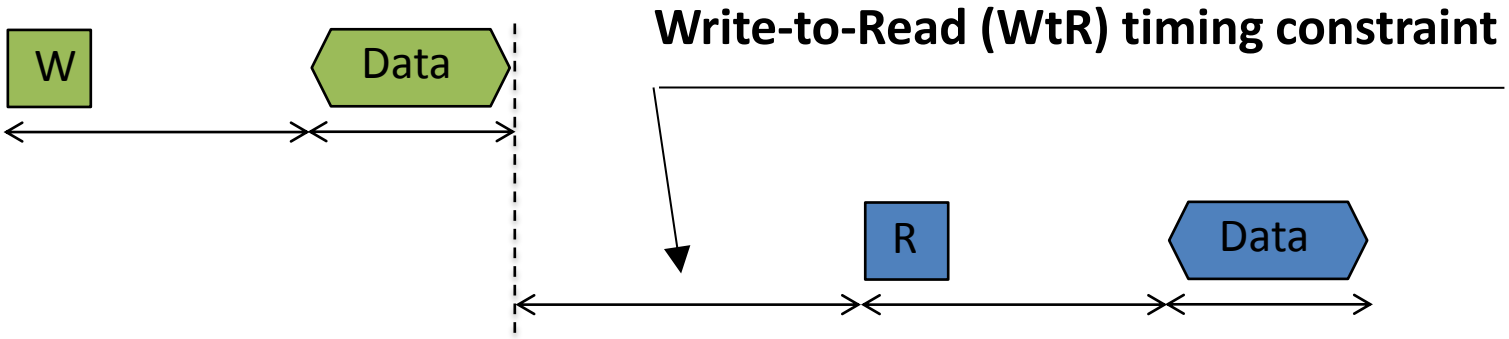
- Transactions of the same type can be pipelined…



Huge Latency Penalty!

We need a solution

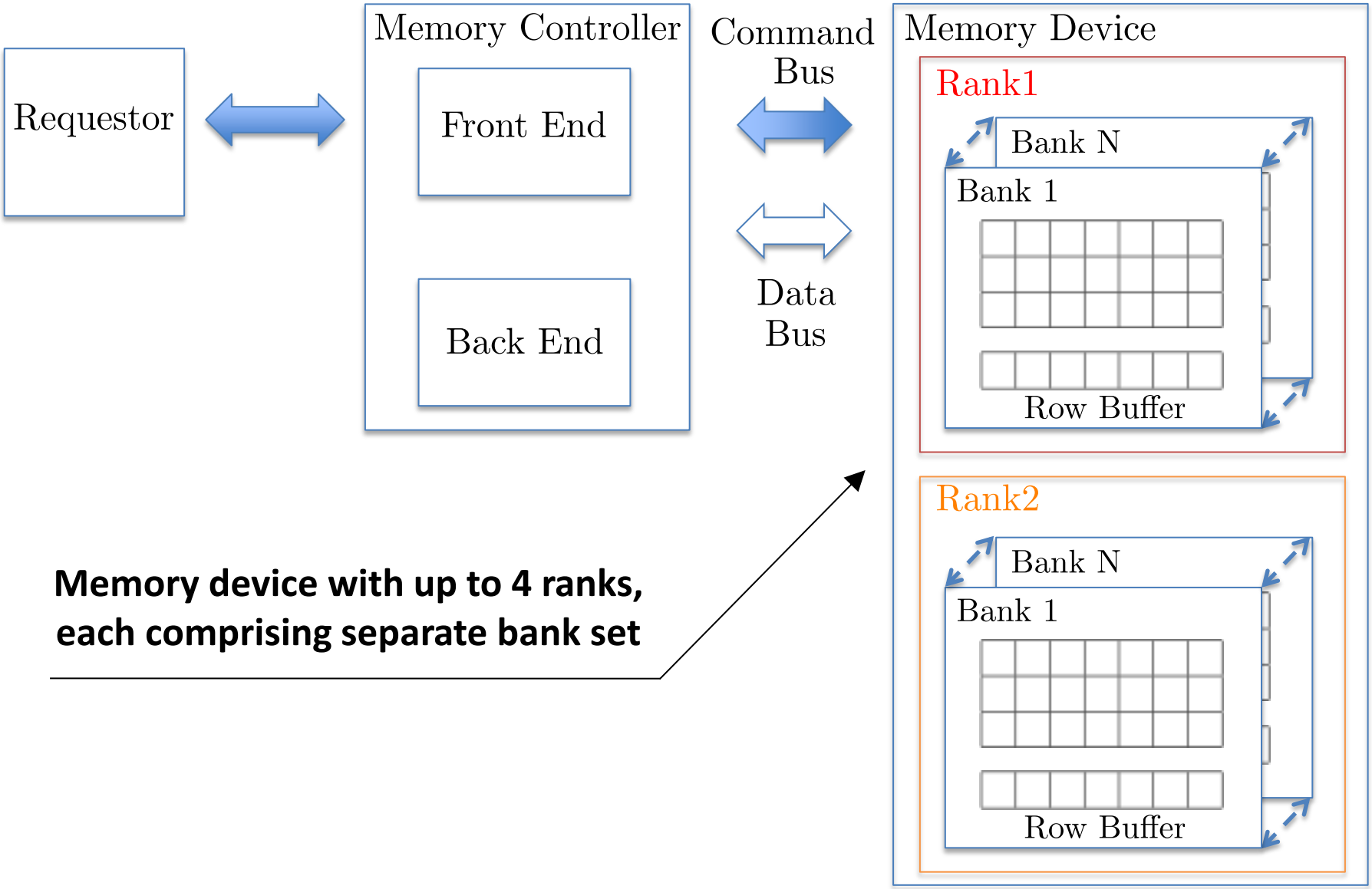
- … but a read after a write cannot.



Outline

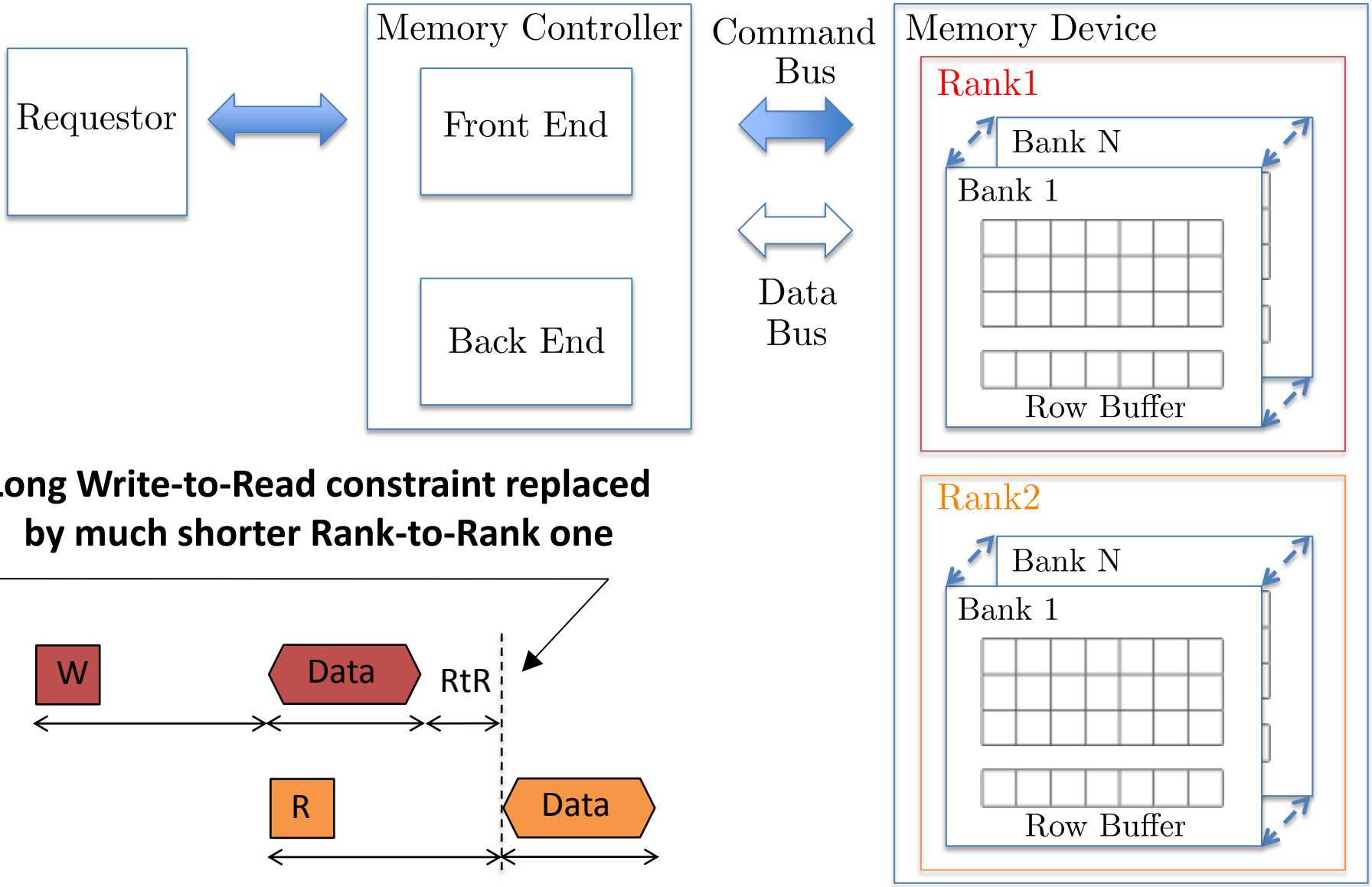
1. Background & Related Work
2. Rank-Switching Mechanism
3. Worst Case Latency Analysis
1. Memory Controller Model
2. Results & Conclusion

Multi-Rank Device



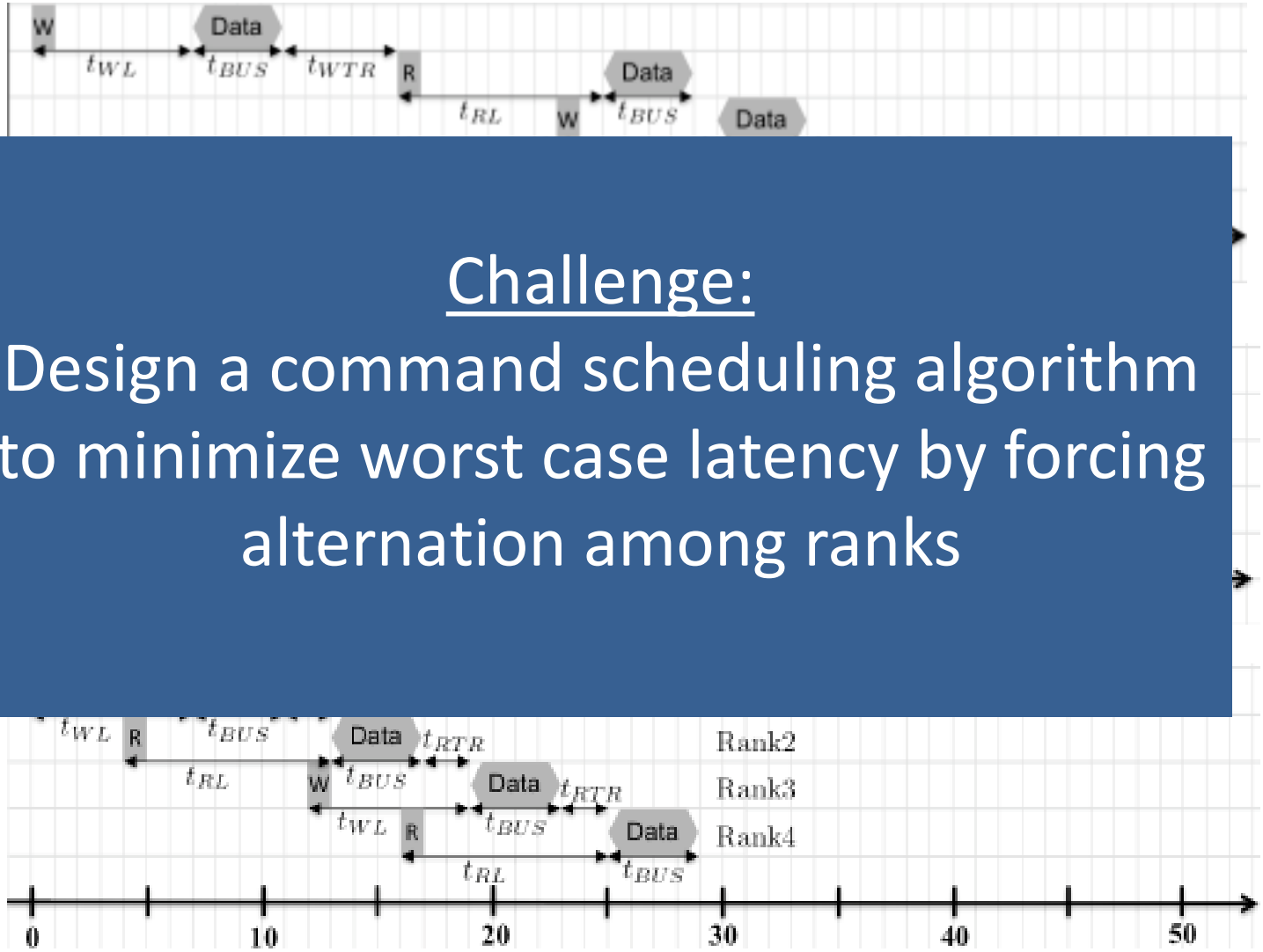
Memory device with up to 4 ranks, each comprising separate bank set

Multi-Rank Device



Long Write-to-Read constraint replaced by much shorter Rank-to-Rank one

Example: Write-Read-Write-Read



1 Rank:
Latency 52 cycles

2 Ranks:
Latency 35 cycles

4 Ranks:
Latency 29 cycles

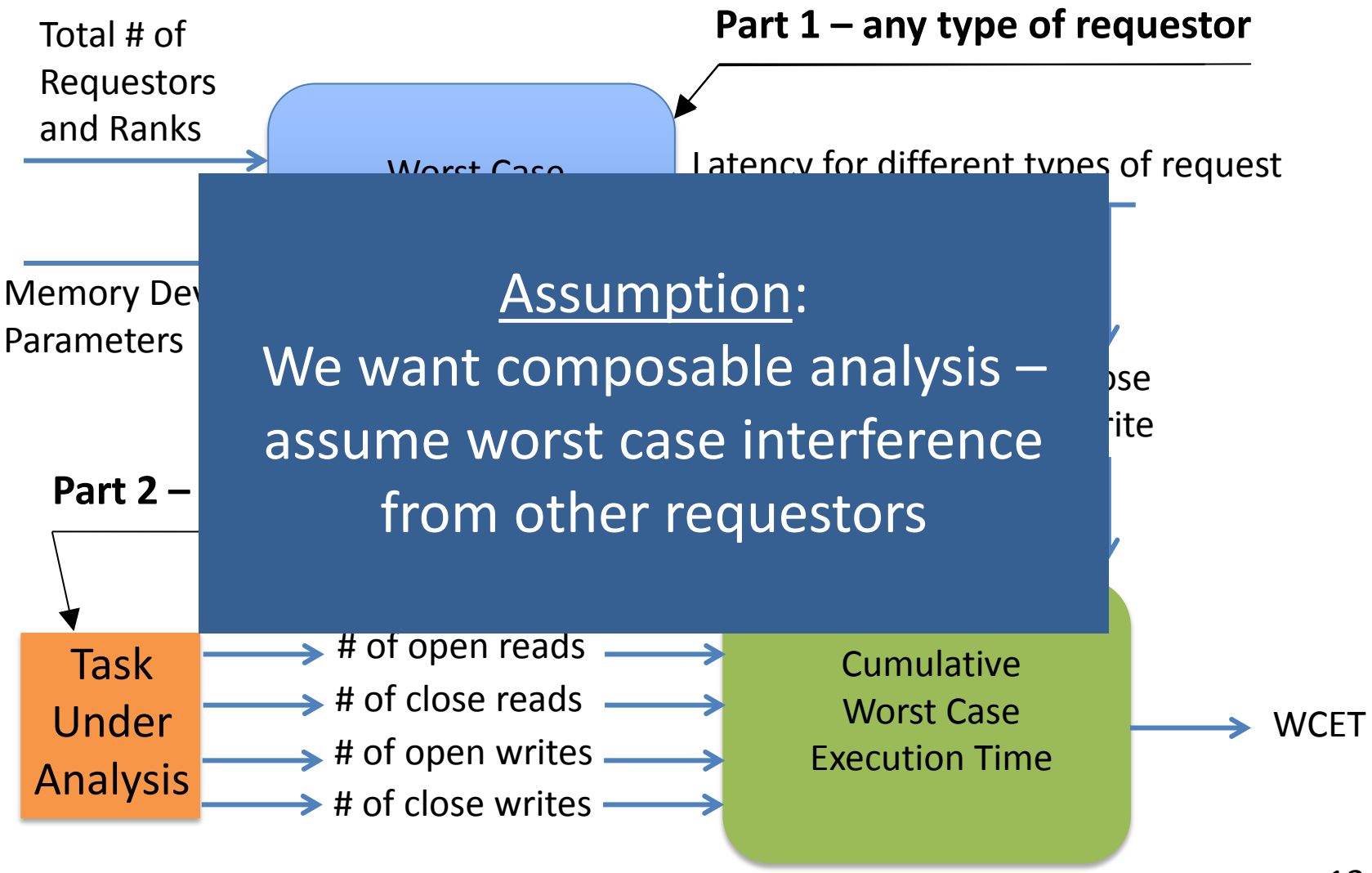
Challenge:

Design a command scheduling algorithm to minimize worst case latency by forcing alternation among ranks

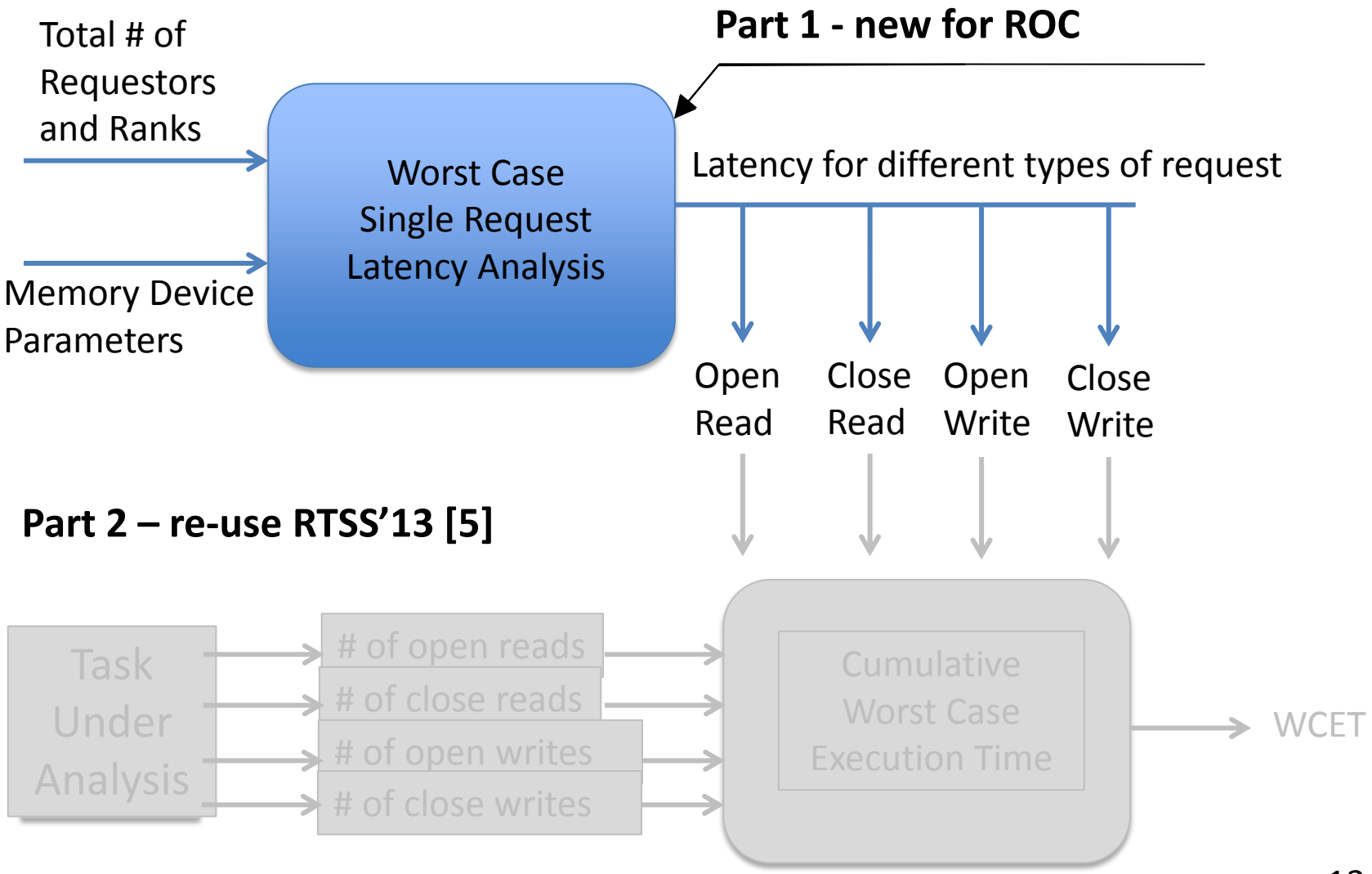
Outline

1. Background & Related Work
2. Rank-Switching Mechanism
3. Worst Case Latency Analysis
4. Memory Controller Model
5. Results & Conclusion

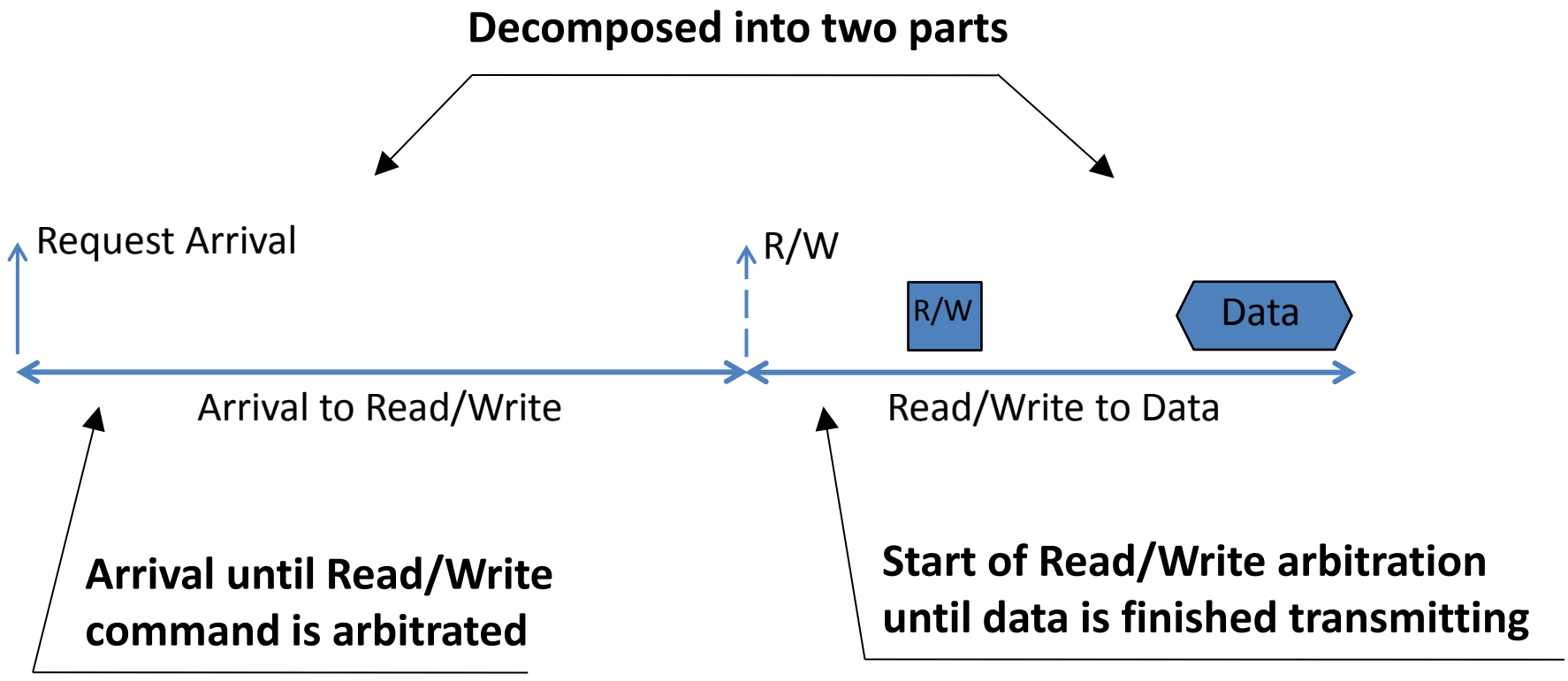
Worst Case Analysis



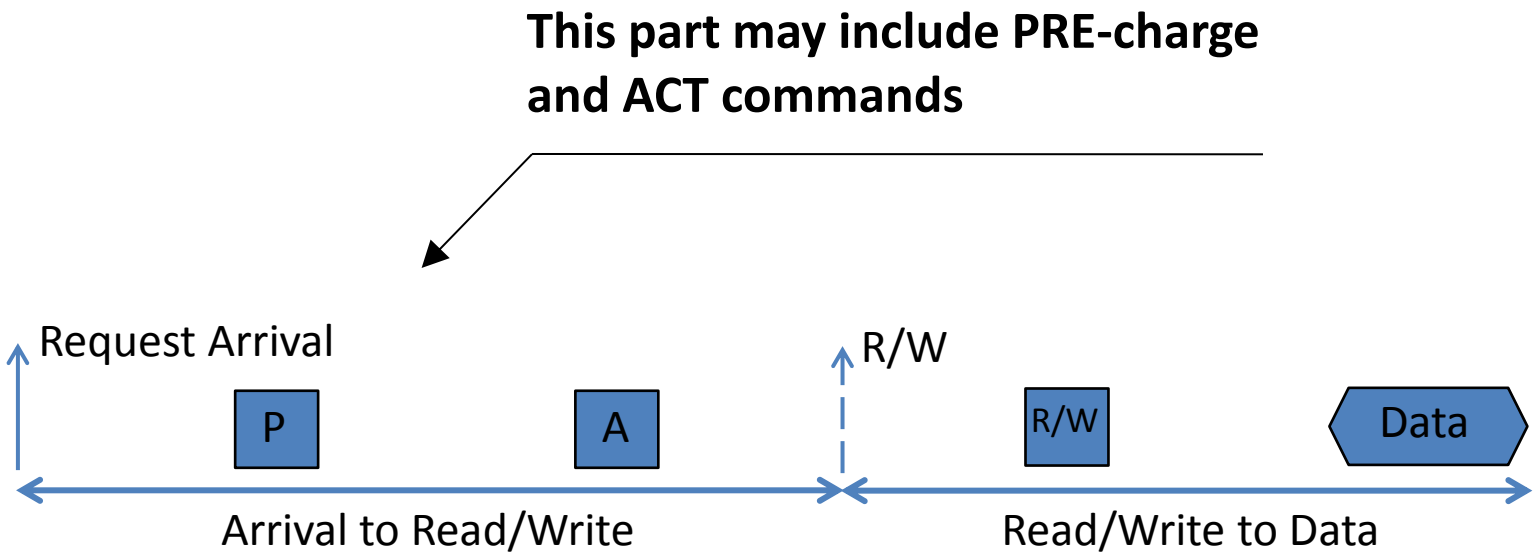
Worst Case Analysis



Single Request Latency



Single Request Latency



Our arbitration mechanism similarly distinguishes between PRE/ACT and CAS (R/W) commands

Outline

1. Background & Related Work

2. Rank-Switching Mechanism

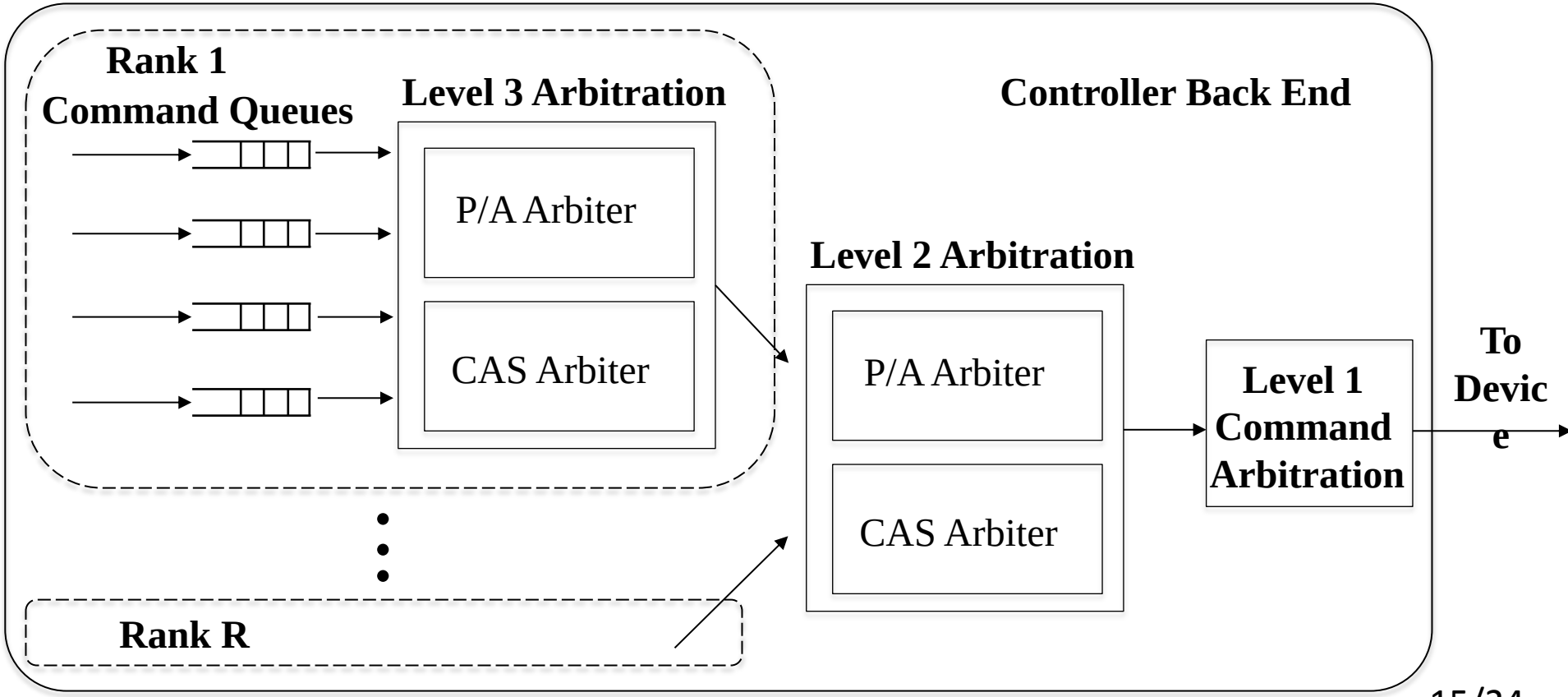
1. Worst Case Latency Analysis

2. Memory Controller Model

3. Results & Conclusion

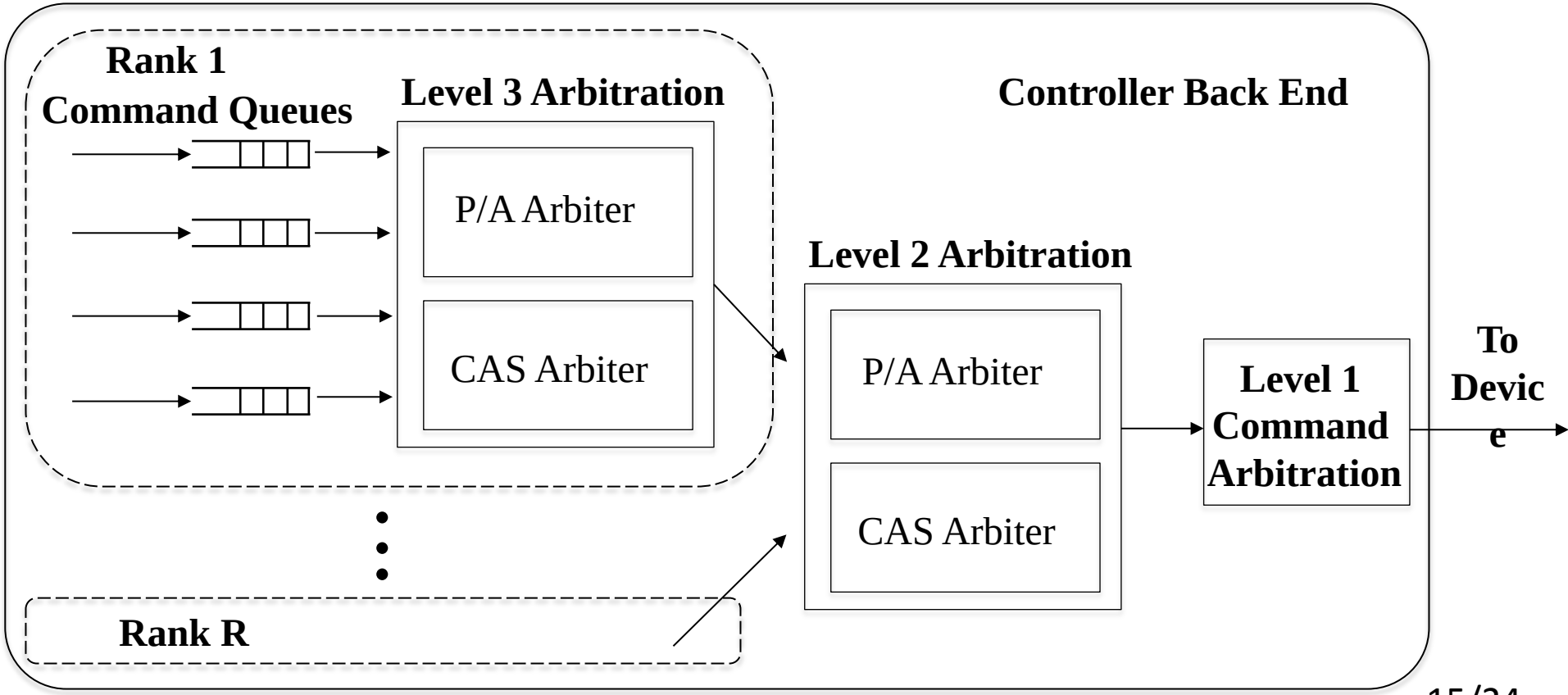
Back End Model

- Front End adds constant delay – focus on Back End
- Three levels arbitration



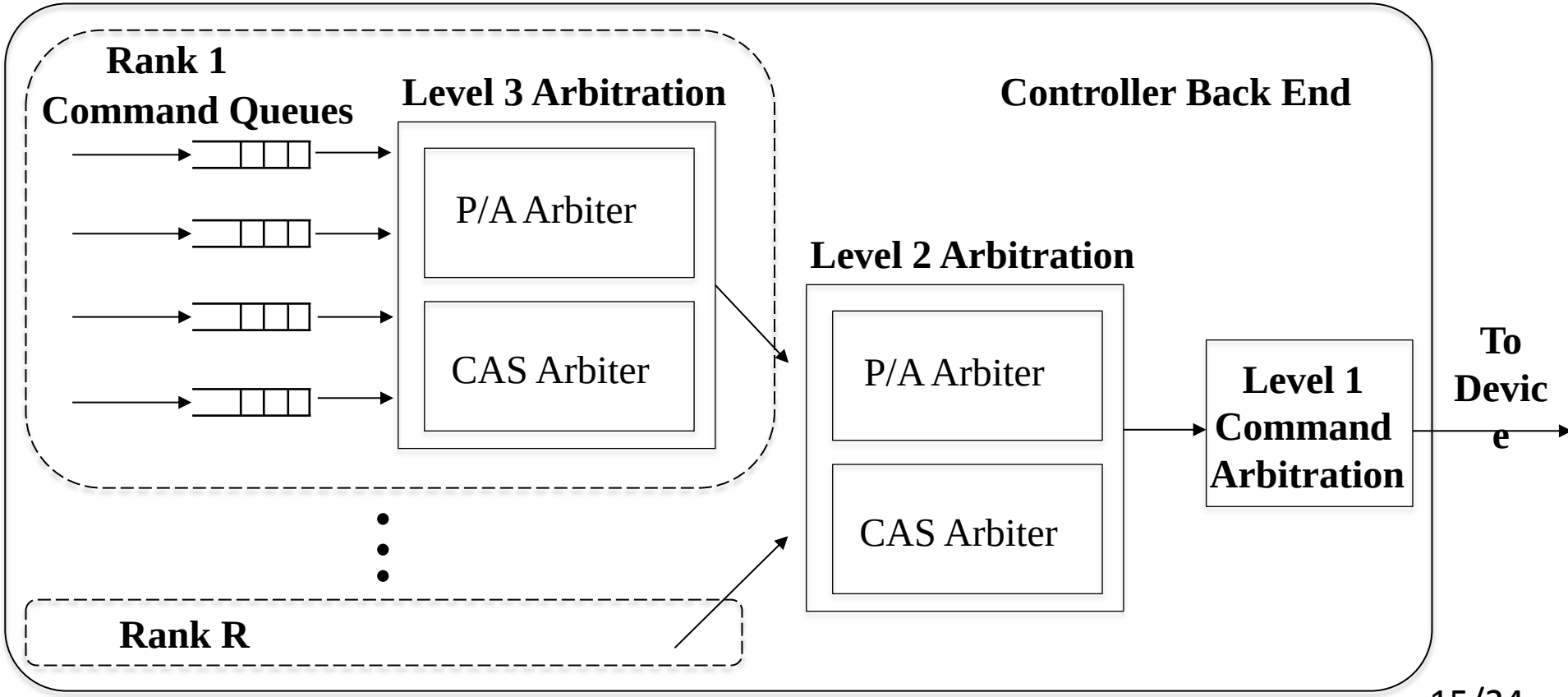
Back End Model

- L1: CAS (R/W) commands have higher priority than PRE/ACT commands – priority to data bus contention



Back End Model

- L2 alternates among ranks
- L3 alternates among requestors within a rank



Back End Model

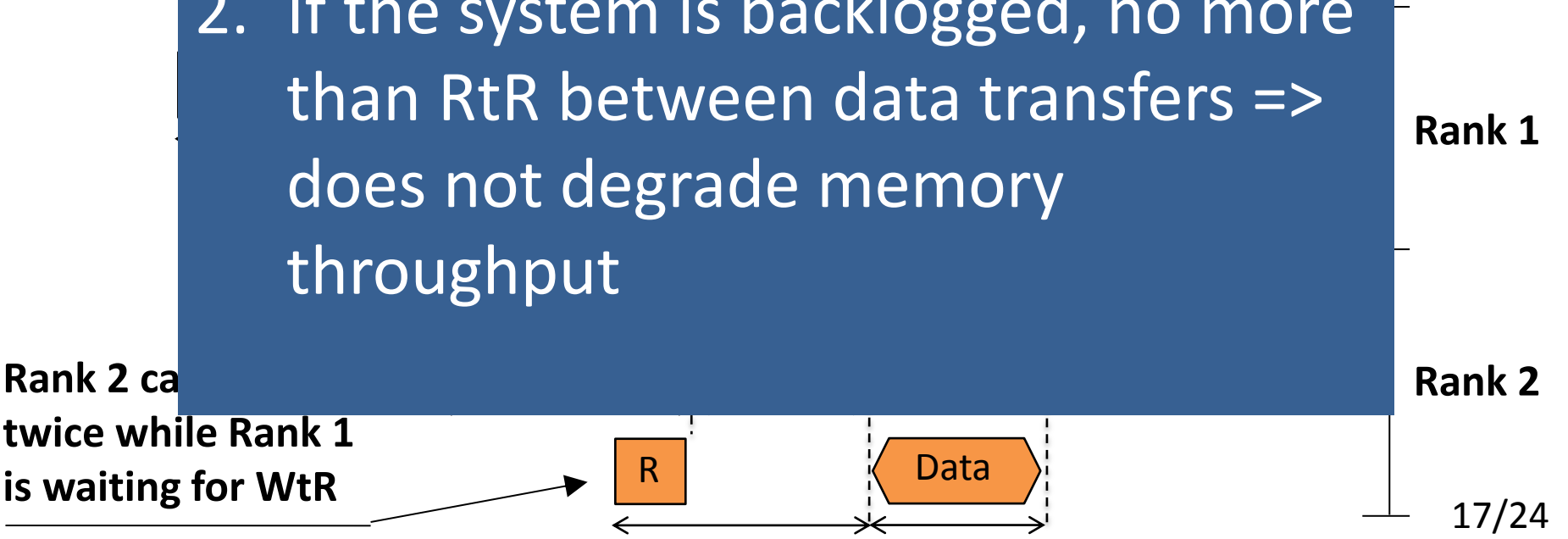
- Let:
  - R : number of ranks
  - M_r : number of requestors for rank r
- Then given the alternation in L2/L3, the latency of each command is a function of $R \cdot M_r$
  - Isolation property: the latency of a requestor does not depend on the # of requestors or scheduling policy used in other ranks
  - We can dedicate some ranks to hard real-time requestors, and others to soft real-time requestors
  - Optimize hard requestors for latency, soft requestors for bandwidth
- Full details for hard requestors in the paper…

CAS Rank Switching Rule

- Key id
  - S … the d… nstraint
  - V… straint), re…

We prove:

1. Reordering does not increase worst-case latency if R = 4
2. If the system is backlogged, no more than RtR between data transfers => does not degrade memory throughput



Outline

1. Background & Related Work
2. Rank-Switching Mechanism
3. Memory Controller Model
4. Worst Case Latency Analysis
5. Results & Conclusion

Results

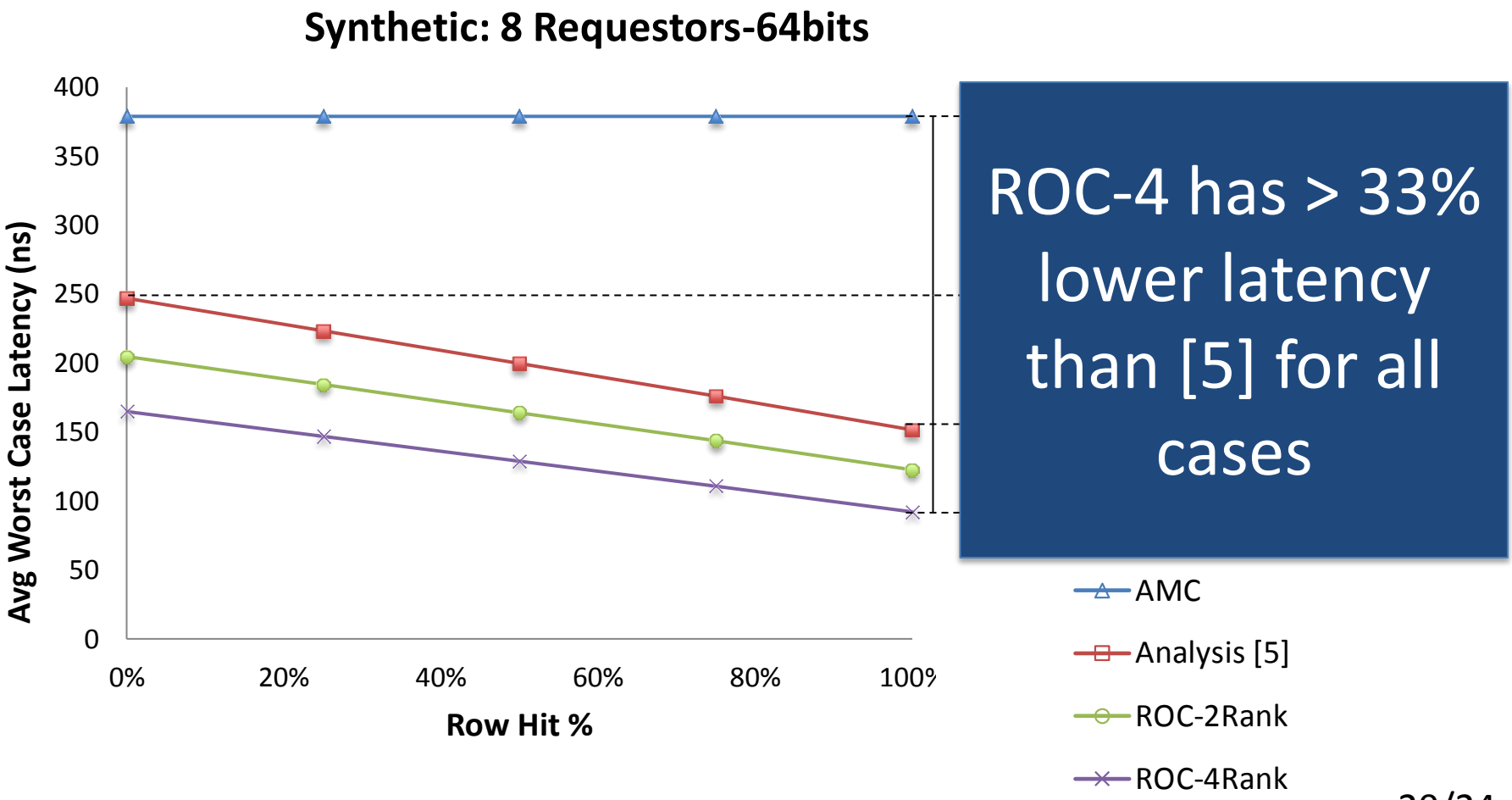
- Comparison against Analyzable Memory Controller (AMC) [1]
  - Fair arbitration (Round Robin) similar to our approach
  - Focus on WCET guarantees for hard real-time tasks
- Synthetic Benchmarks
  - Used to show how worst case latency bound varies as parameters are changed
- CHStone Benchmarks
  - Memory traces are obtained from gem5 simulator and used as inputs to both analysis and simulators
  - Core under analysis is in-order
  - Interfering requestors are out-of-order running *lbm*

Results

- DDR3-1333H memory device
  - 64 and 32 bits data bus width
- Simulations
  - Python simulators for our RTSS'13 work [5] and AMC [1]
- ROC Implementation
  - Three stages pipelined implementation in Verilog RTL
  - Synthesizes to Xilinx FPGA at 340Mhz (original soft memory controller: 400Mhz)
  - ASIC implementation could likely be significantly faster…
- Code available at http://ece.uwaterloo.ca/~rpellizz/roc.php

Results

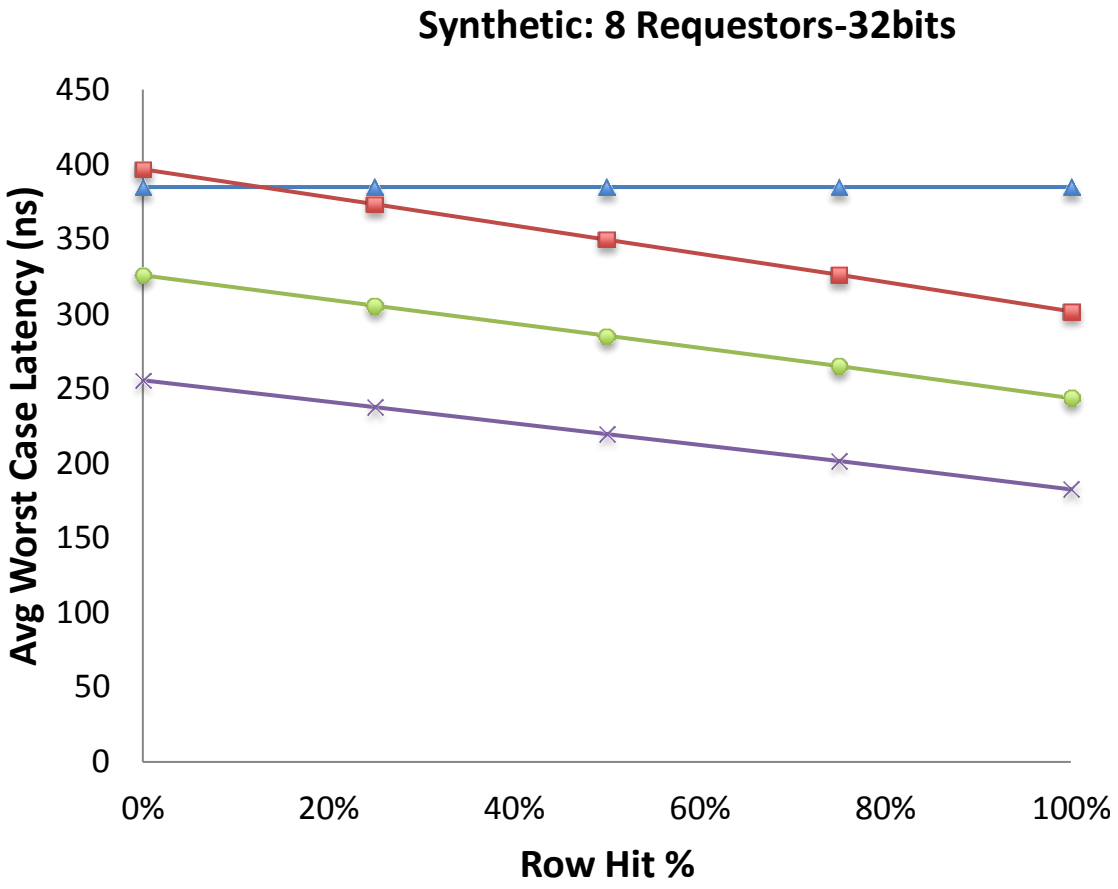
- Synthetic Benchmarks, 20% write



ROC-4 has > 33% lower latency than [5] for all cases

Results

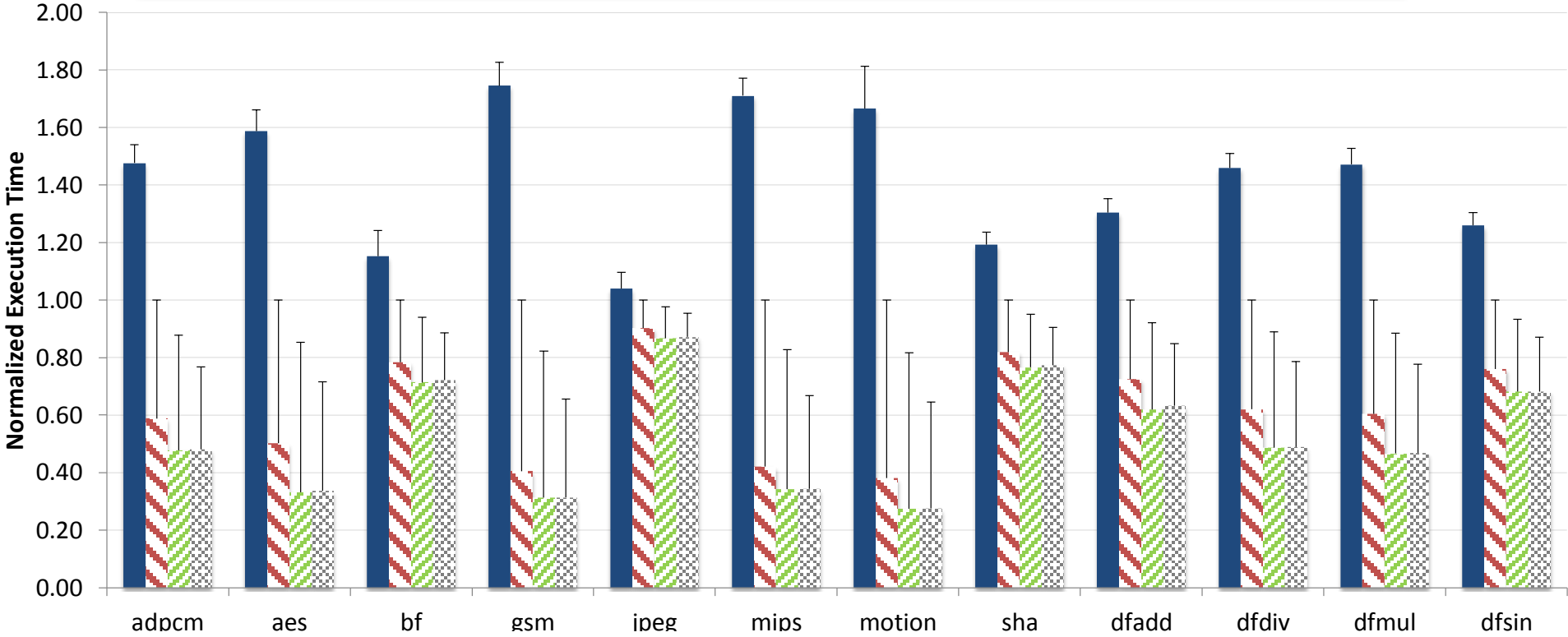
- Synthetic Benchmarks, 20% write



AMC can parallelize transactions over 2 banks – no advantage from private bank parallelism

Results

- ROC-4 has between 5 and 35% lower WCET than [5]



Conclusions

- Architectural optimizations designed for general purpose systems do not necessarily work for time-predictable systems
- We need to design the architecture around the concept of guaranteed worst case latency
- We introduced a new DRAM optimization targeted at reducing worst case latency: rank-switching
- The implemented ROC memory controller significantly reduces latency for hard requestors and guarantees strong isolation between hard/soft requestors

Future Work

- Implementation:
  - Support for shared data
  - Soft requestor optimizations
  - Improved RTL code

- Extended comparison with other real-time controllers

References

[1] M. Paolieri, E. Quinones, F. Cazorla, and M. Valero, "An Analyzable Memory Controller for Hard Real-Time CMPs," *Embedded Systems Letters, IEE*

[2] B. Akes ator: a predictable SDRAM me . 251–256.

[3] S. Goos rvative Open- page Policy for M " in *DATE*, 2013.

[4] J. Reine "Pret dram controller: mporal isolation," in CODES+ISSS, 2011, pp. 99–108.

[5] Z. P. Wu, Y. Krish, and R. Pellizzoni, "Worst Case Analysis of DRAM Latency in Multi-Requestor Systems"

Thank you!

Questions?