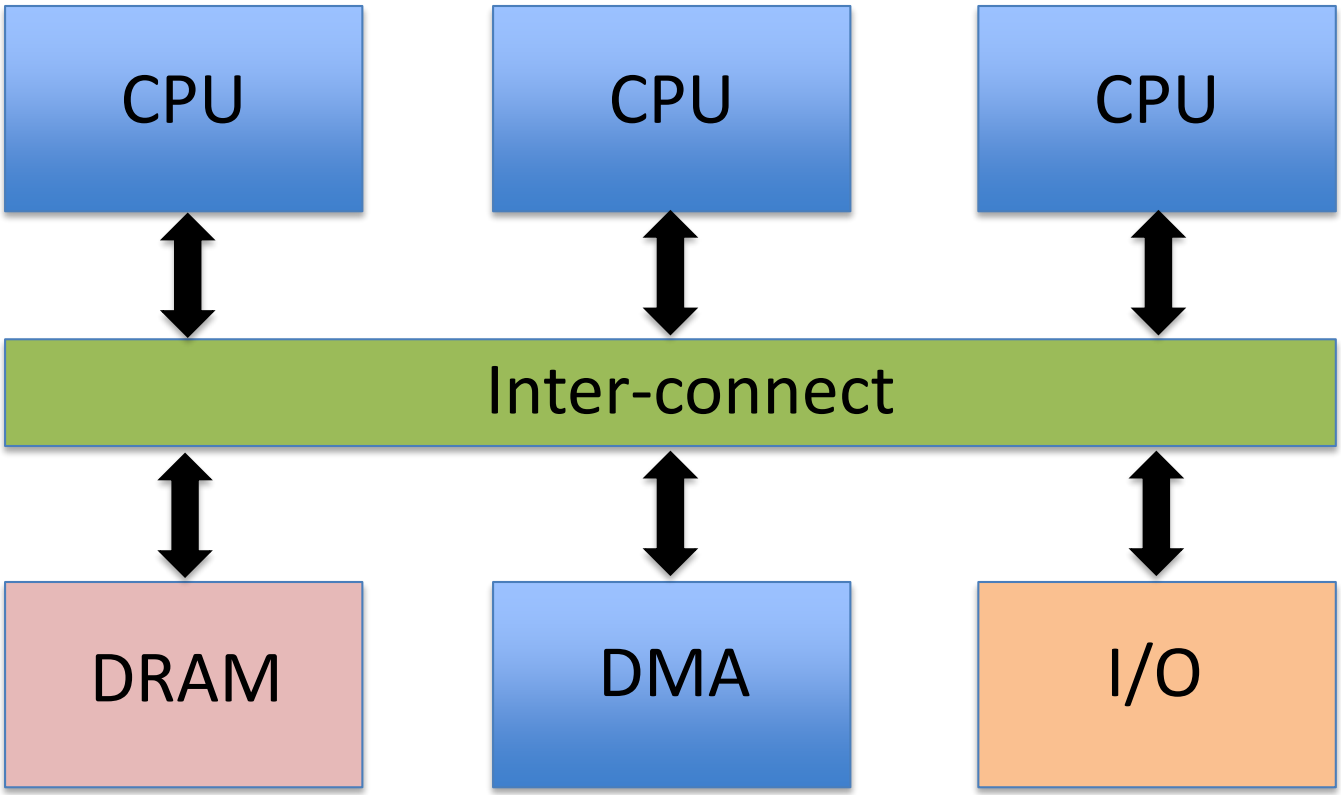
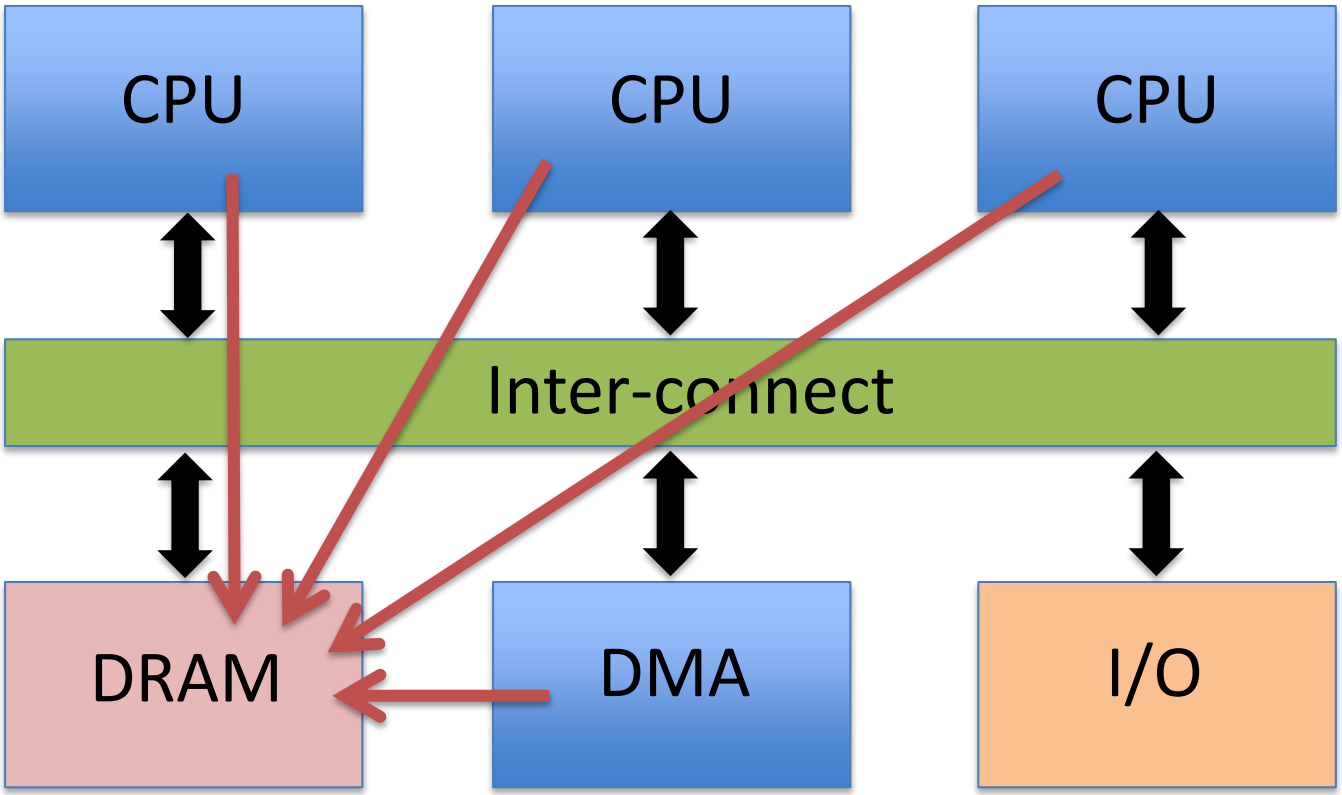
Worst Case Analysis of DRAM Latency in Multi-Requestor Systems

Zheng Pei Wu
Yogen Krish
Rodolfo Pellizzoni

Multi-Requestor Systems

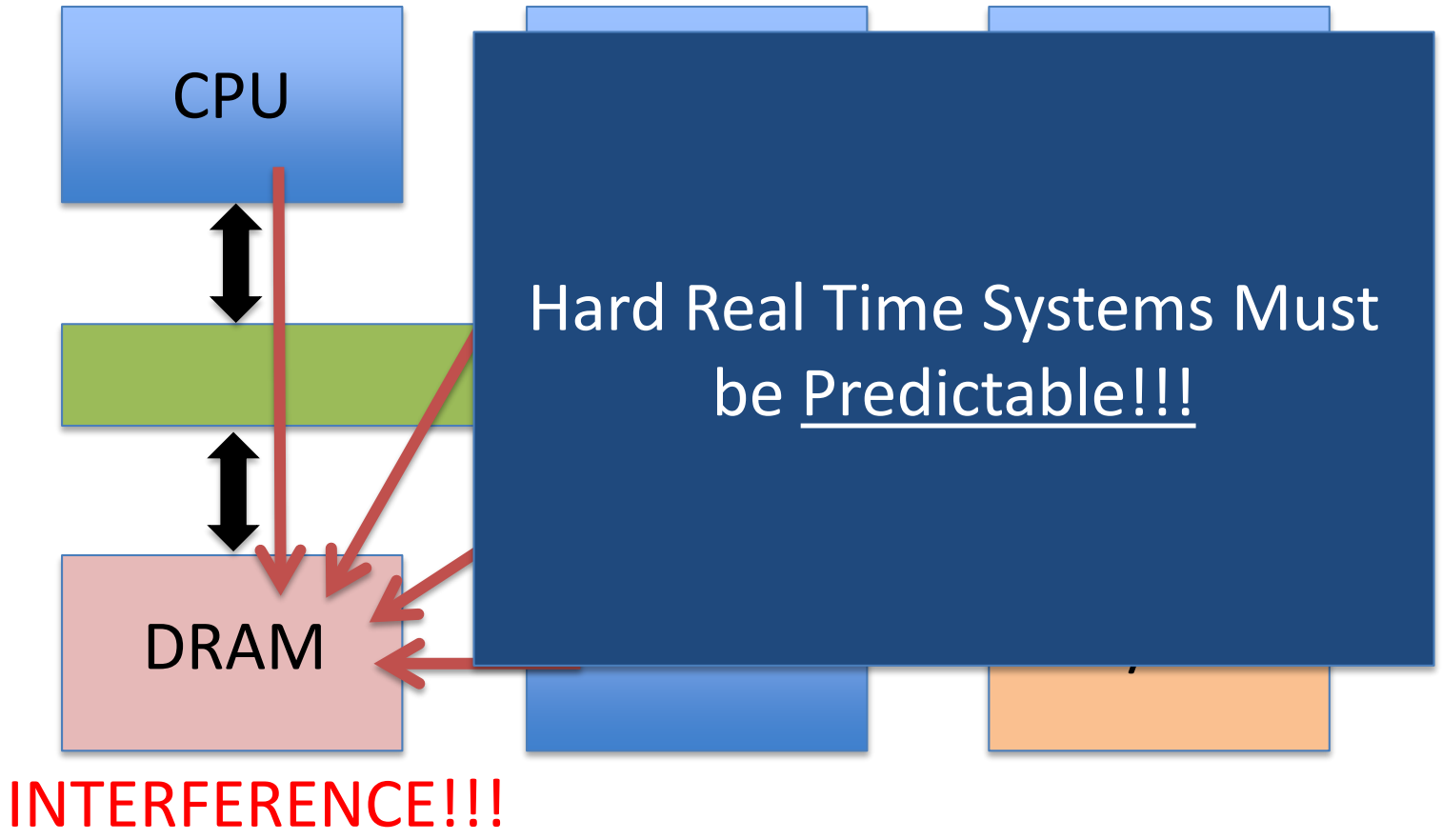


Multi-Requestor Systems



INTERFERENCE!!!

Multi-Requestor Systems



Multi-Requestor Systems

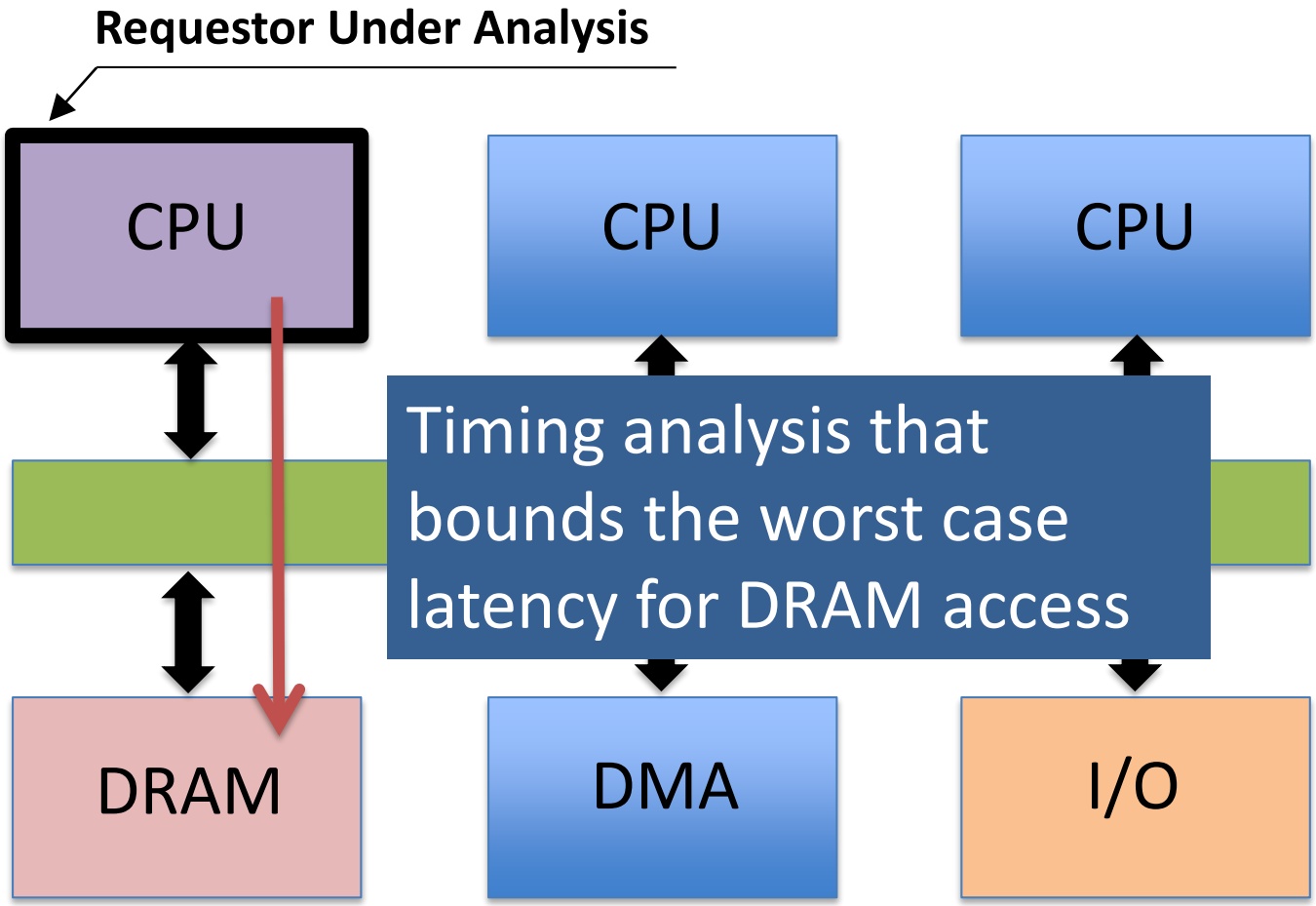
- Schedulability Analysis: needs WCET as input
- WCET depends on hardware platform
- WCET: needs Latency to access shared resource (e.g. cache, DRAM)
- Existing approaches can bound the interference but they assume the latency for DRAM access is constant

Multi-Requestor Systems

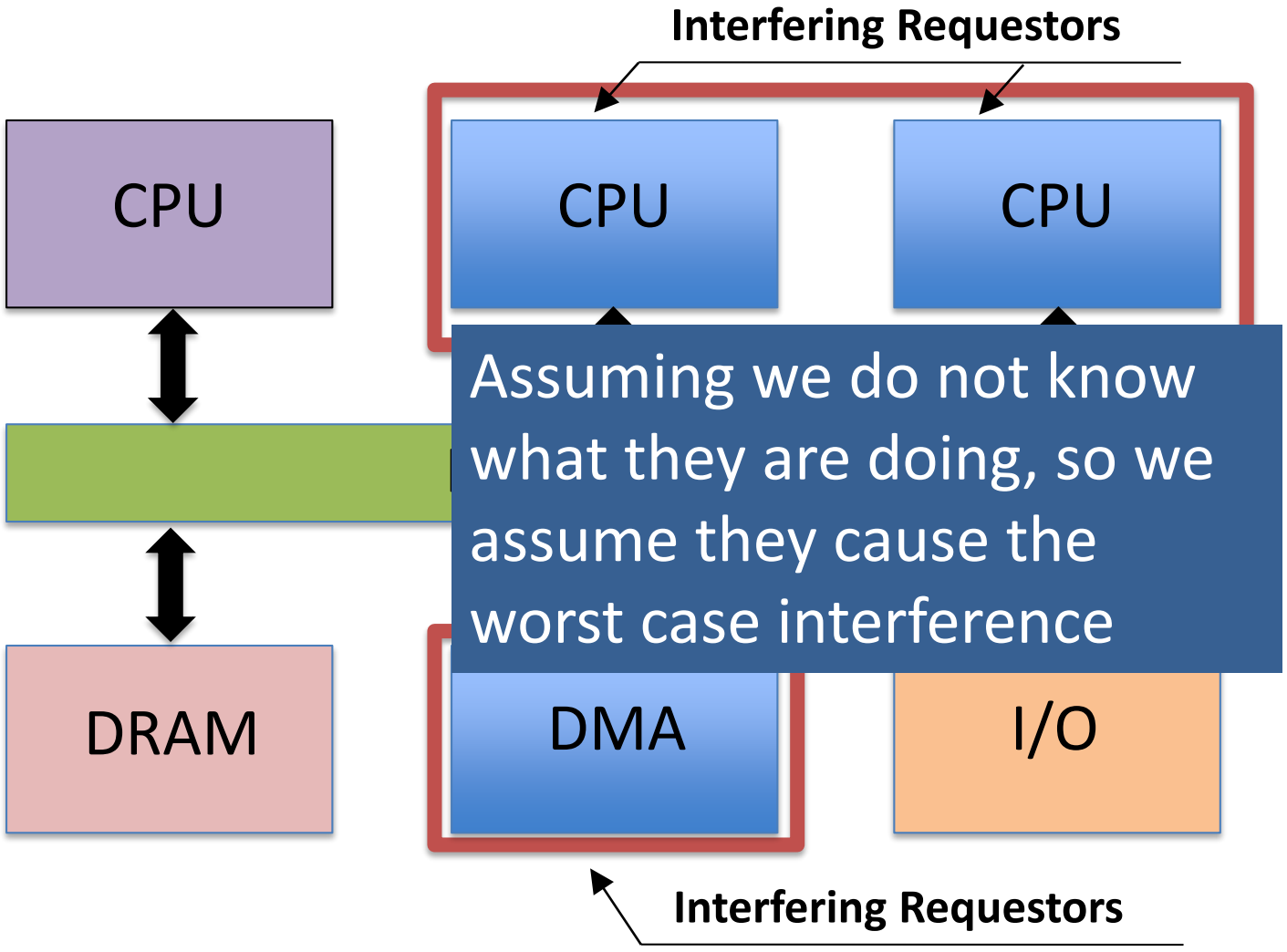
- Schedula ut
- WCET d
- WCET: n ource (e.g. cache, D
- Existing approaches can bound the interference but they assume the latency for DRAM access is constant

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

Contribution



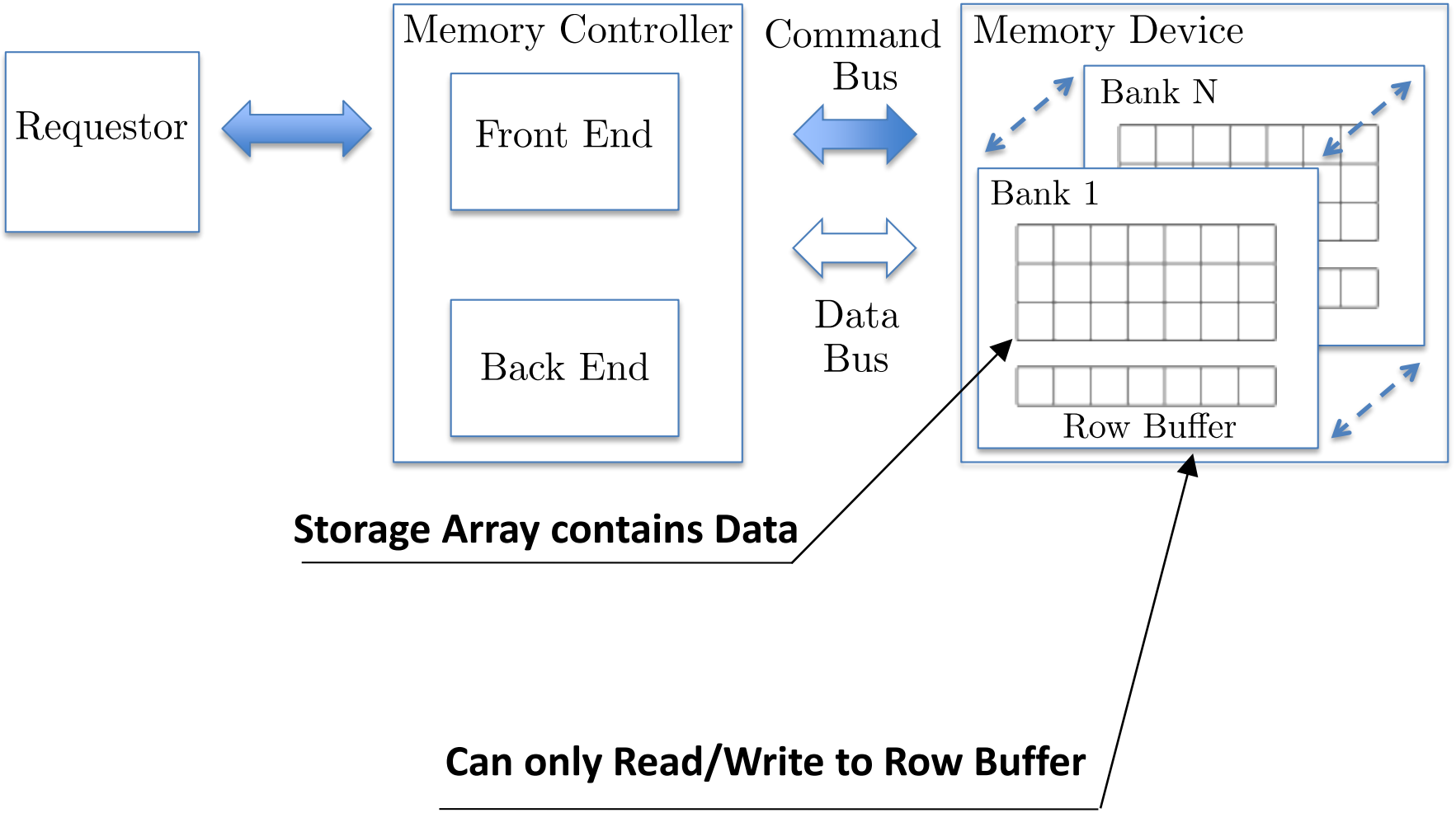
Contribution



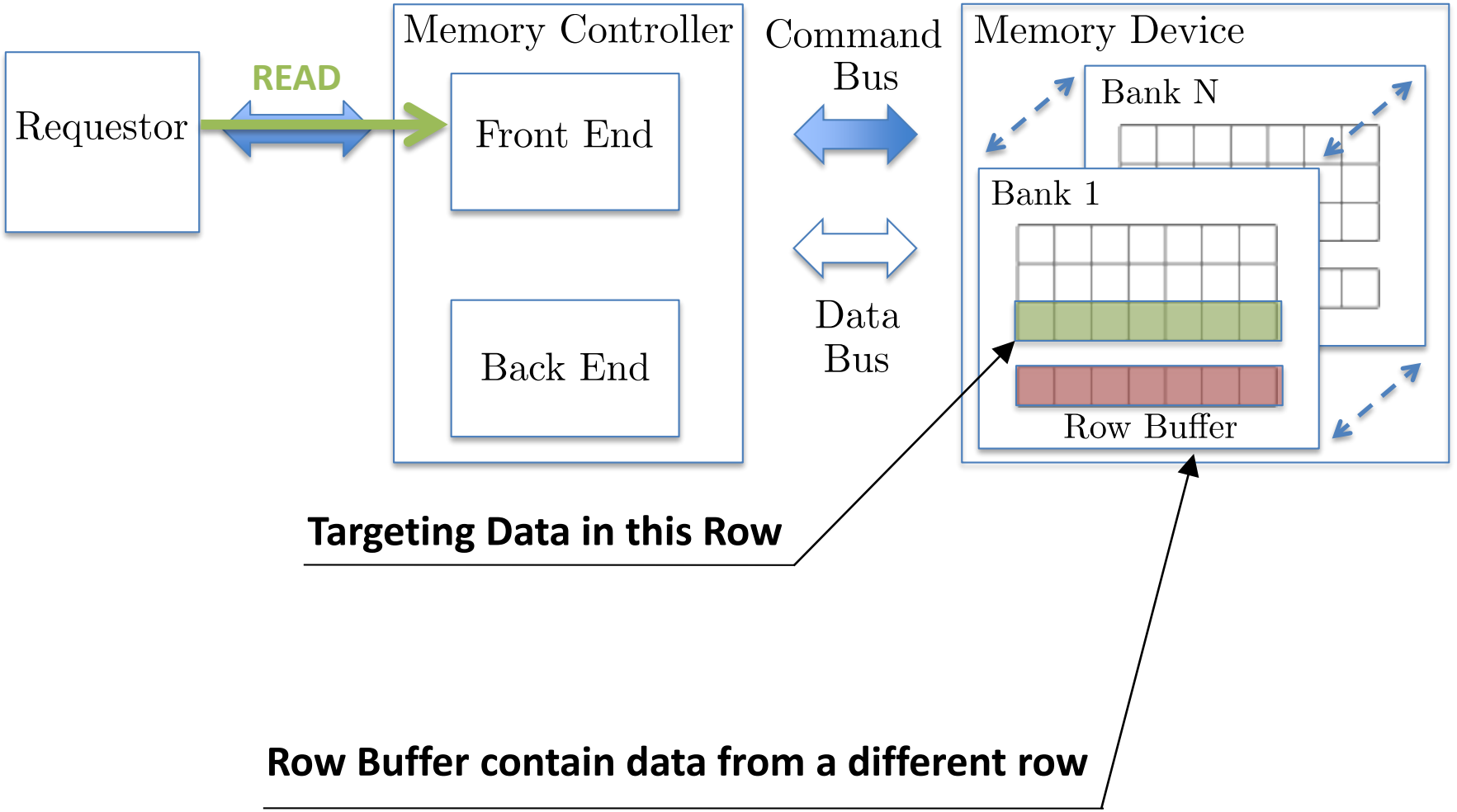
Outline

1. Background & Related Work
2. Memory Controller Model
3. Worst Case Latency Analysis
4. Results & Conclusion

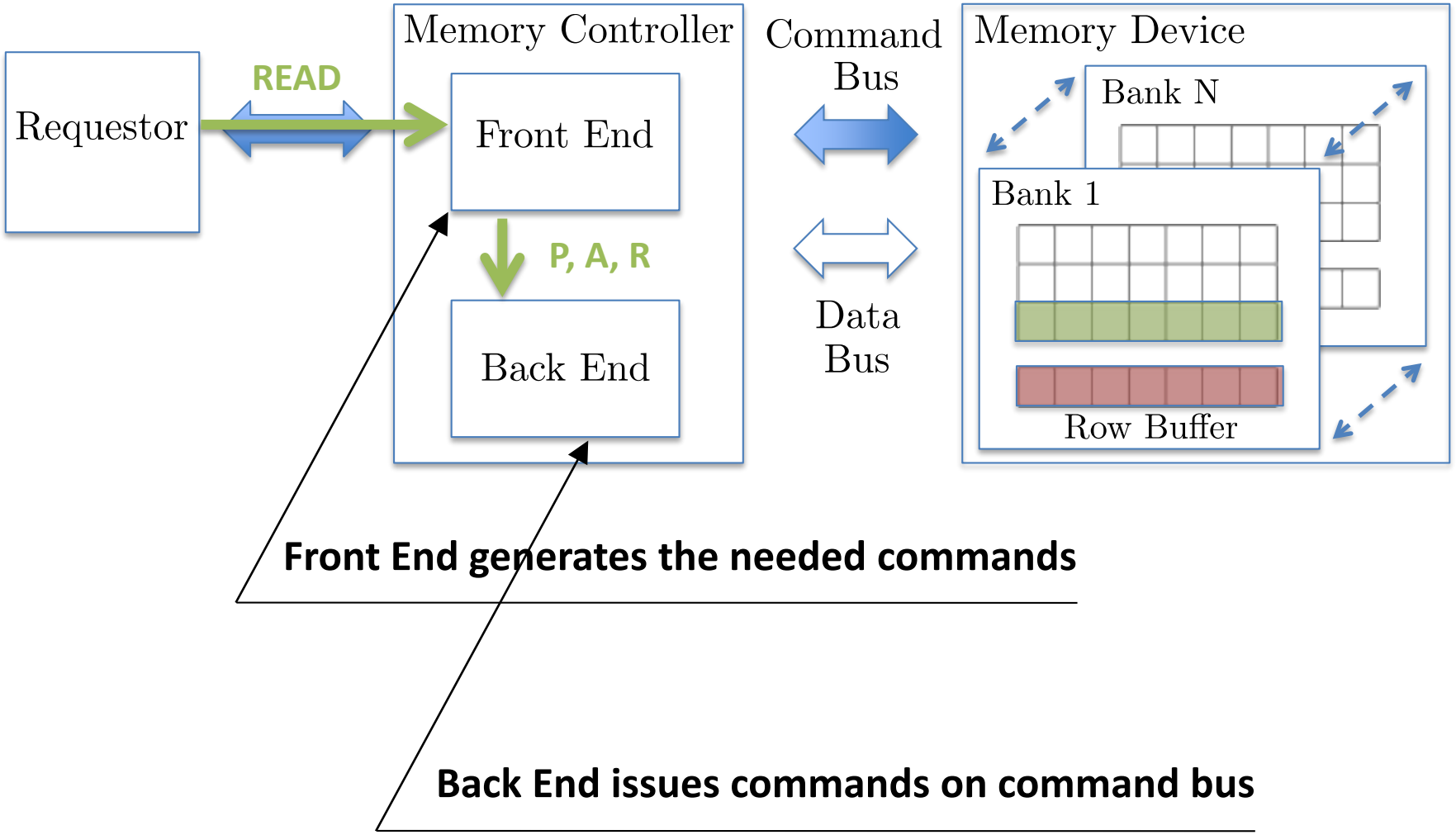
Background


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

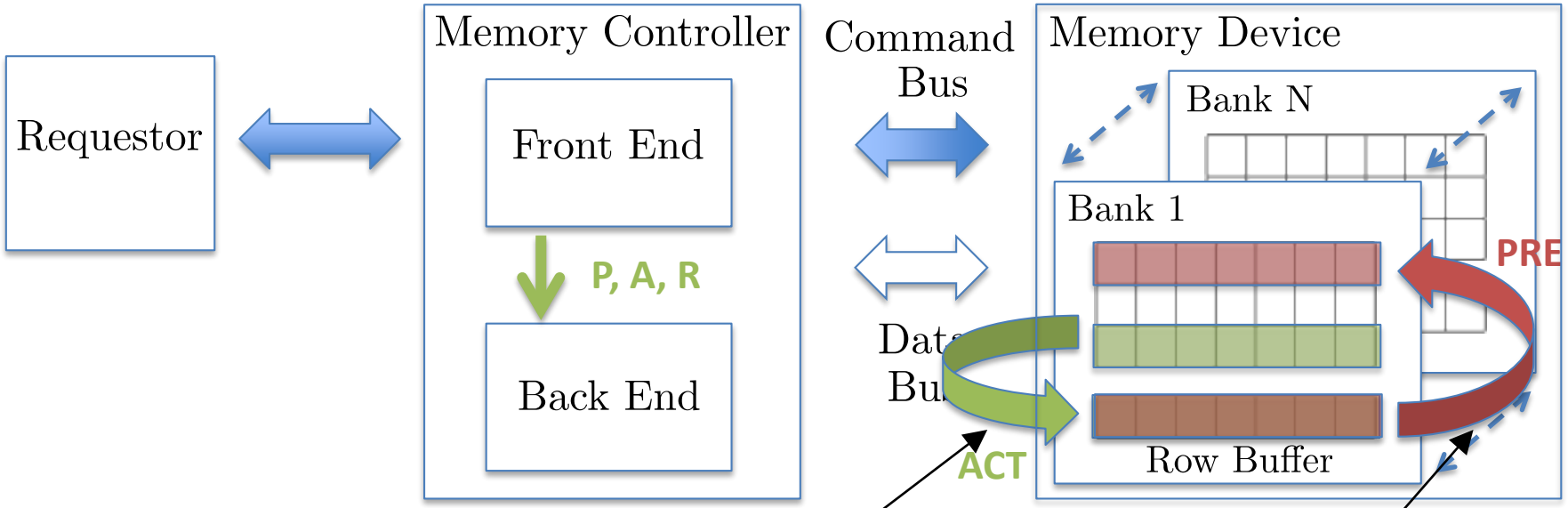
Background


Requestor
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

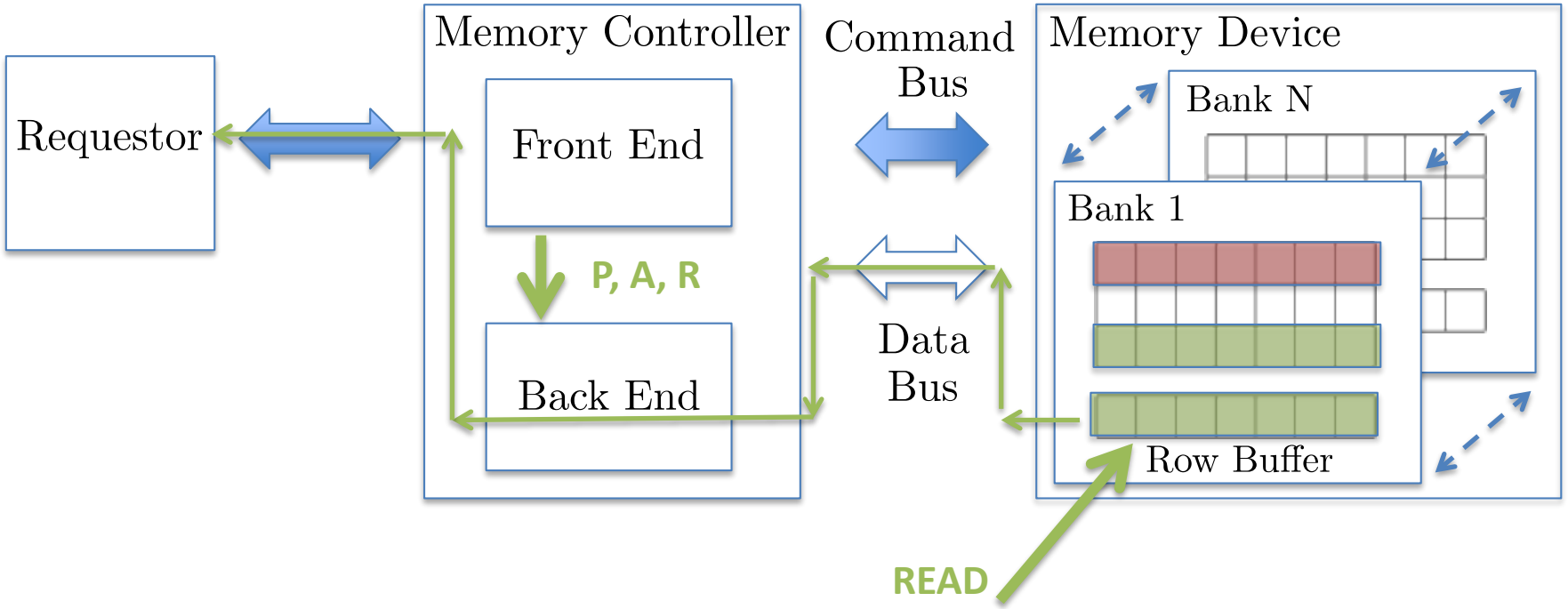

Requestor
READ
Memory Controller
Front End
P, A, R
Back End
Command Bus
Data Bus
Memory Device
Bank N
Bank 1
Row Buffer
Front End generates the needed commands
Back End issues commands on command bus

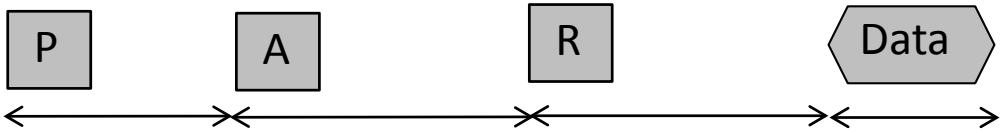
Background


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

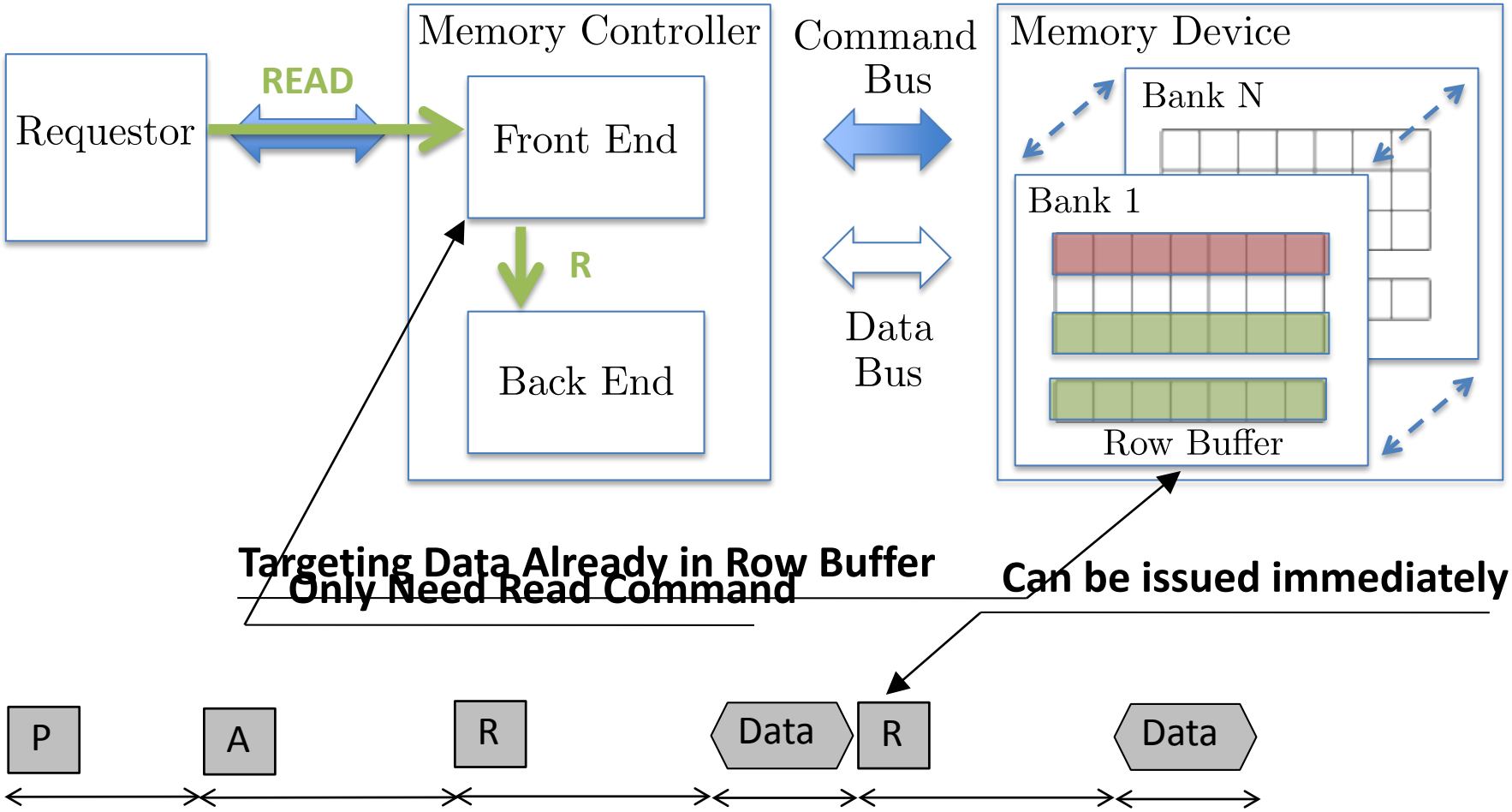

P
A
Pre-charge command issued on command bus
Timing Constraint

Background

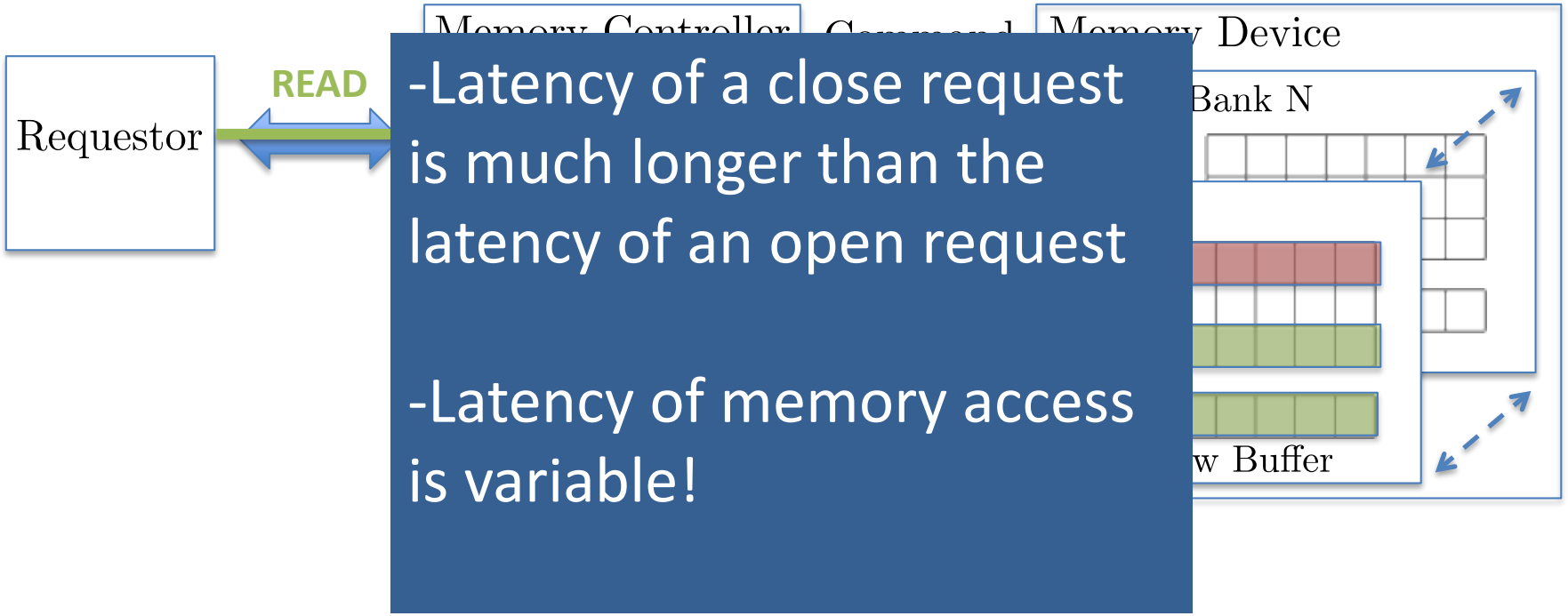

Memory Controller
Command Bus
Memory Device
Requestor
Front End
Bank N
Bank 1
P, A, R
Back End
Data Bus
Row Buffer
READ

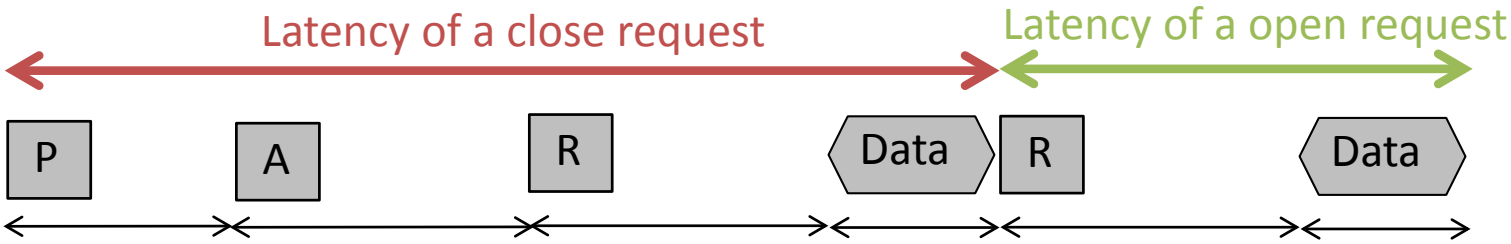

P
A
R
Data

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

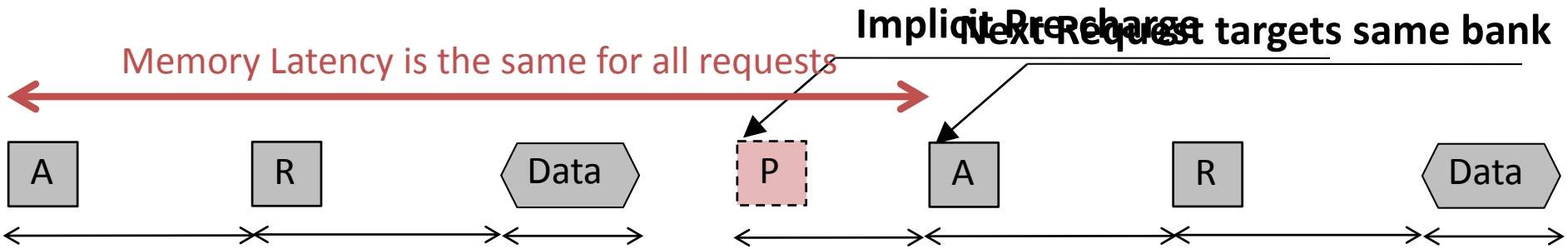

Requestor
READ
Memory Device
Bank N
w Buffer
-Latency of a close request is much longer than the latency of an open request
-Latency of memory access is variable!


Latency of a close request
Latency of a open request
P
A
R
Data
R
Data

Predictable Memory Controllers

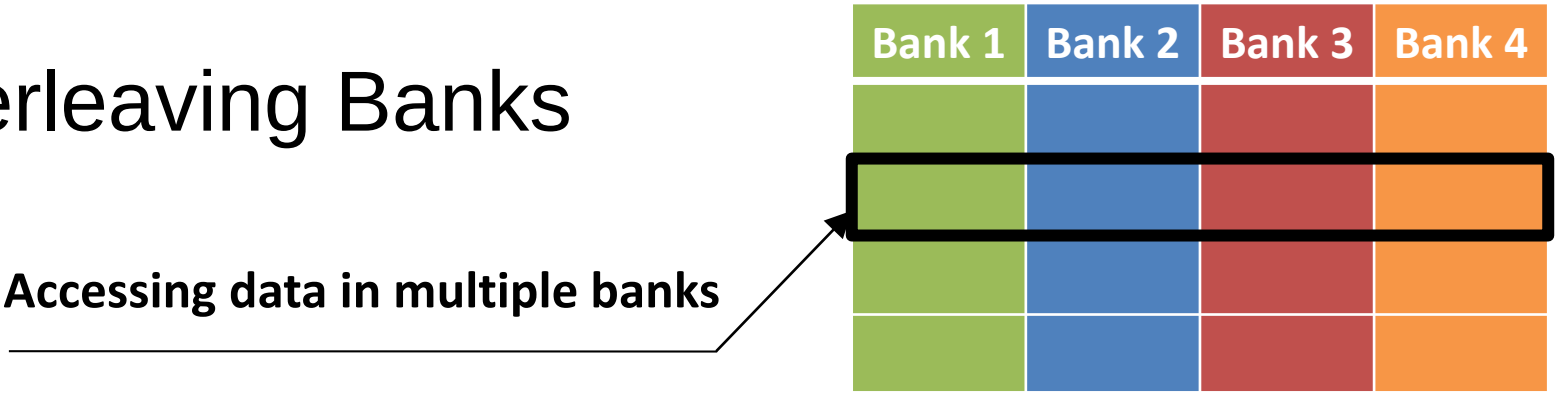
- Close Row Policy:
  - After ea automat

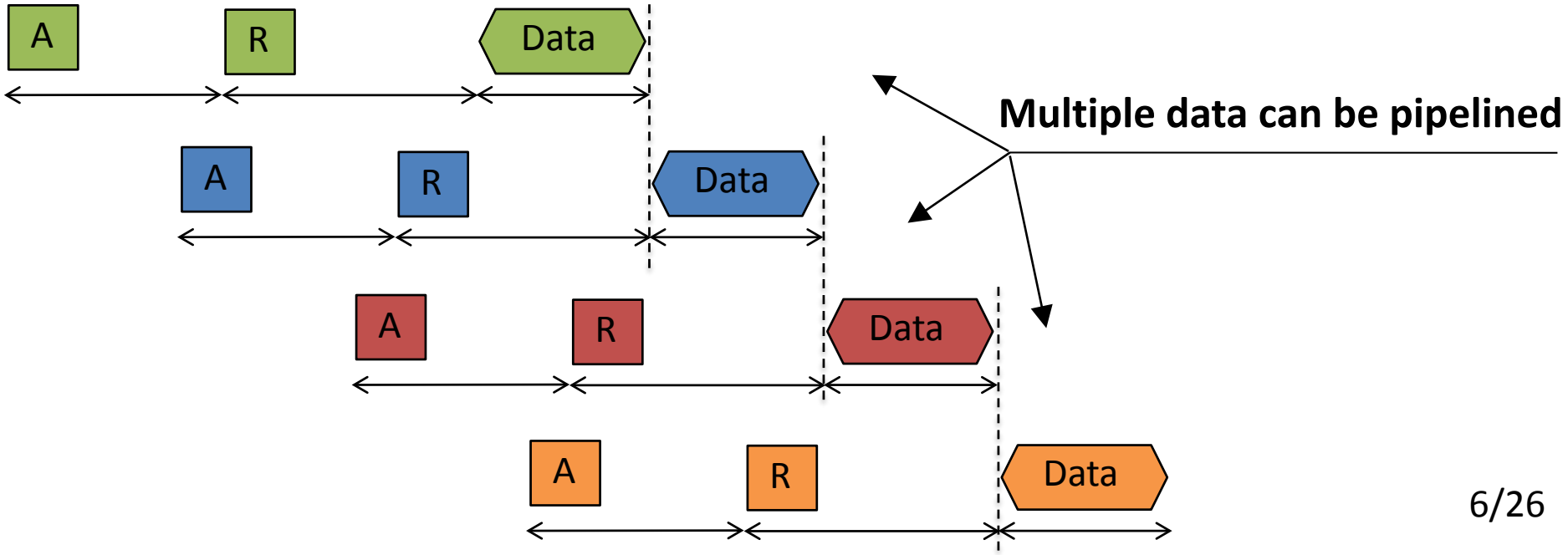
-Can not take advantage of locality (row hits)
-Latency is much longer than open request



Predictable Memory Controllers

- Interleaving Banks

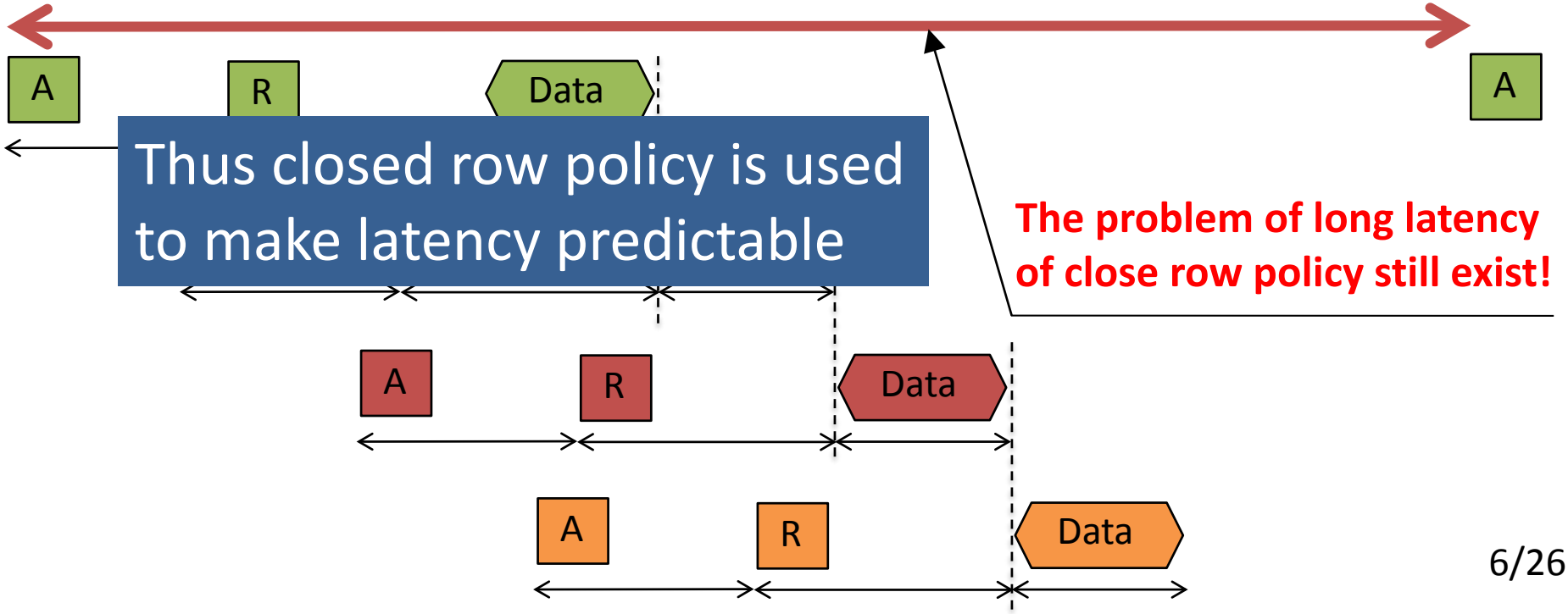




Predictable Memory Controllers

- Interleaving Banks

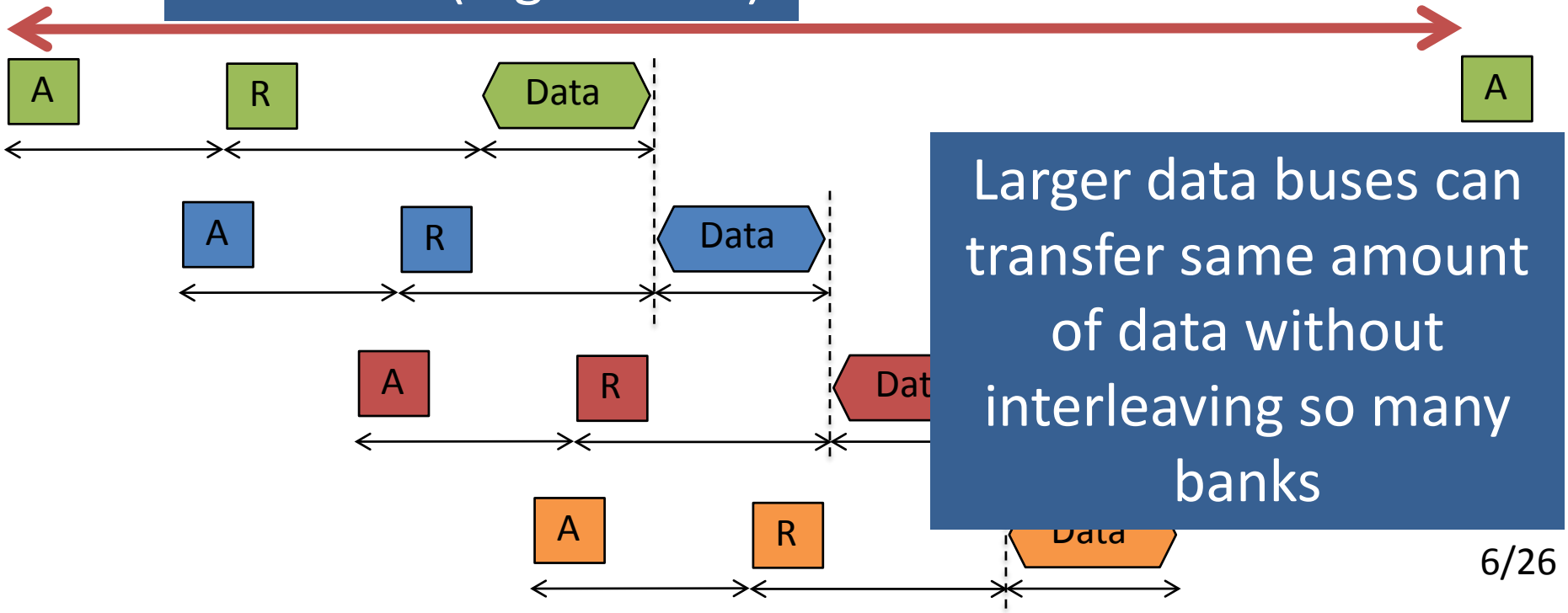
Problem: requestors can close each other's row buffer since they can access all banks



Predictable Memory Controllers

- Interleaving Banks

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

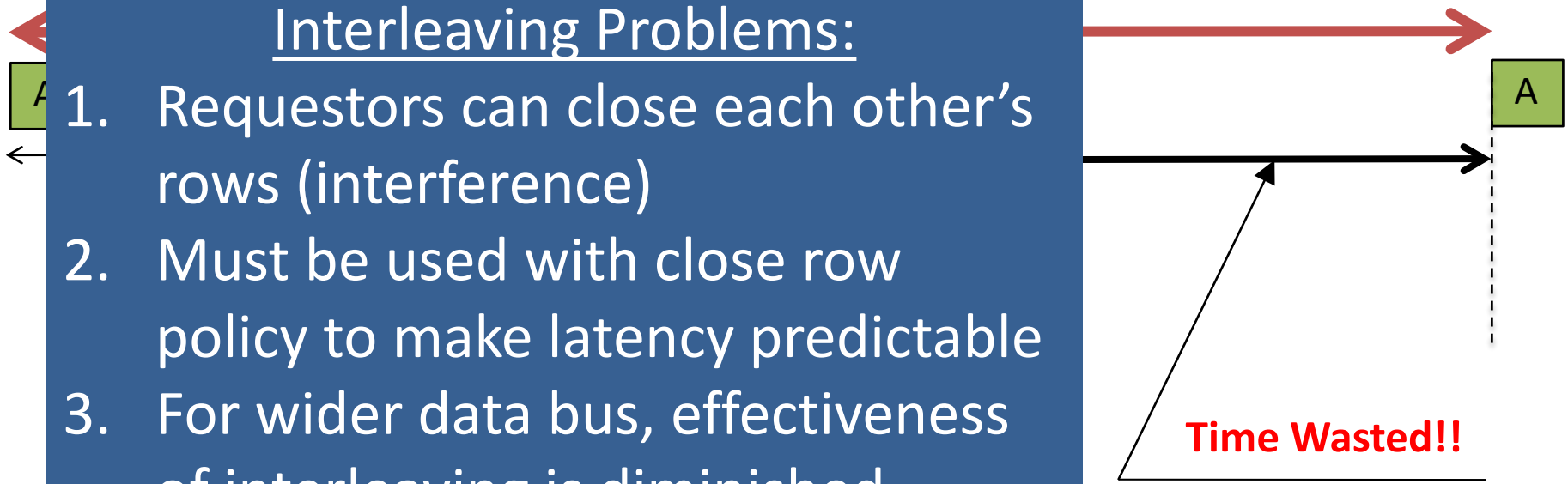


Larger data buses can transfer same amount of data without interleaving so many banks

Predictable Memory Controllers

- Interleaving Banks

Interleaving two banks for wider data bus (e.g. 32 bits)

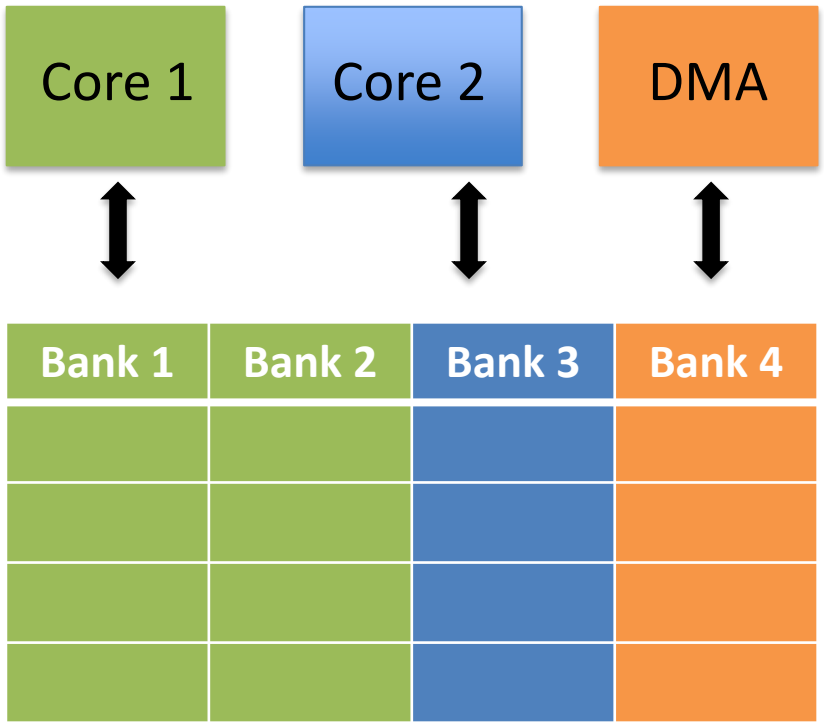


Interleaving Problems:

1. Requestors can close each other's rows (interference)
2. Must be used with close row policy to make latency predictable
3. For wider data bus, effectiveness of interleaving is diminished

Predictable Memory Controllers

- Private Banks



- Can partition banks to either requestors or tasks
- This can be done by:
  - Hardware if Memory controller supports
  - By compiler
  - In OS, using virtual memory

Related Work

- AMC[1] and Predator [2]:
  -Close Row Policy
  -Interleaved Bank

- Conservative Open-Page [3]:
  – Interleaved Bank
  – Leave row open for a small window of time

- PRET DRAM Controller [4]:
  – Close Row Policy
  – Private Bank

Our Approach

- Private Bank
  - elim… other requ…

Challenge:

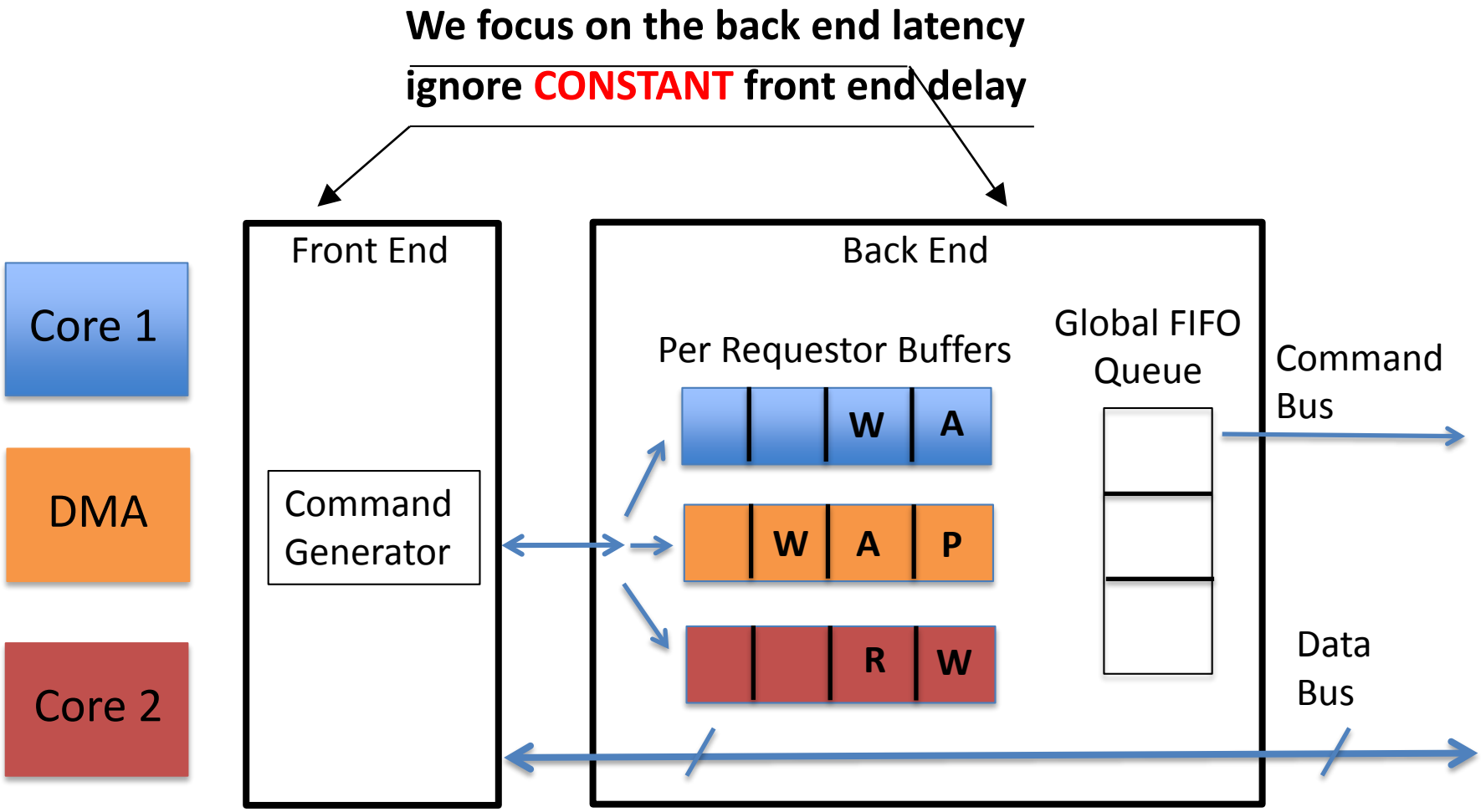
1. Analysis is more complex
2. More than 20 timing constraints
3. Latency depends on the dynamic state of DRAM

- Open Row Policy
  - reduce latency and take advantage or row hit ratio (locality)

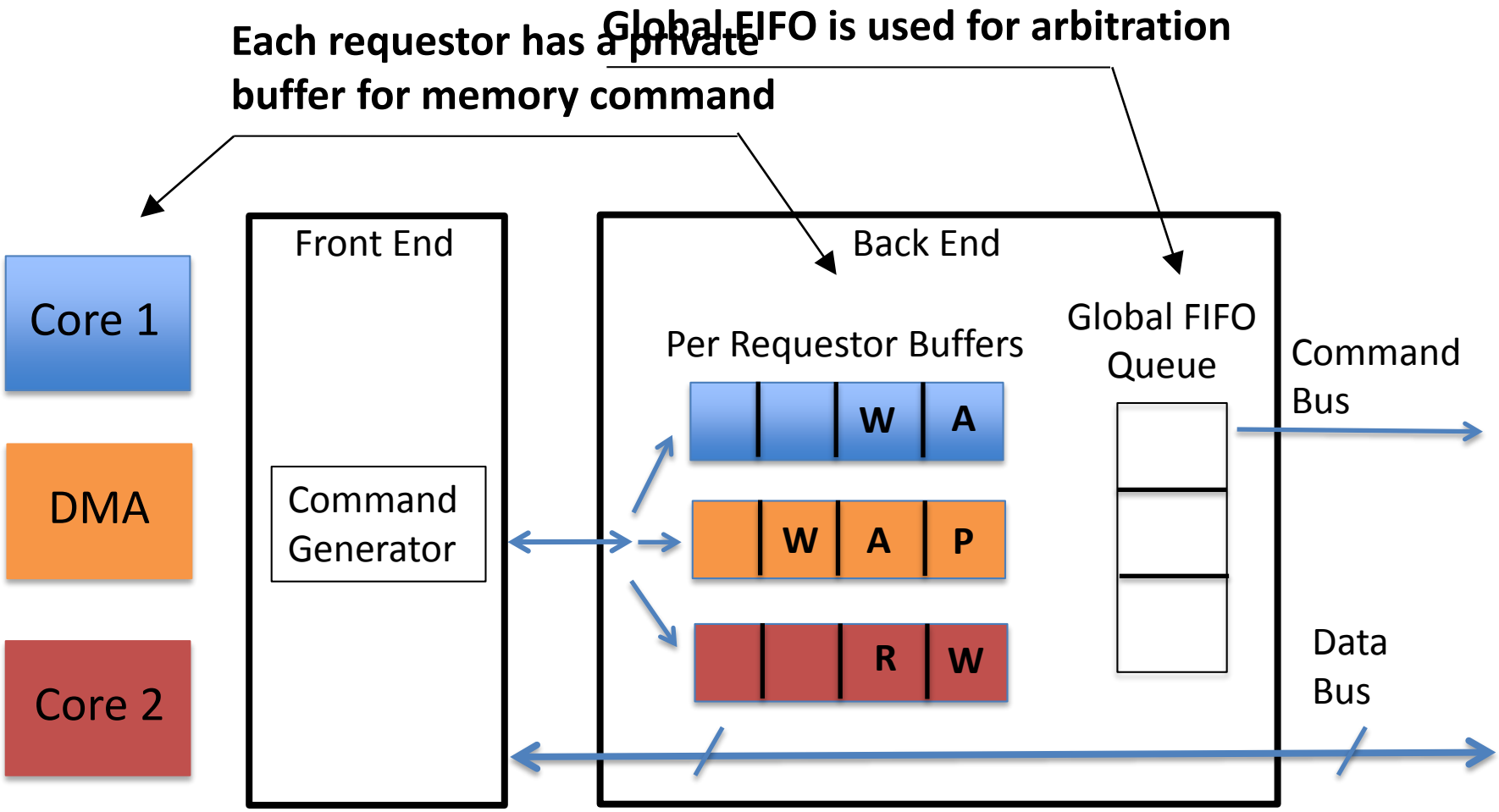
Outline

1. Background & Related Work
2. Memory Controller Model
3. Worst Case Latency Analysis
4. Results & Conclusion

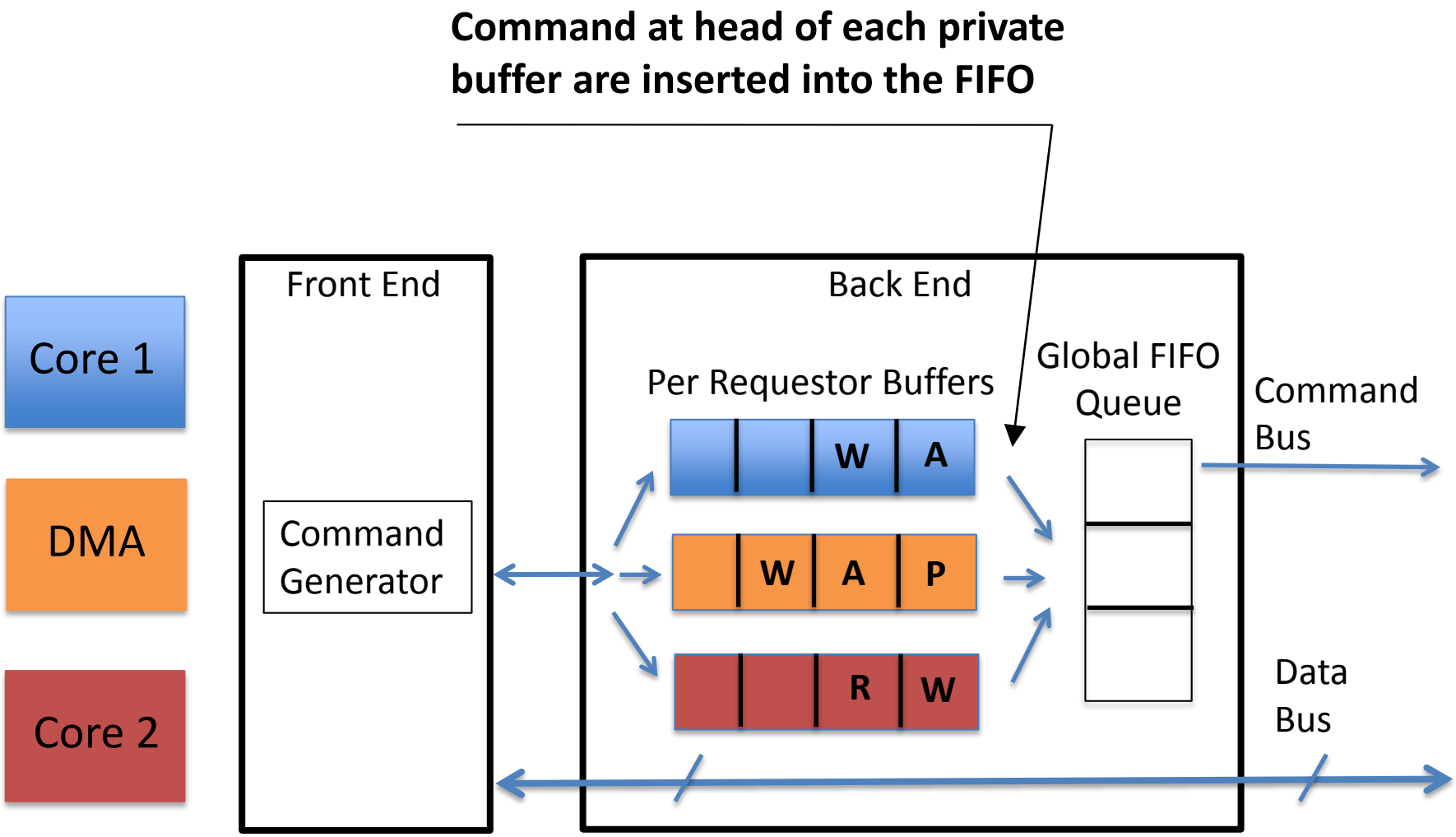
Memory Controller Model



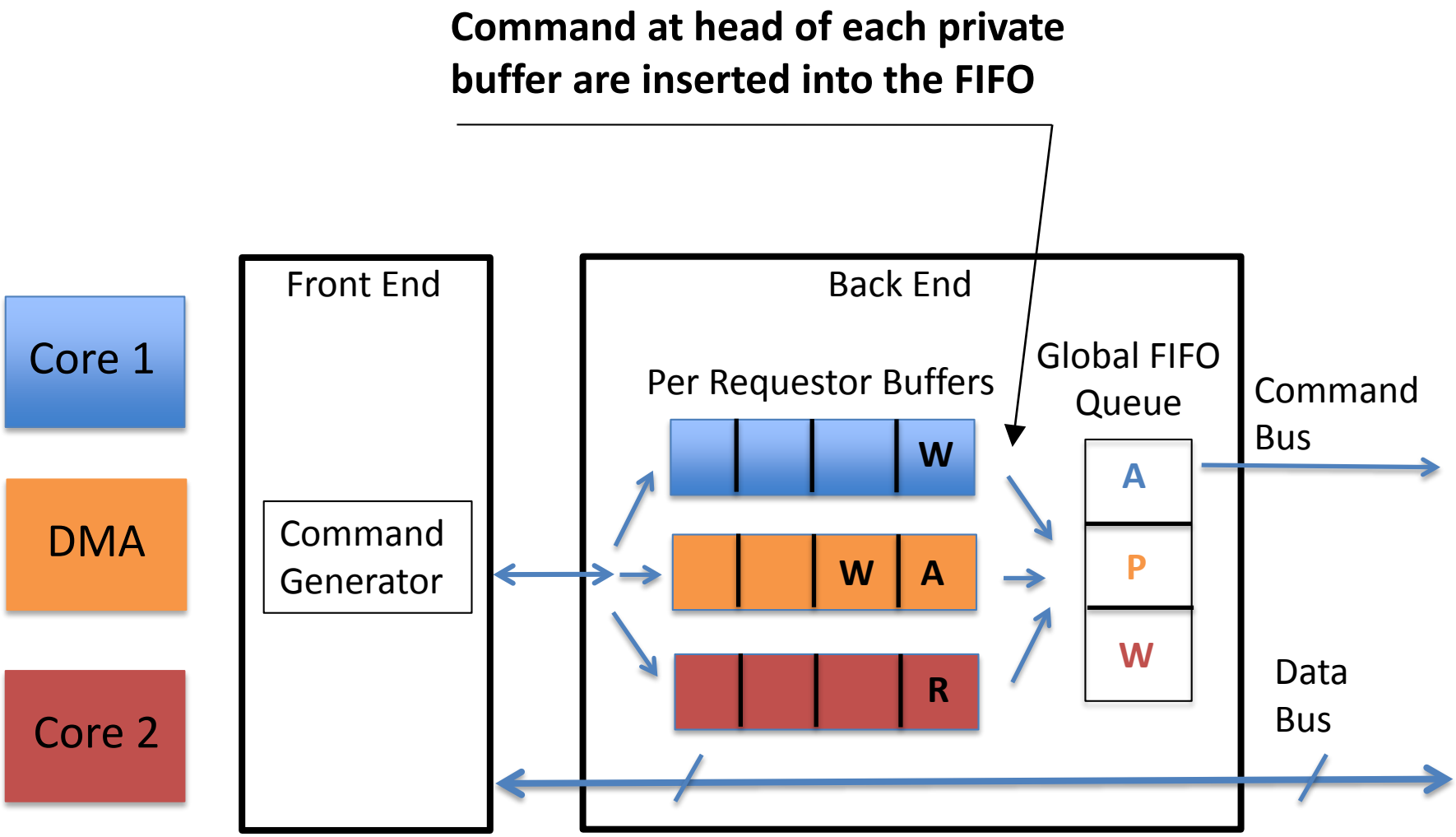
Memory Controller Model



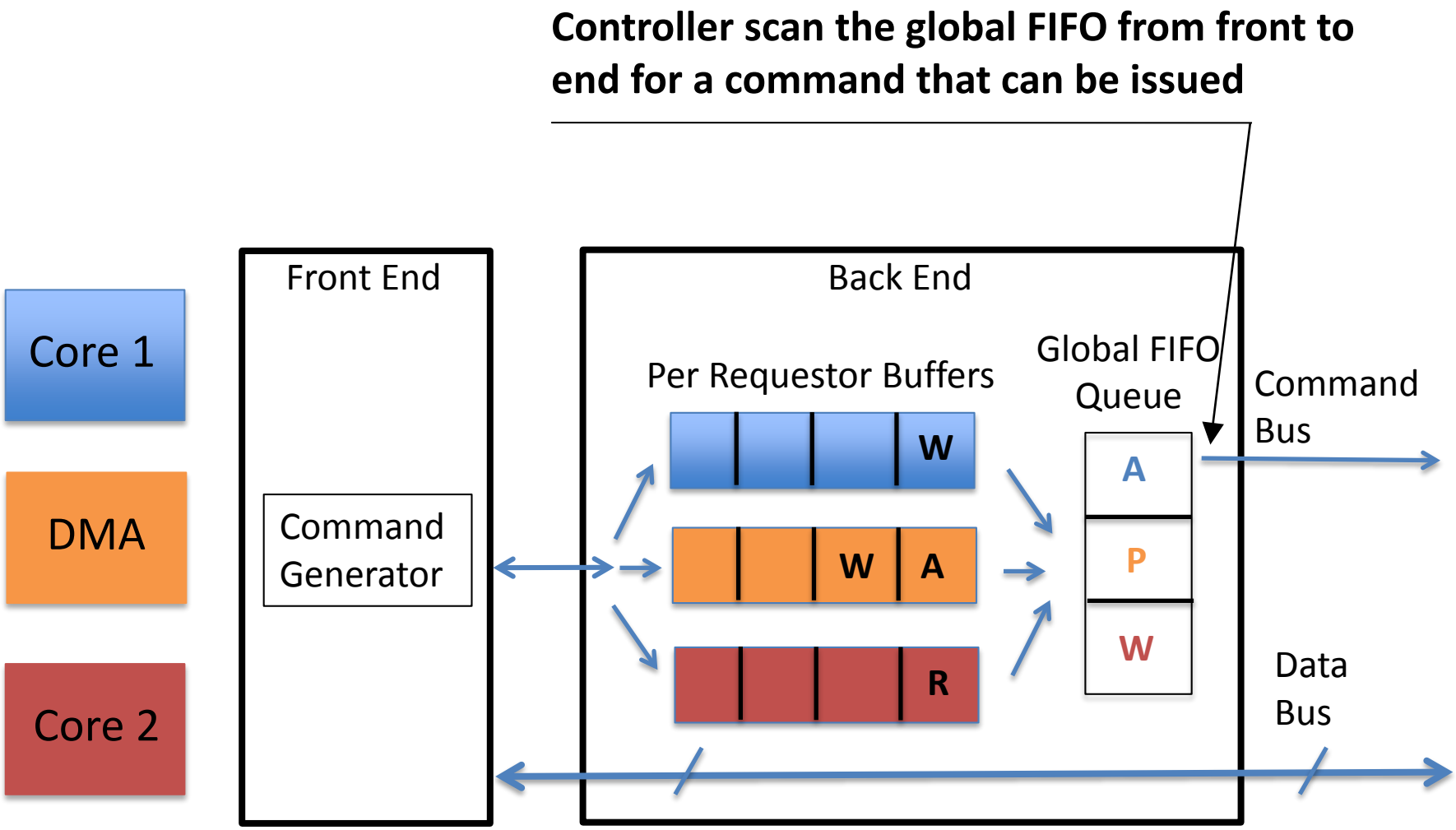
Memory Controller Model



Memory Controller Model

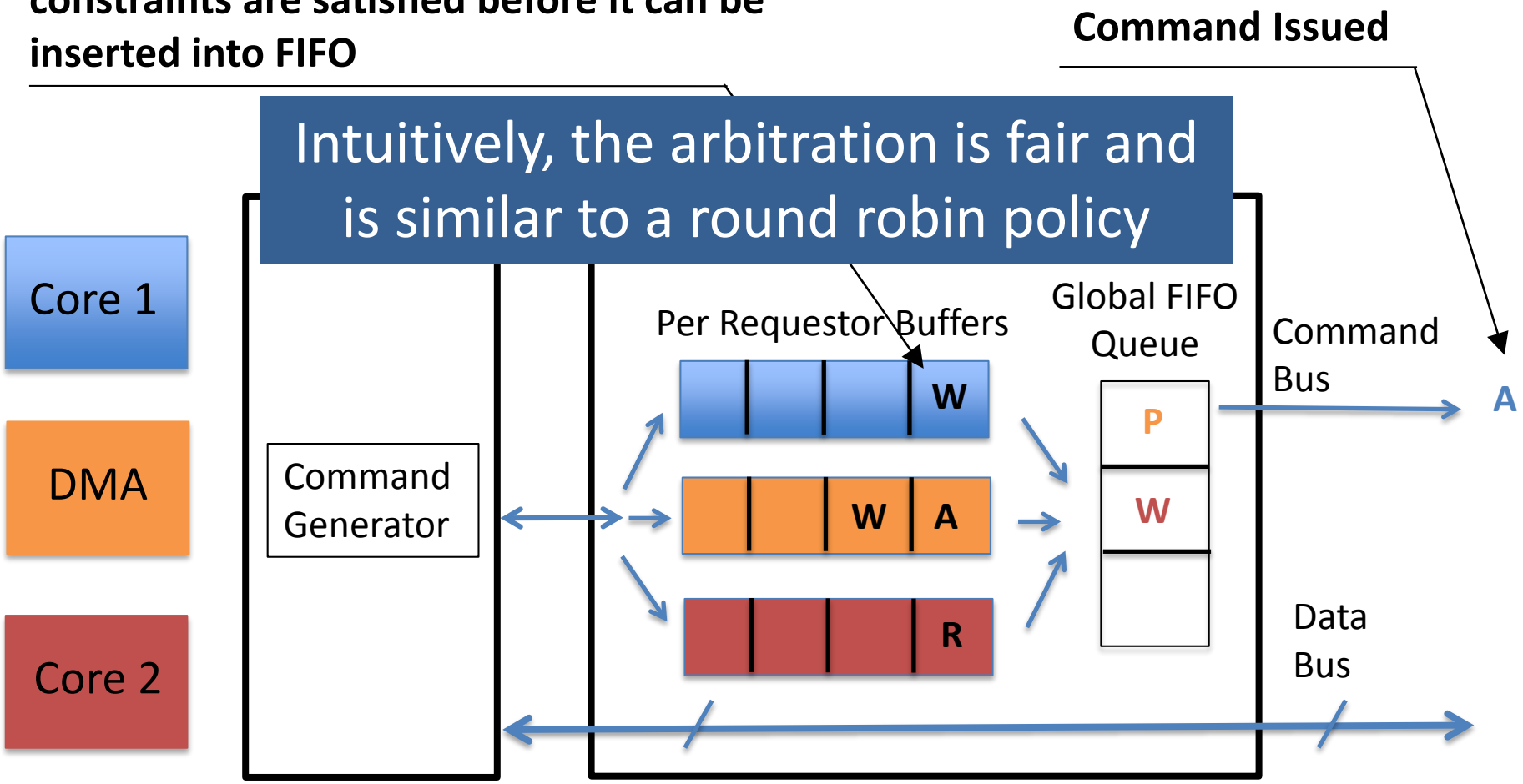


Memory Controller Model



Memory Controller Model

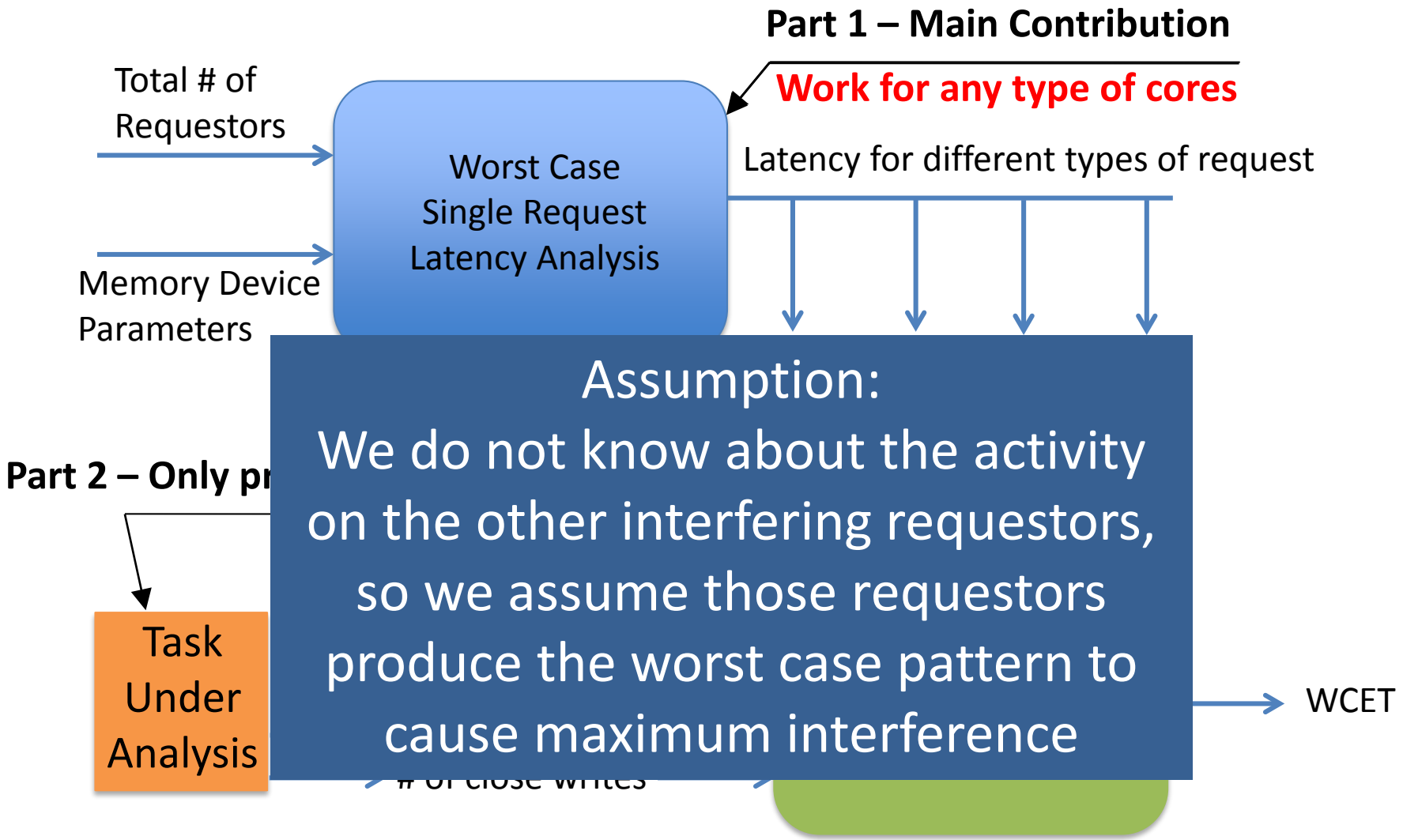
Next command must wait until timing constraints are satisfied before it can be inserted into FIFO



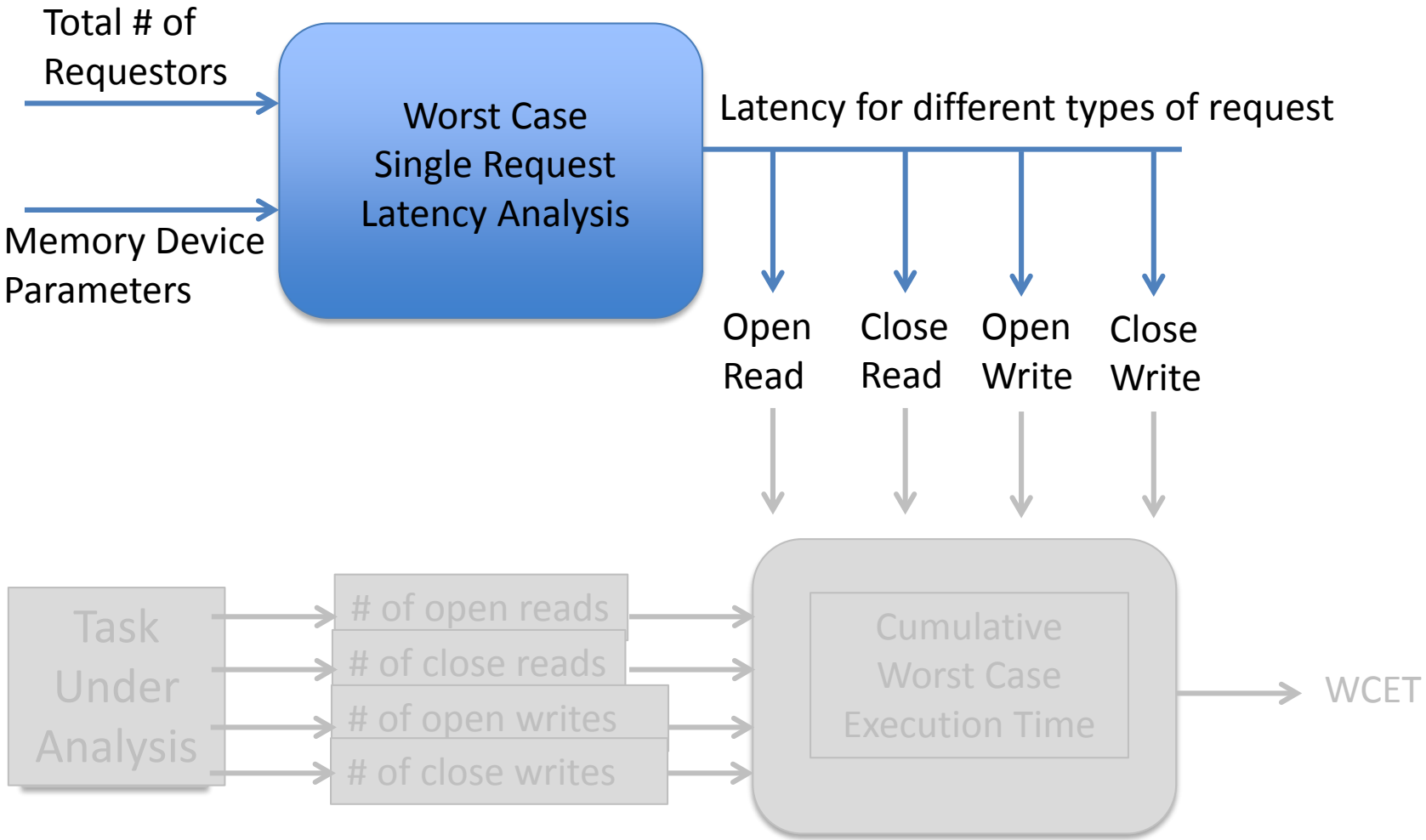
Outline

1. Background & Related Work
2. Memory Controller Model
3. Worst Case Latency Analysis
4. Results & Conclusion

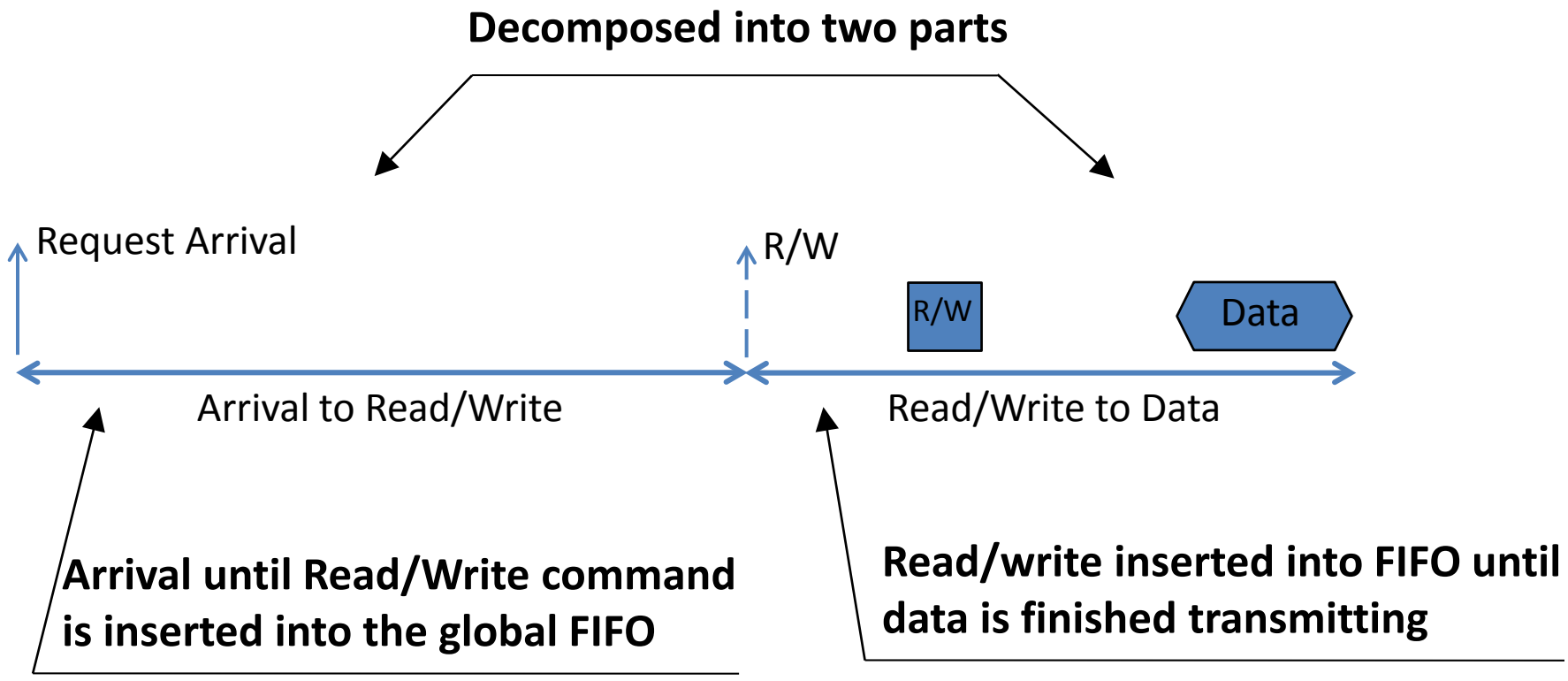
Worst Case Analysis



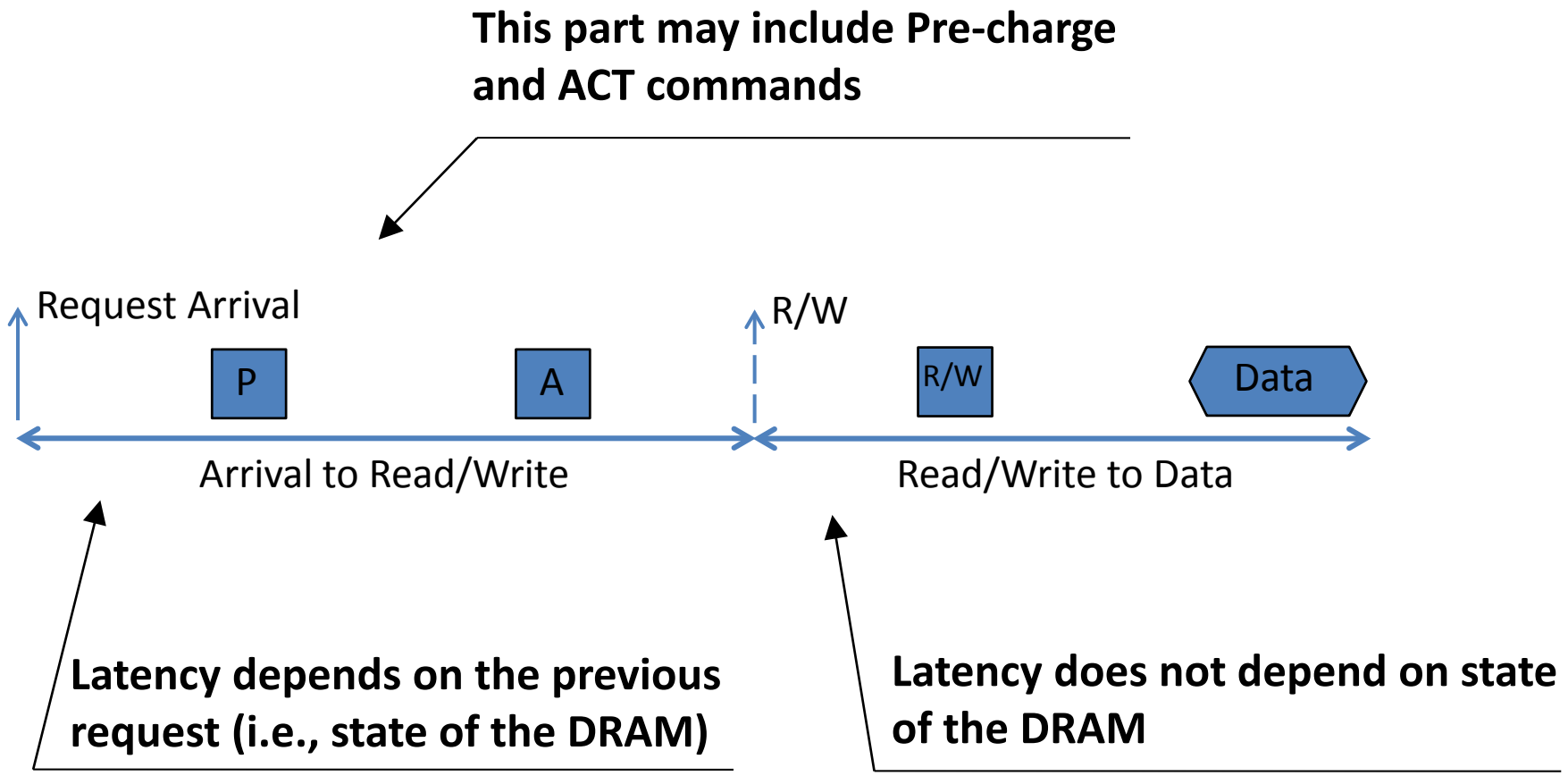
Worst Case Analysis



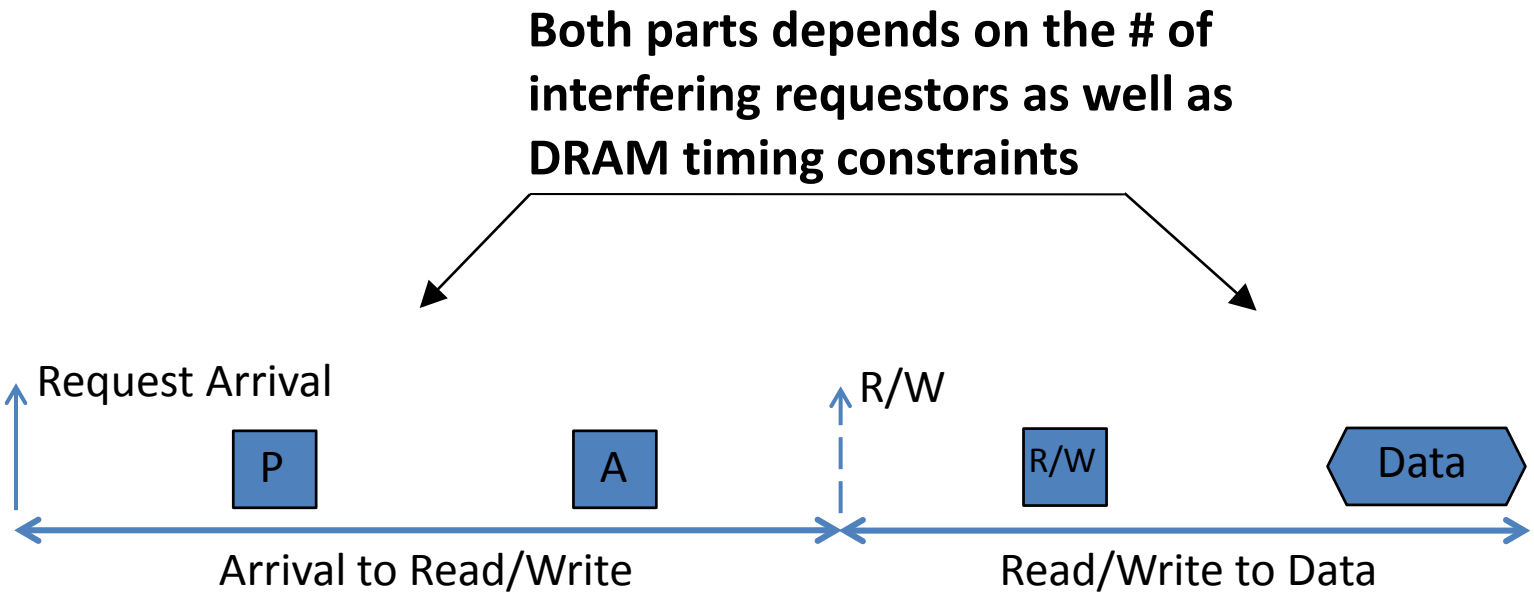
Single Request Latency



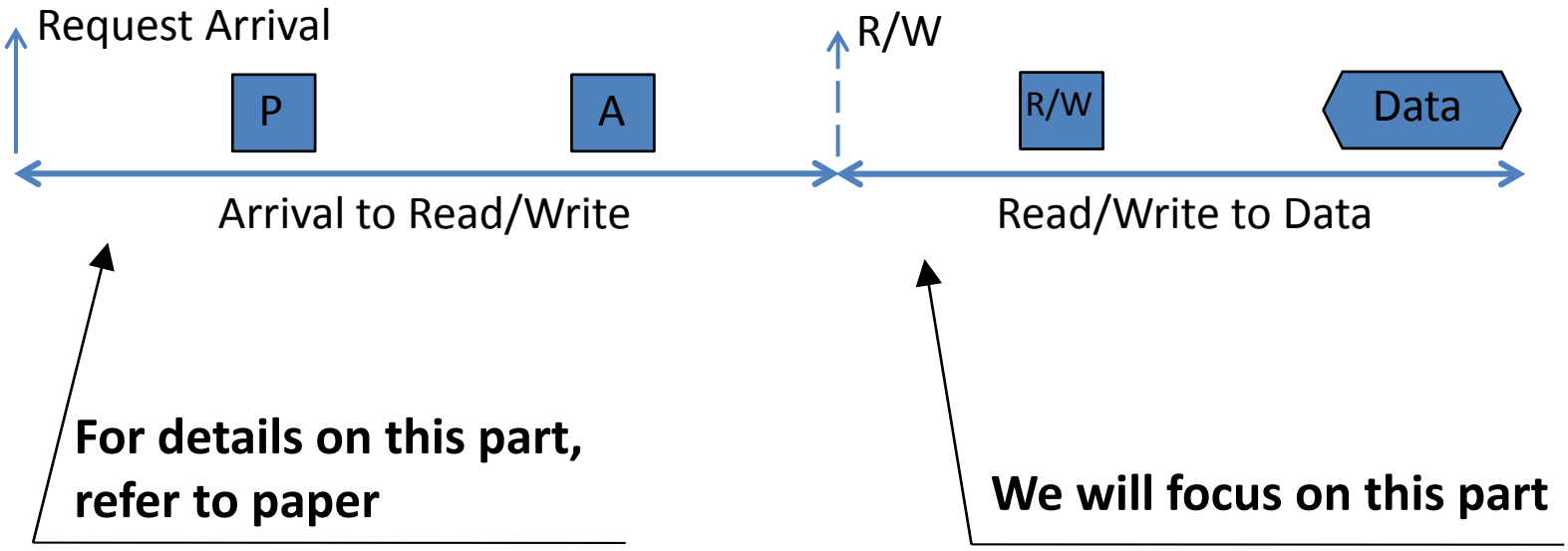
Single Request Latency



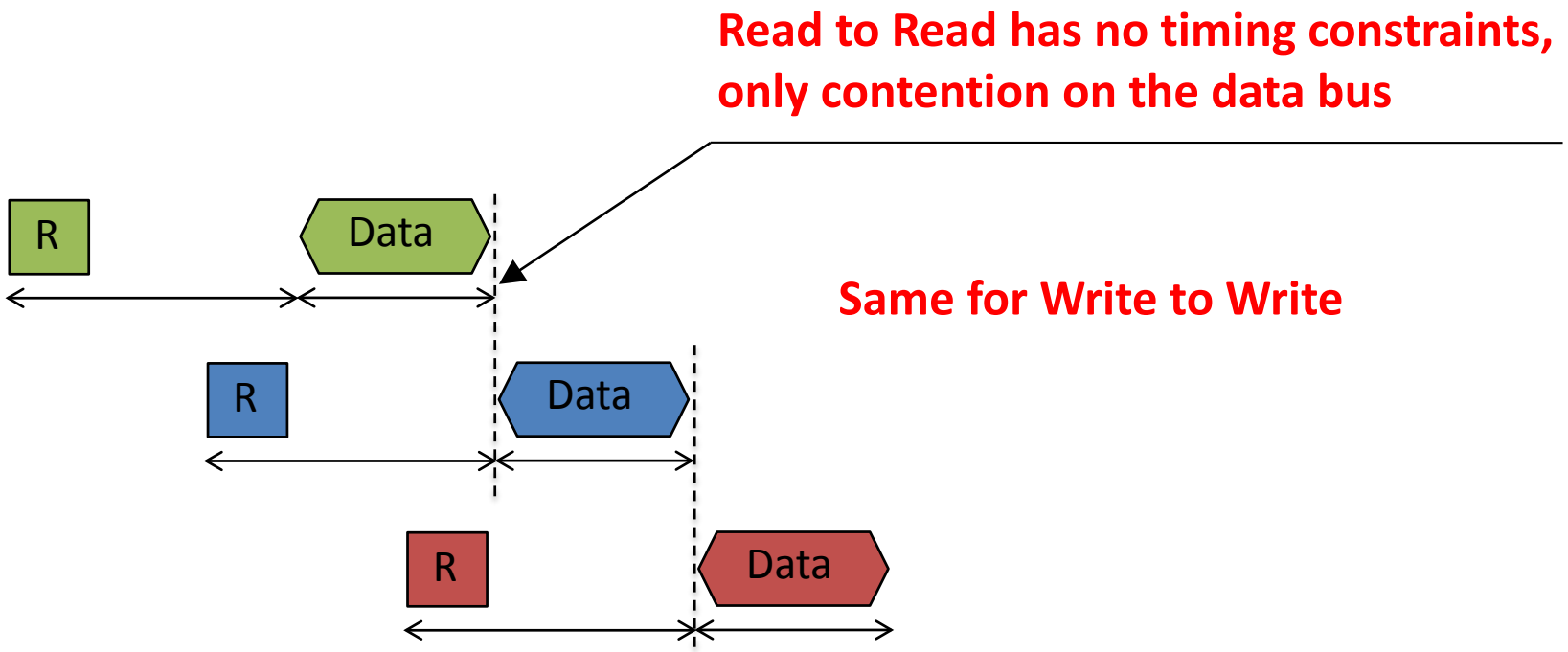
Single Request Latency



Single Request Latency

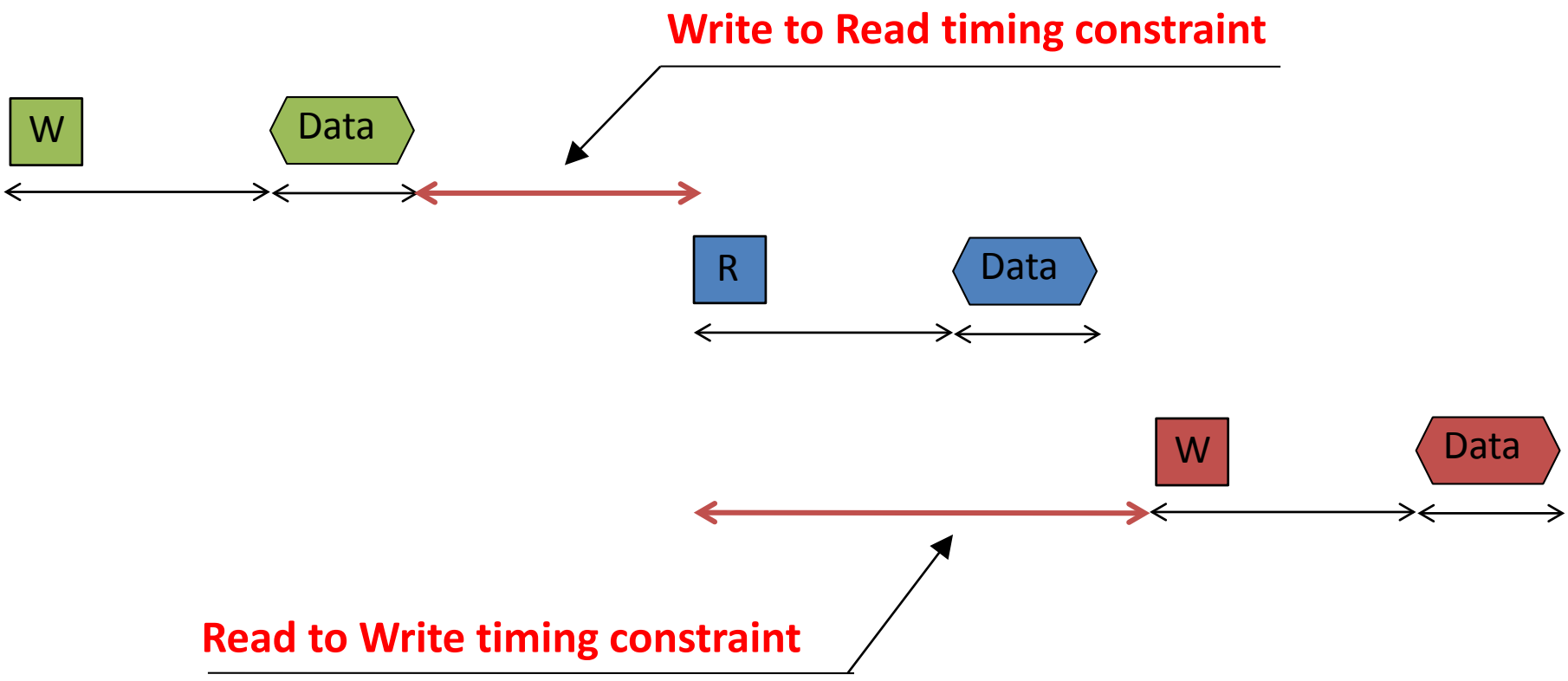


Read/Write to Data Latency



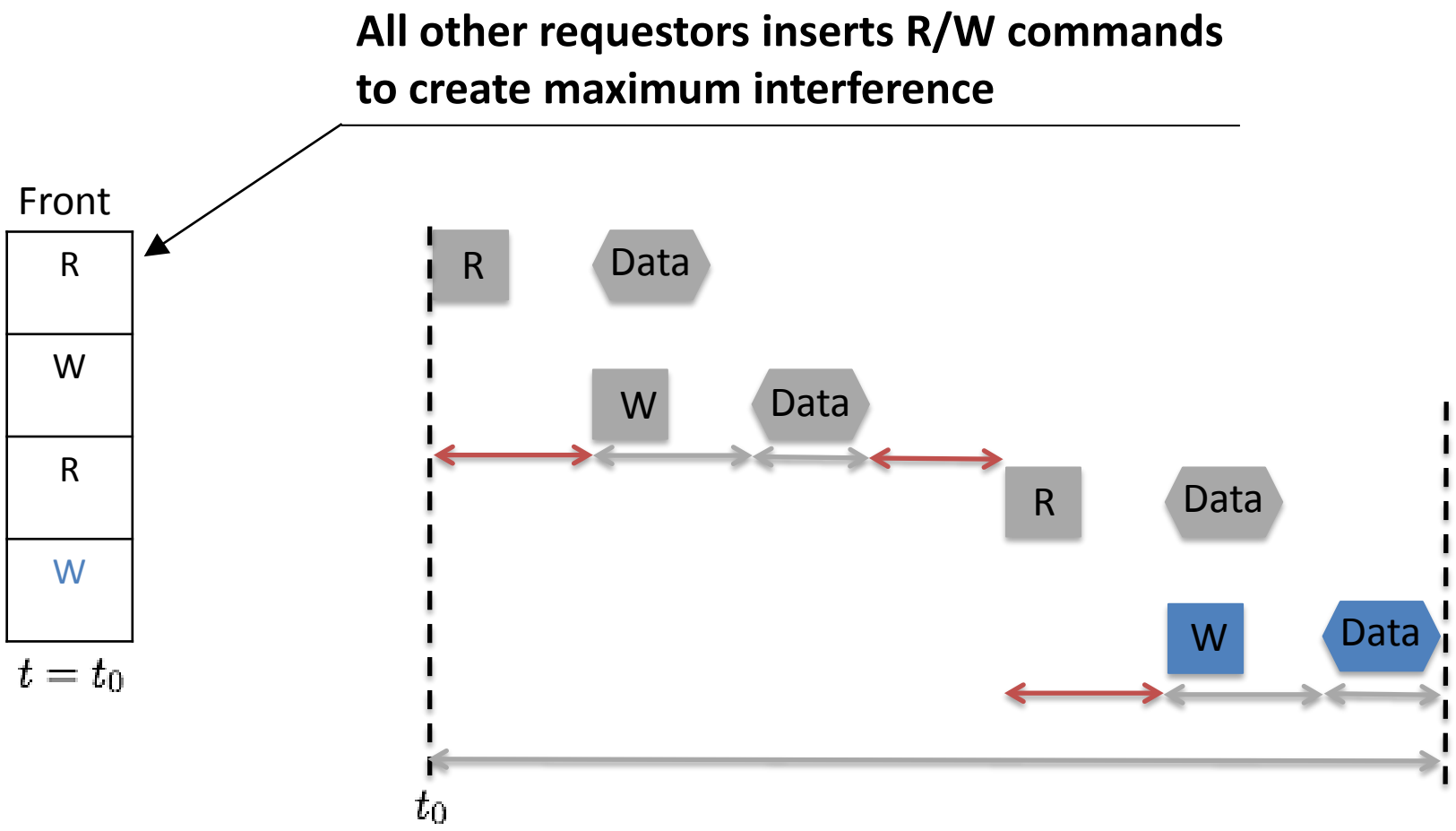
Read/Write to Data Latency

Therefore, an alternation of read and write commands produce longer latency



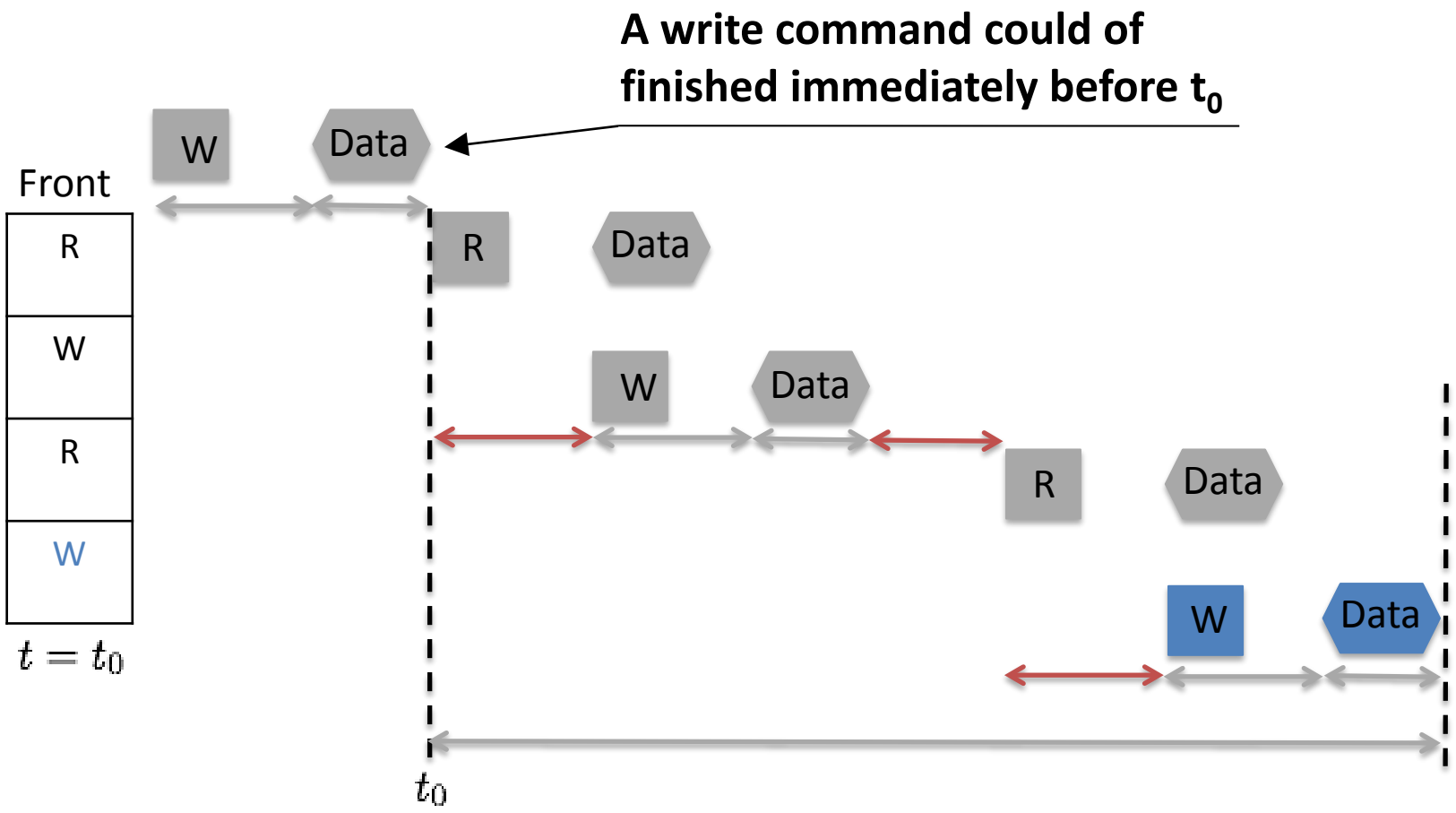
Read/Write to Data Latency

- Interference on Write command



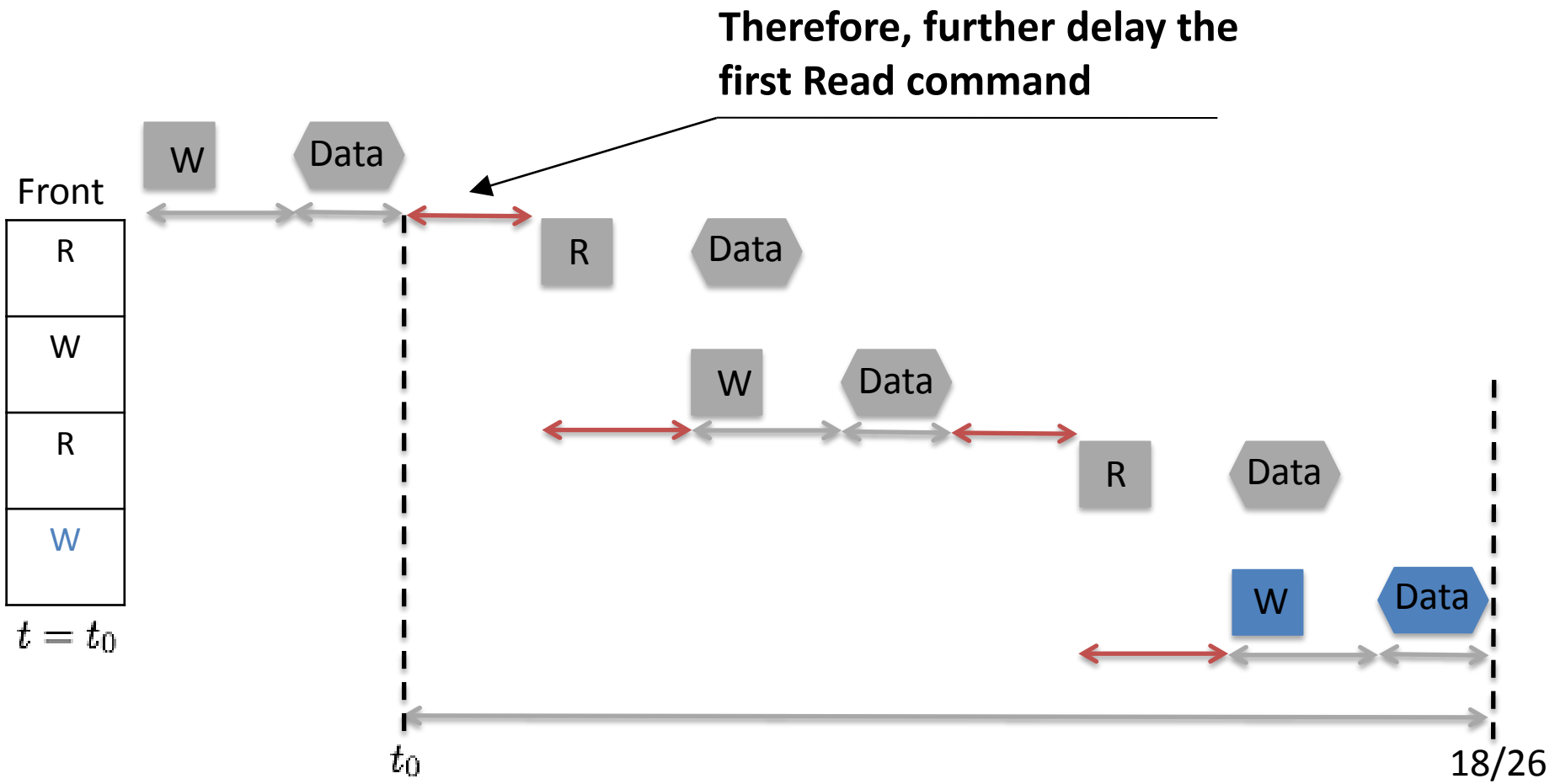
Read/Write to Data Latency

- Interference on Write command

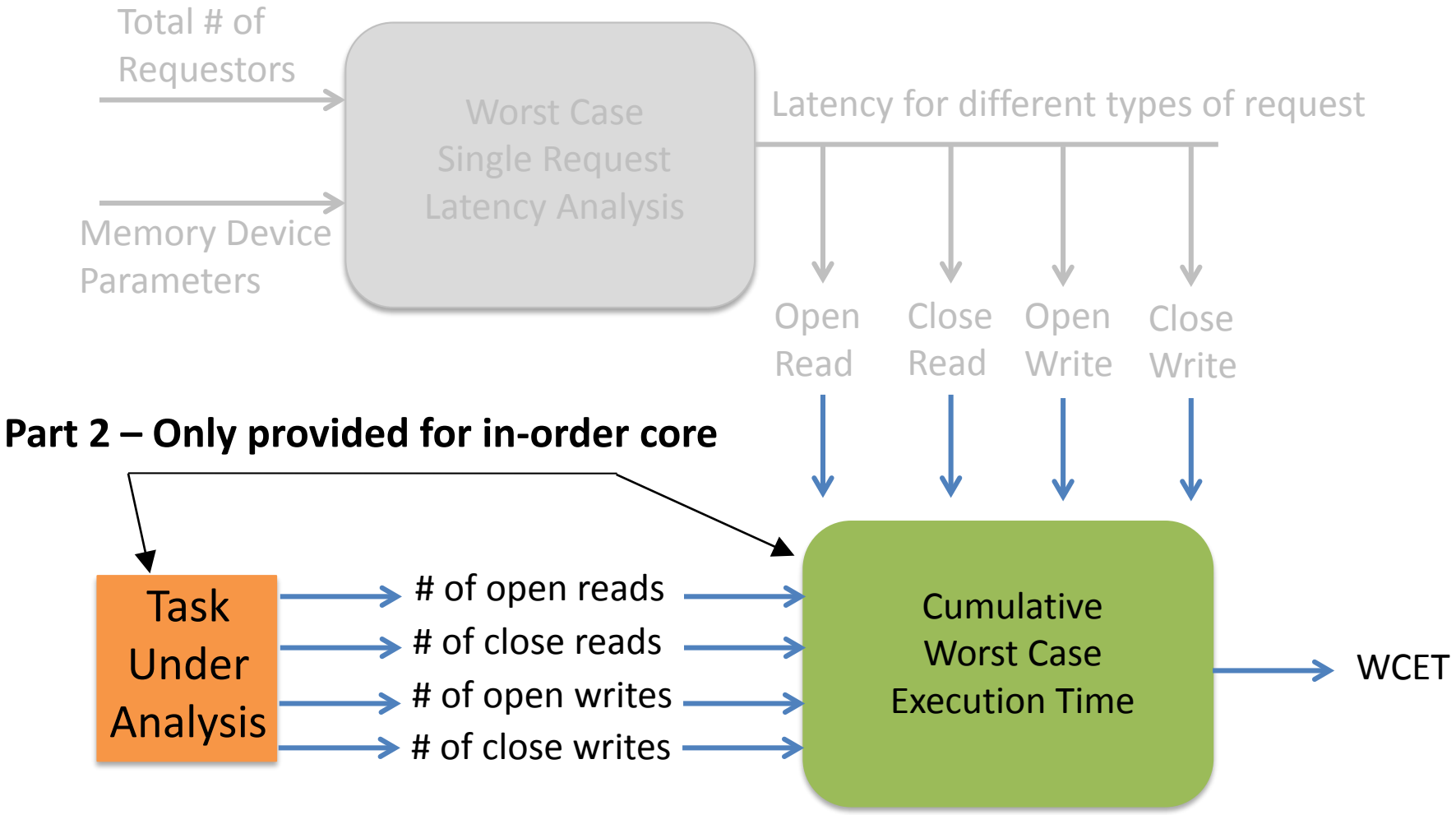


Read/Write to Data Latency

- Interference on Write command



Worst Case Analysis



Cumulative Latency

Task Under Analysis:

Cumulative Latency

Open F

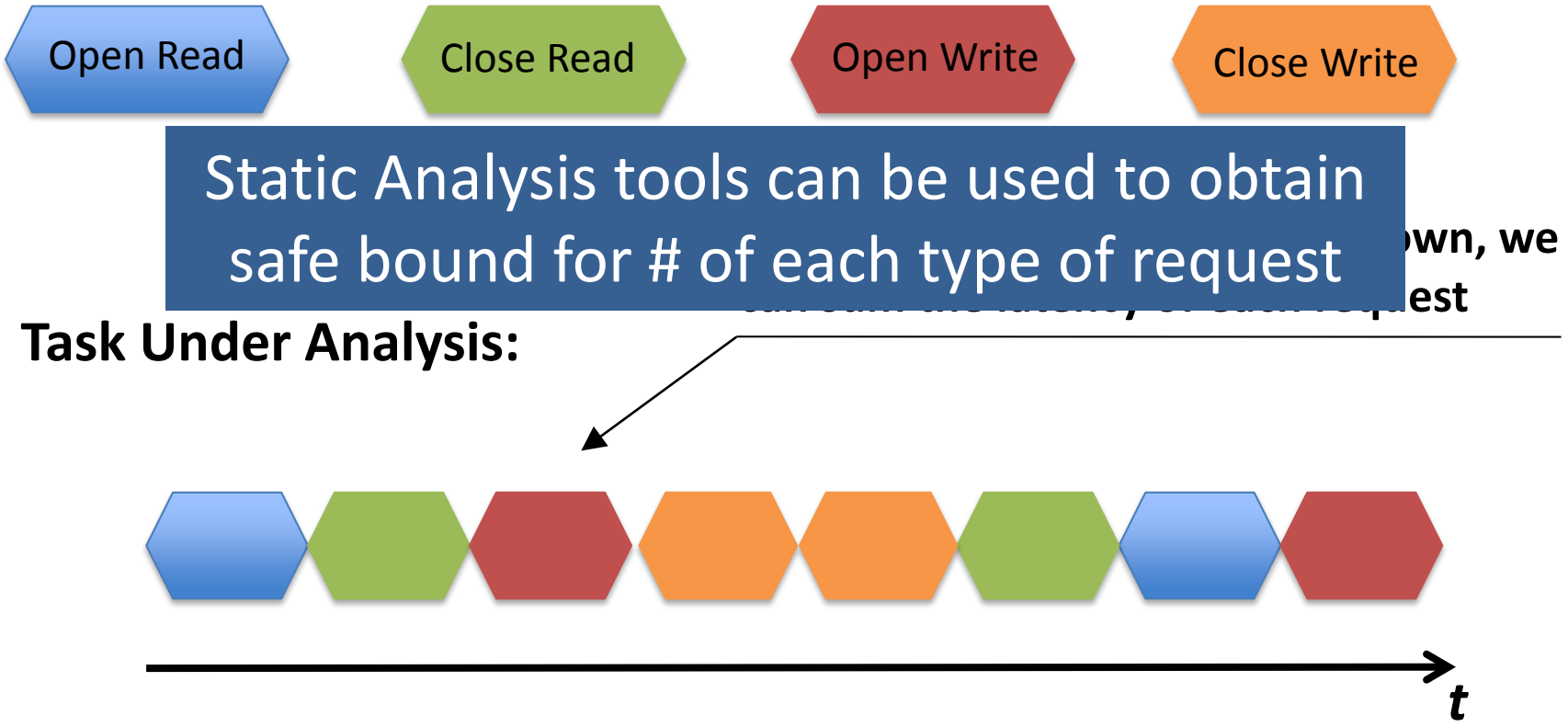
Worst case request order depends on input value, code path, cache state, etc.

rite

If worst case request order is known, we can sum the latency of each request

Task Under Analysis:

Cumulative Latency



Cumulative Latency

This problem can be solved in constant time; see paper for detail

Task Under Analysis:

Which pattern leads to worst case latency?

Outline

1. Background & Related Work

2. Memory Controller Model

3. Worst Case Latency Analysis
   - Single Request Latency
   - Cumulative Latency

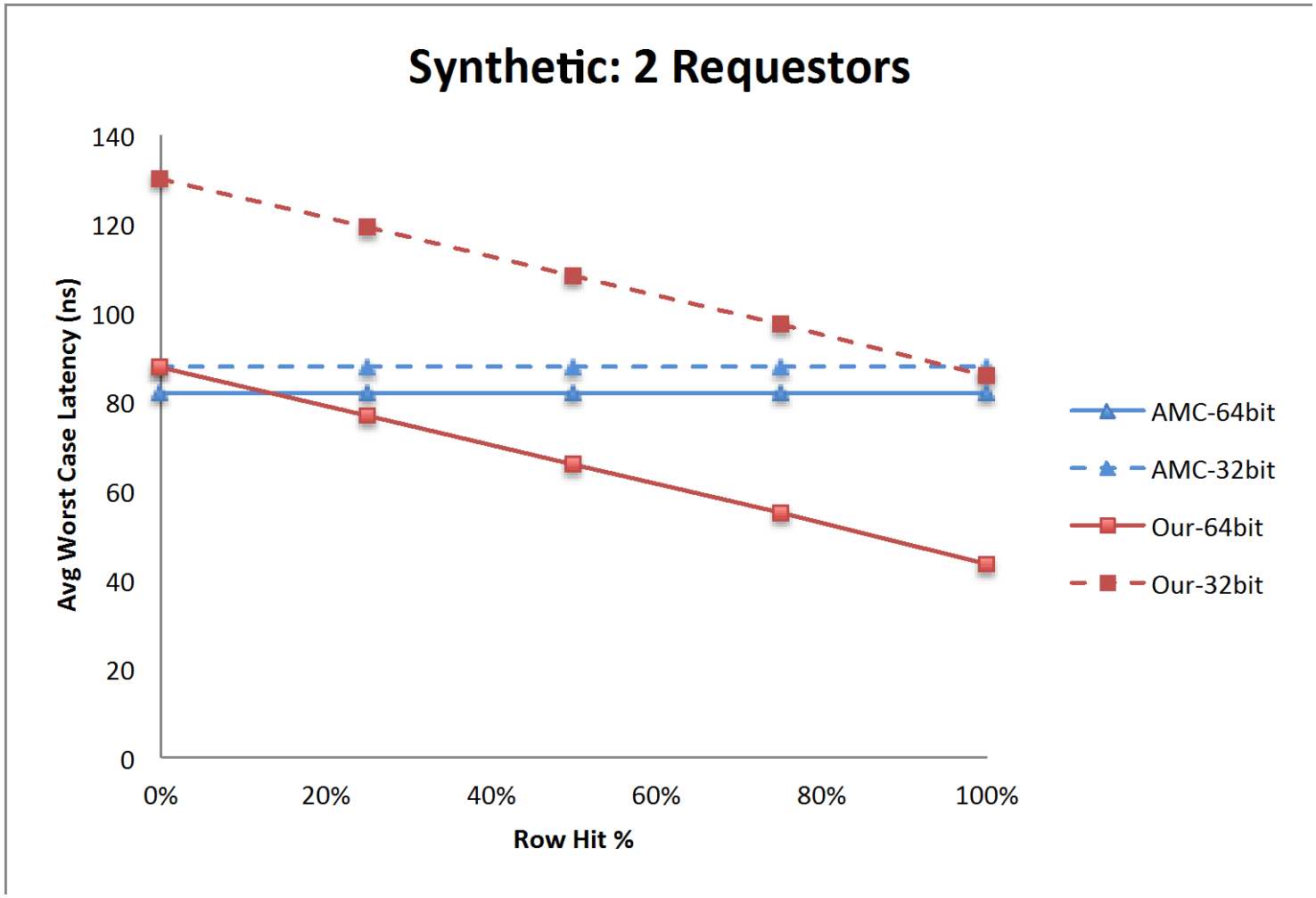
4. Results & Conclusion

Results

- Comparison against Analyzable Memory Controller [1]
  - Since they use fair arbitration (Round Robin) which is similar to our approach
- Synthetic Benchmarks
  - Used to show how worst case latency varies as parameters are changed
- CHStone Benchmarks
  - Memory traces are obtained from gem5 simulator
  - Memory traces are used as input the worst case analysis

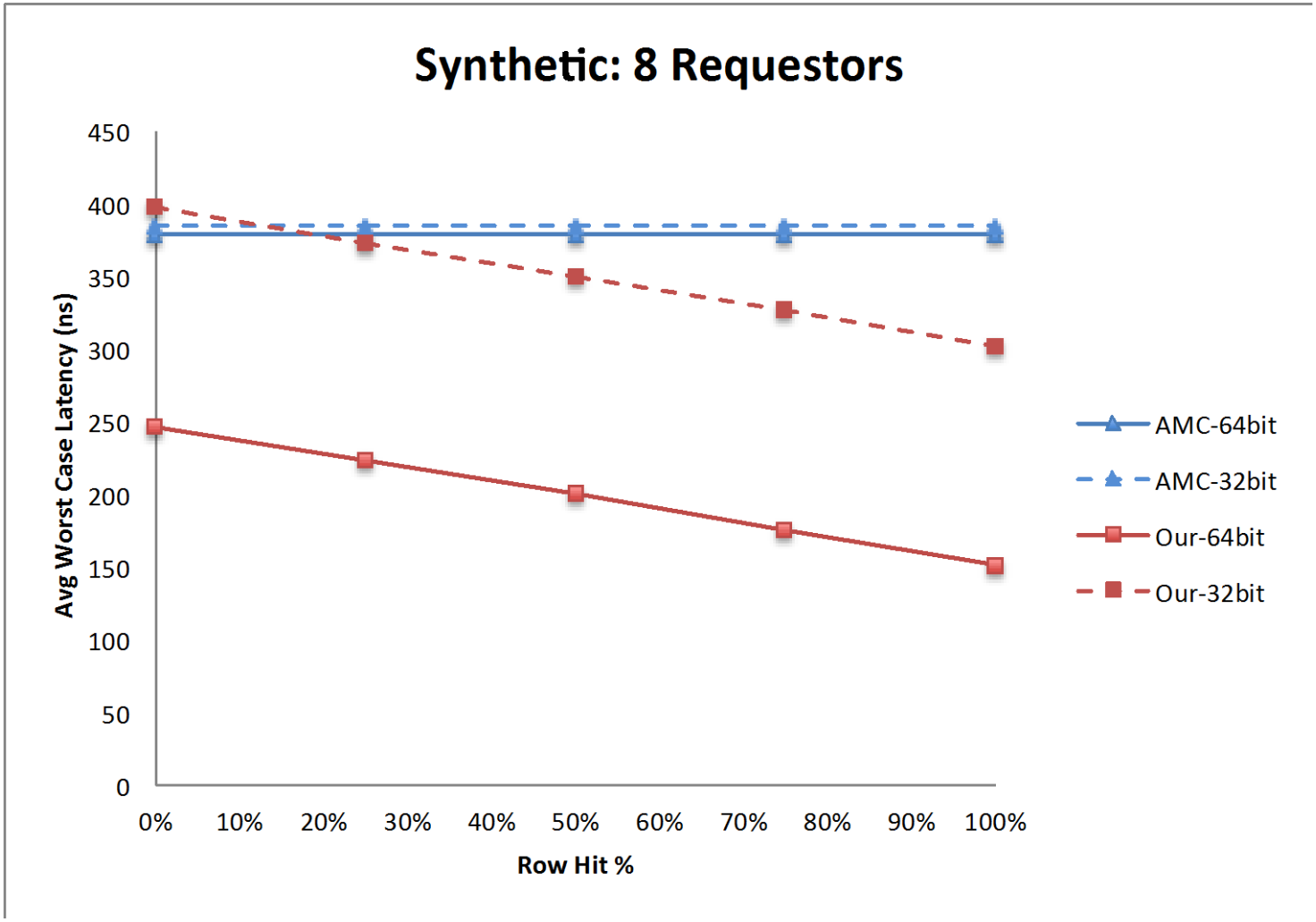
Results

- Synthetic Benchmarks



Results

- Synthetic Benchmarks



Results

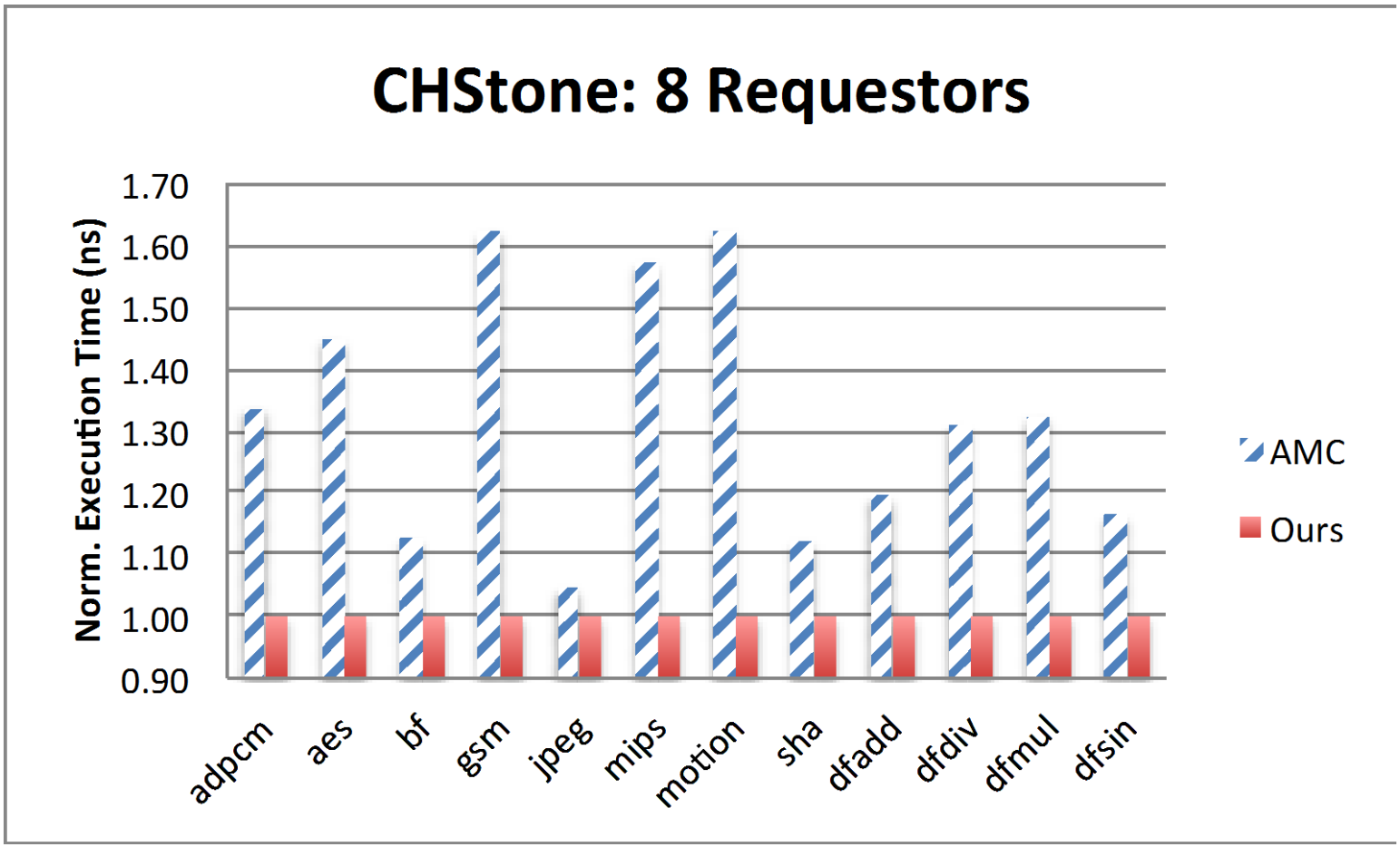
- As memory devices becomes faster, the difference between open and close access is getting larger and therefore close row is becoming too pessimistic

50% Row Hit Ratio, 4 Requestors, 20% Writes

Devices	800D (ns)	1066F (ns)	1333H (ns)	1600K (ns)	1866L (ns)	2133N (ns)	% better
AMC (64 bits)	185	185.27	180.9	178	169.84	163	11.89%
Our (64 bits)	125.2	112.47	104.85	102.18	96.97	92.85	25.84%

Results

- CHStone Benchmarks for 64bits bus



Conclusion

- A novel worst case analysis that takes dynamic state into account
- Open row policy can reduce memory latency as devices are becoming faster
- Private bank scheme is used to eliminate row buffer interference from other requestors

Future Work

- Discussion of shared data
- Bus utilization is still poor due to read/write switching
- Read/Write optimization to reduce latency bound
- Handle Multiple Ranks
- Implementation in hardware

References

[1] M. Paolieri, E. Quiñones, F. Cazorla, and M. Valero, “An Analyzable Memory Controller for Hard Real-Time CMPs,” *Embedded Systems Letters, IEEE*, vol. 1, no. 4, pp. 86–90, 2009.

[2] B. Akesson, K. Goossens, and M. Ringhofer, “Predator: a predictable SDRAM memory controller,” in CODES+ISSS, 2007, pp. 251–256.

[3] S. Goossens, B. Akesson, and K. Goossens, “Conservative Open- page Policy for Mixed Time-Criticality Memory Controllers,” in *DATE*, 2013.

[4] J. Reineke, I. Liu, H. D. Patel, S. Kim, and E. A. Lee, “Pret dram controller: Bank privatization for predictability and temporal isolation,” in CODES+ISSS, 2011, pp. 99–108.