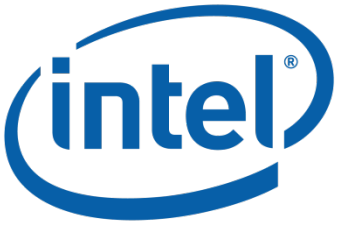
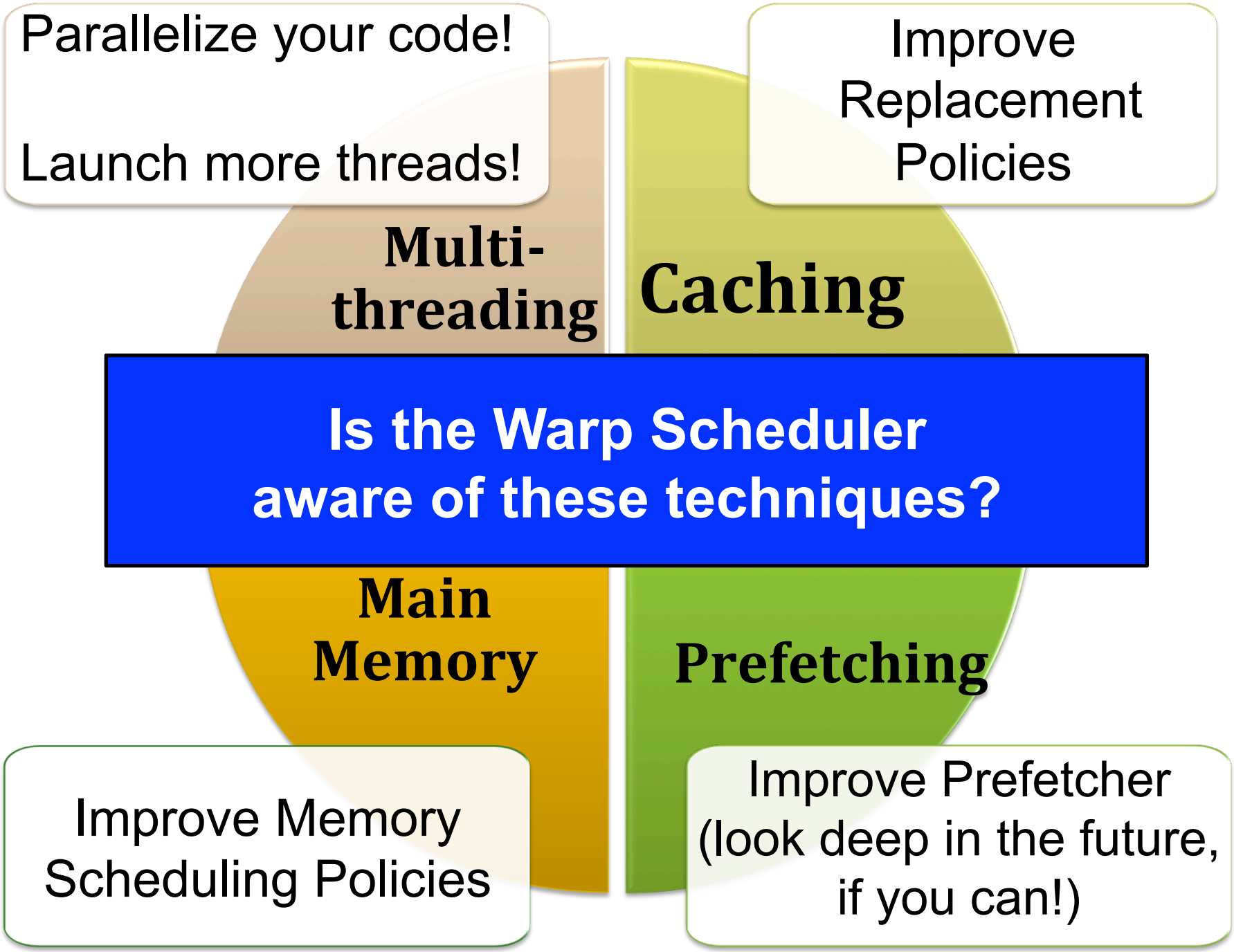
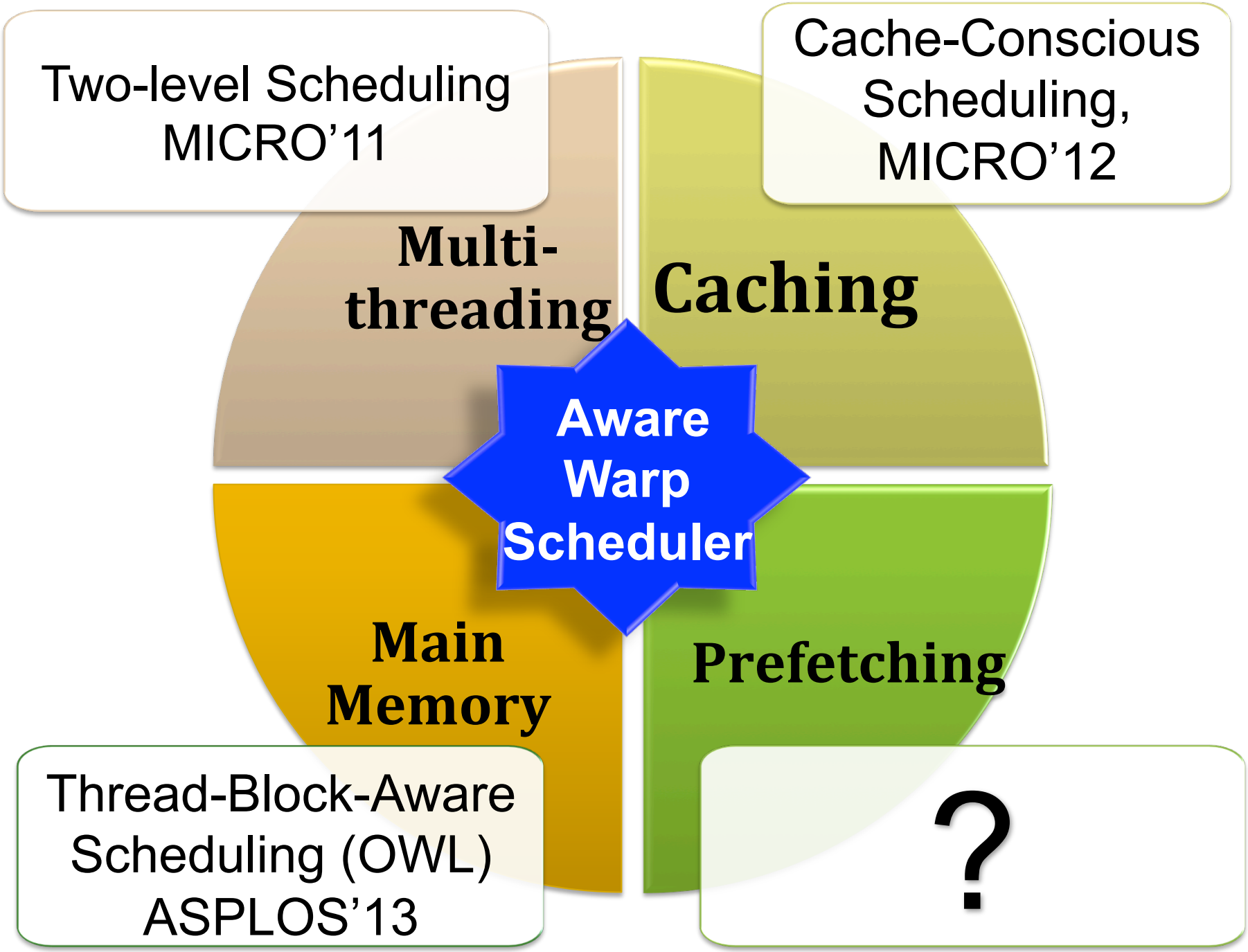
Orchestrated Scheduling and Prefetching for GPGPUs

Adwait Jog, Onur Kayiran, Asit Mishra, Mahmut Kandemir, Onur Mutlu, Ravi Iyer, Chita Das




Parallelize your code!
Launch more threads!
Improve Replacement Policies
Multi-threading
Caching
Is the Warp Scheduler aware of these techniques?
Main Memory
Prefetching
Improve Memory Scheduling Policies
Improve Prefetcher (look deep in the future, if you can!)


Two-level Scheduling MICRO'11
Cache-Conscious Scheduling, MICRO'12
Multi-threading
Caching
Aware Warp Scheduler
Main Memory
Prefetching
Thread-Block-Aware Scheduling (OWL) ASPLOS'13
?

Our Proposal

- Prefetch Aware Warp Scheduler

- Goals:
  - Make a Simple prefetcher more Capable
  - Improve system performance by orchestrating scheduling and prefetching mechanisms

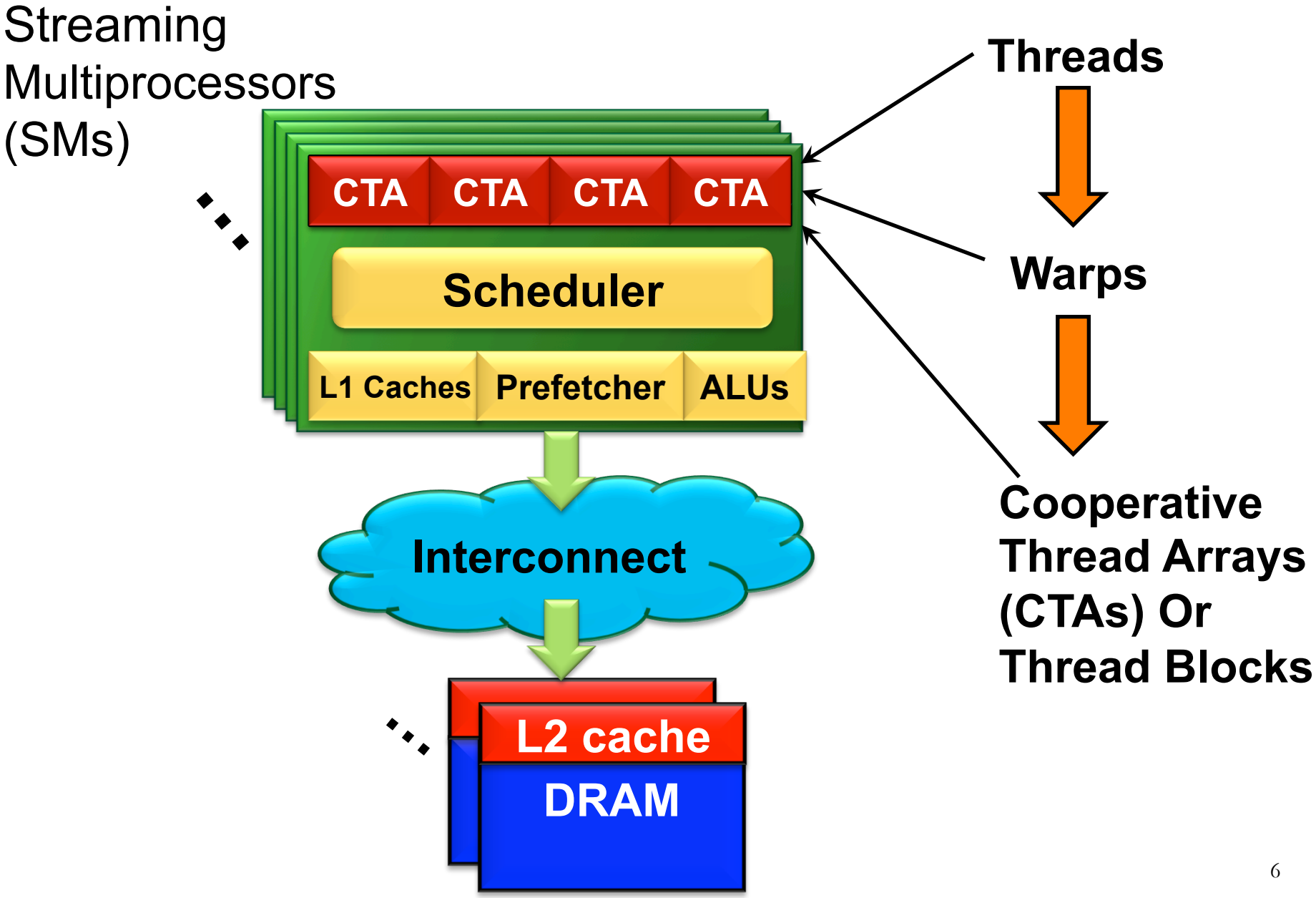
- 25% average IPC improvement over
  - Prefetching + Conventional Warp Scheduling Policy

- 7% average IPC improvement over
  - Prefetching + Best Previous Warp Scheduling Policy

Outline

- Proposal
- Background and Motivation
- Prefetch-aware Scheduling
- Evaluation
- Conclusions

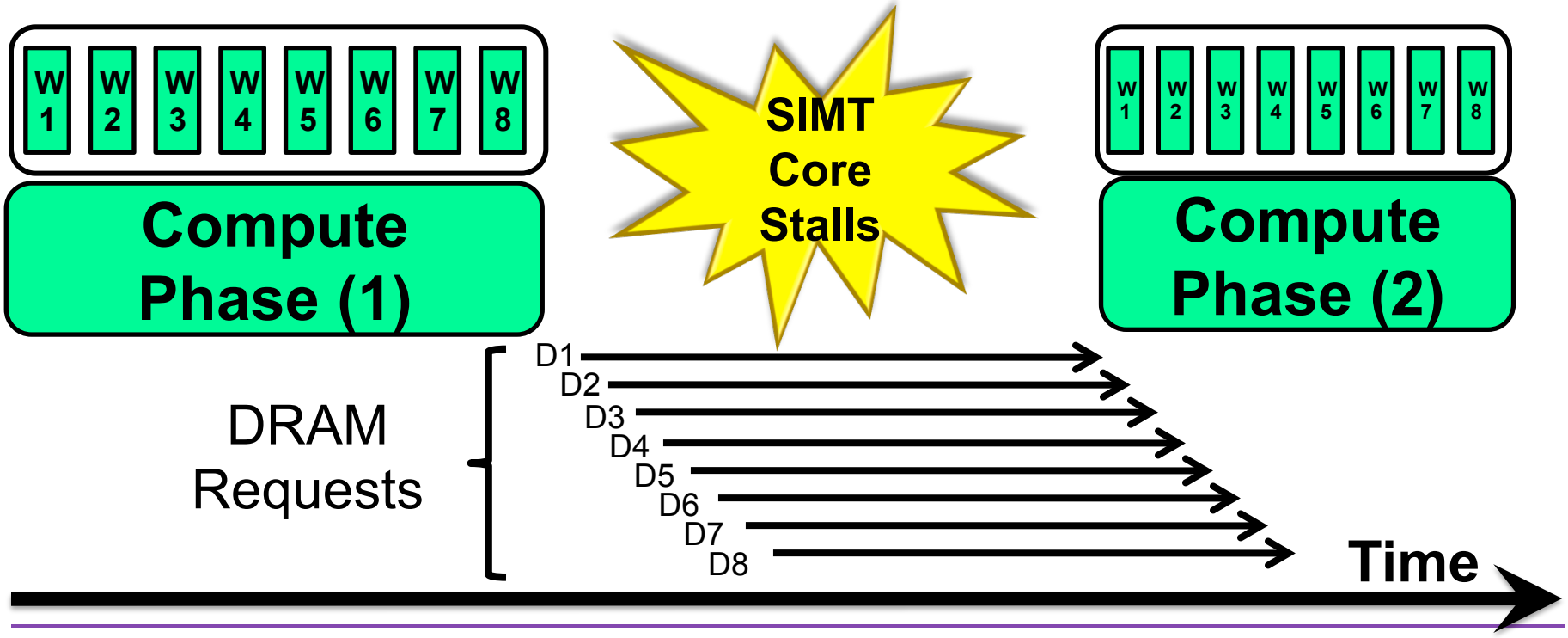
High-Level View of a GPU

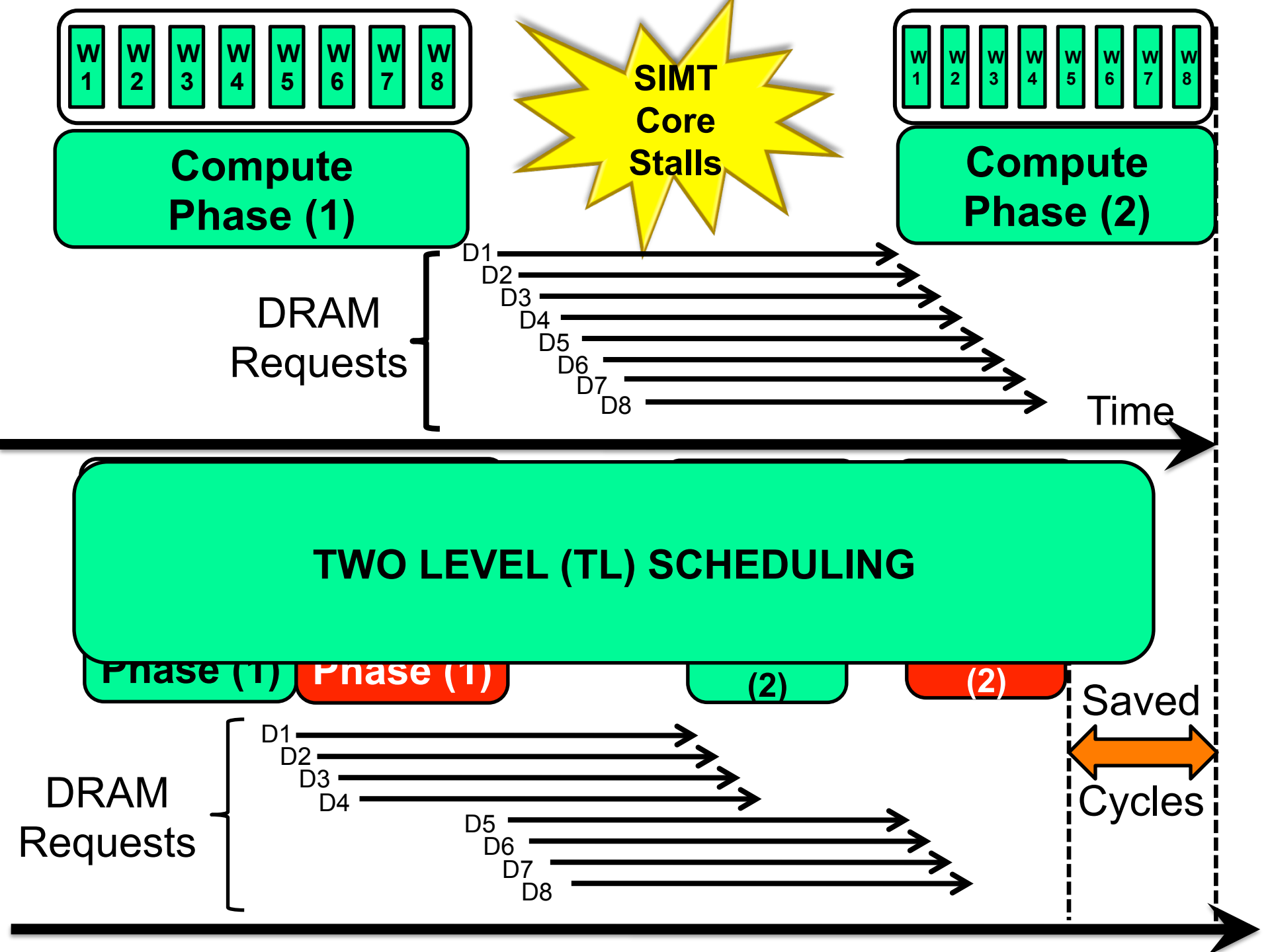


Warp Scheduling Policy

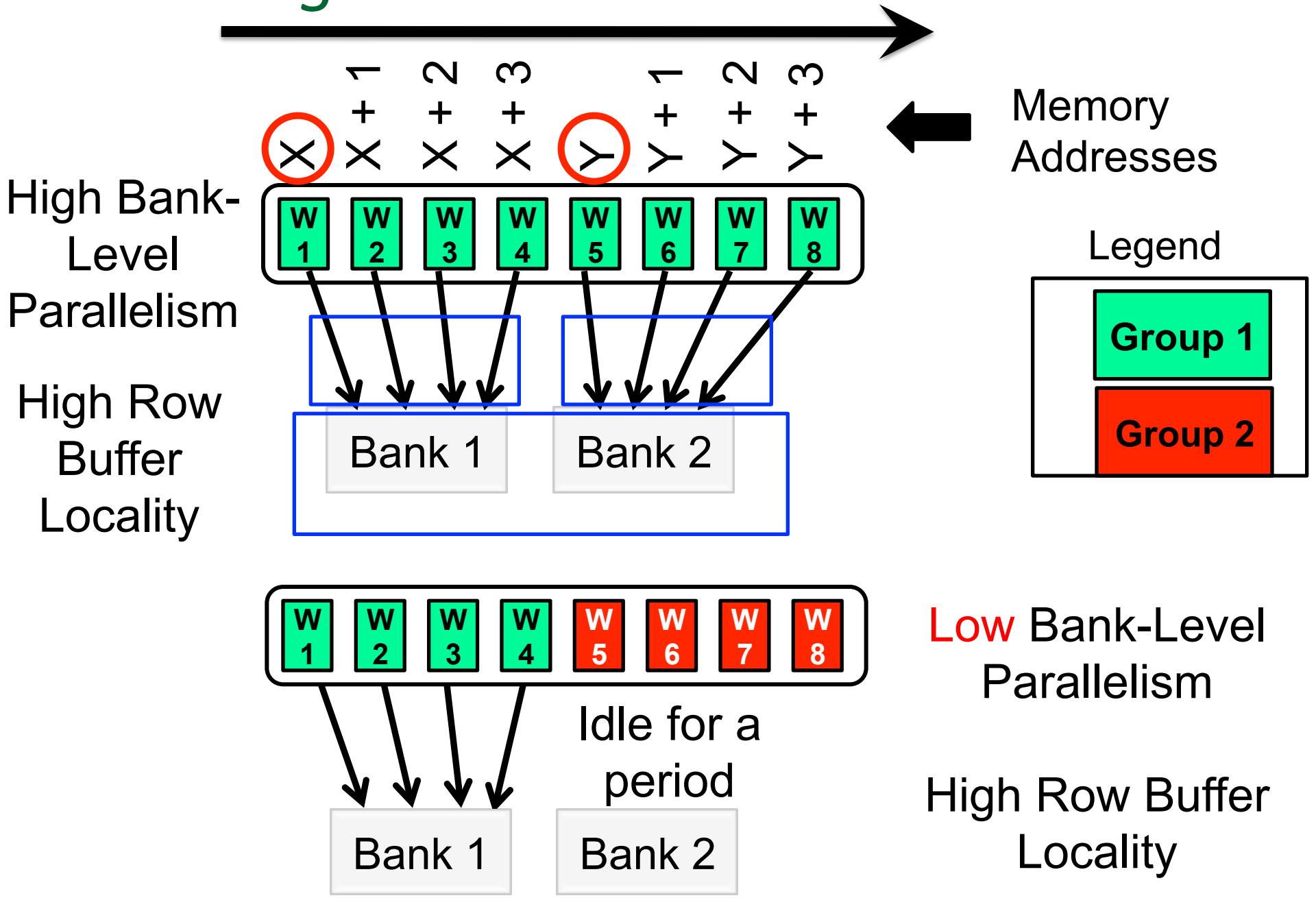
- Equal scheduling priority
  - Round-Robin (RR) execution

- Problem: Warps stall roughly at the same time




W 1
W 2
W 3
W 4
W 5
W 6
W 7
W 8
SIMT Core Stalls
Compute Phase (1)
Compute Phase (2)
DRAM Requests
D1
D2
D3
D4
D5
D6
D7
D8
Time
TWO LEVEL (TL) SCHEDULING
Phase (1)
Phase (1)
(2)
(2)
Saved
Cycles
DRAM Requests
D1
D2
D3
D4
D5
D6
D7
D8

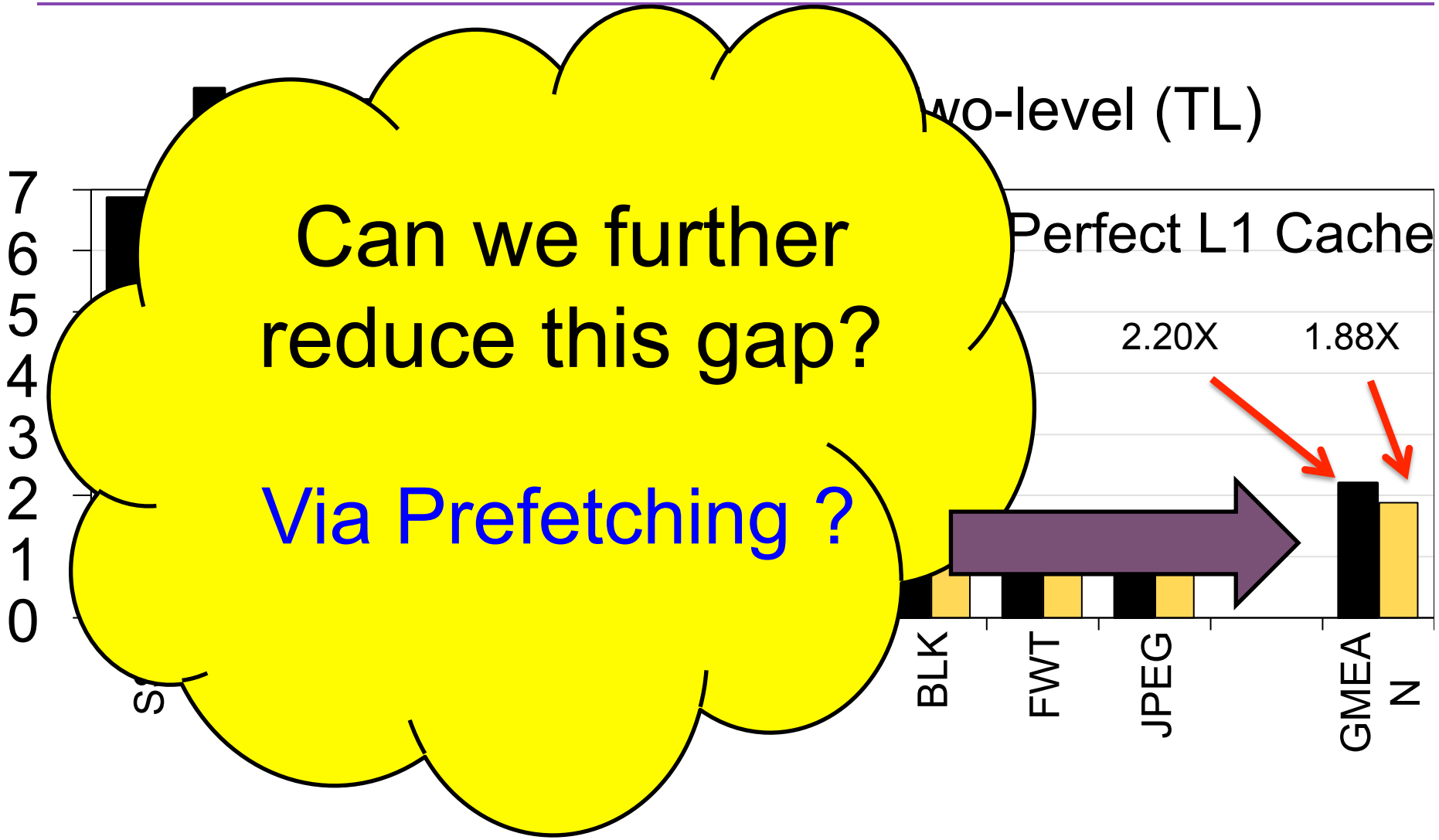
Accessing DRAM …



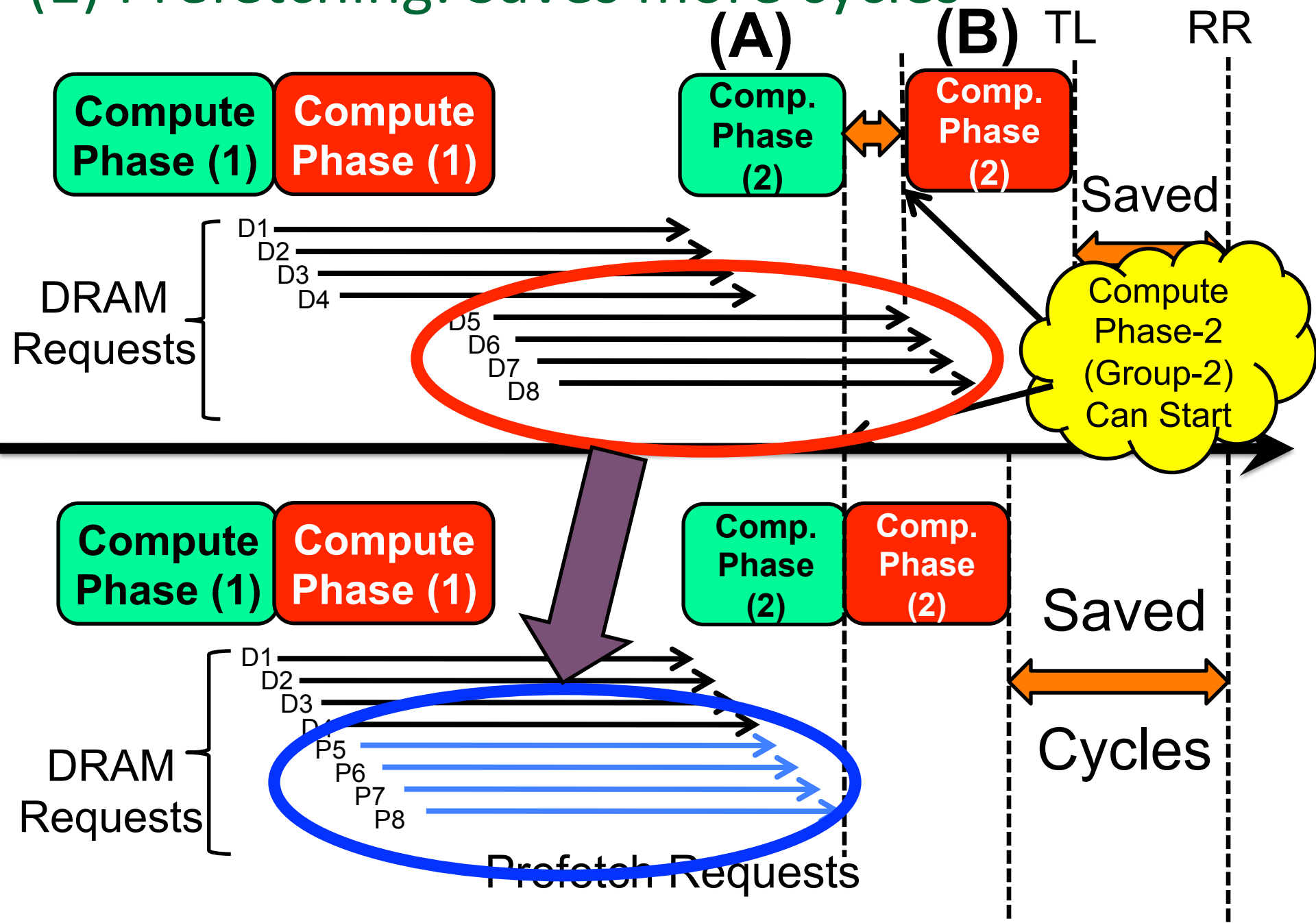
Warp Scheduler Perspective (Summary)

Warp Scheduler	Forms Multiple Warp Groups?	DRAM Bandwidth Utilization	
		Bank Level Parallelism	Row Buffer Locality
Round-Robin (RR)	✖	✔	✔
Two-Level (TL)	✔	✖	✔

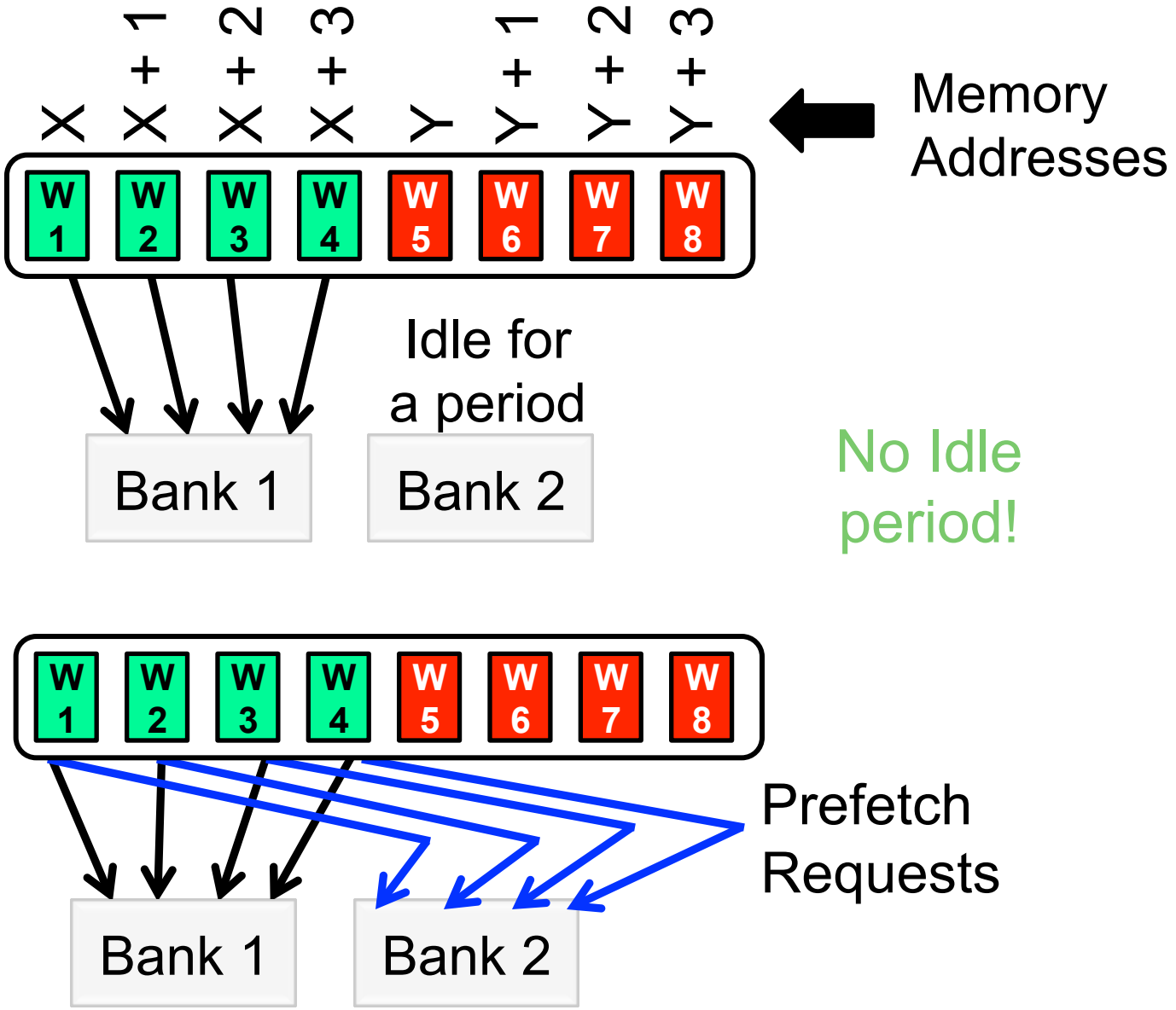
Evaluating RR and TL schedulers



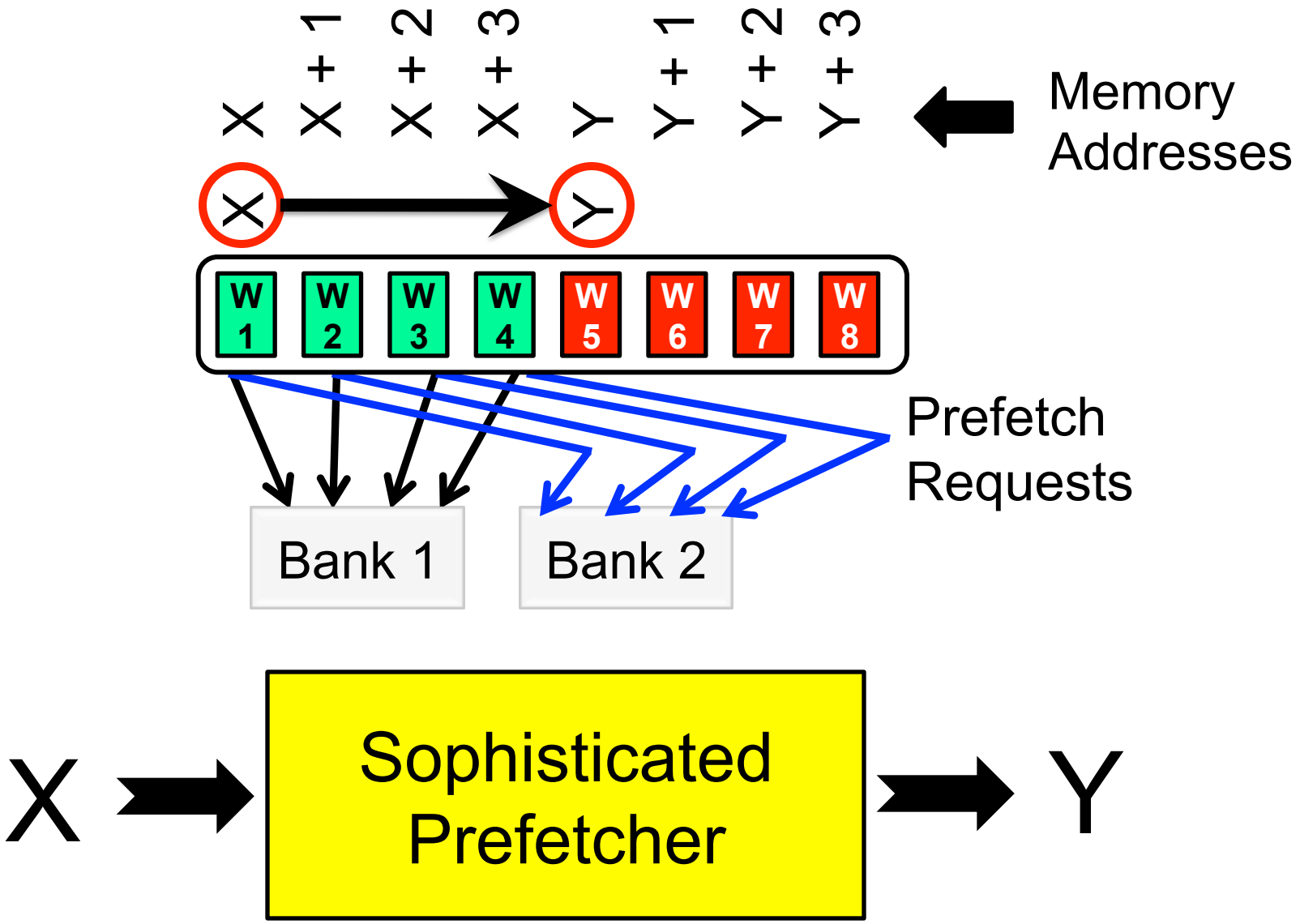
(1) Prefetching: Saves more cycles



(2) Prefetching: Improve DRAM Bandwidth Utilization



Challenge: Designing a Prefetcher



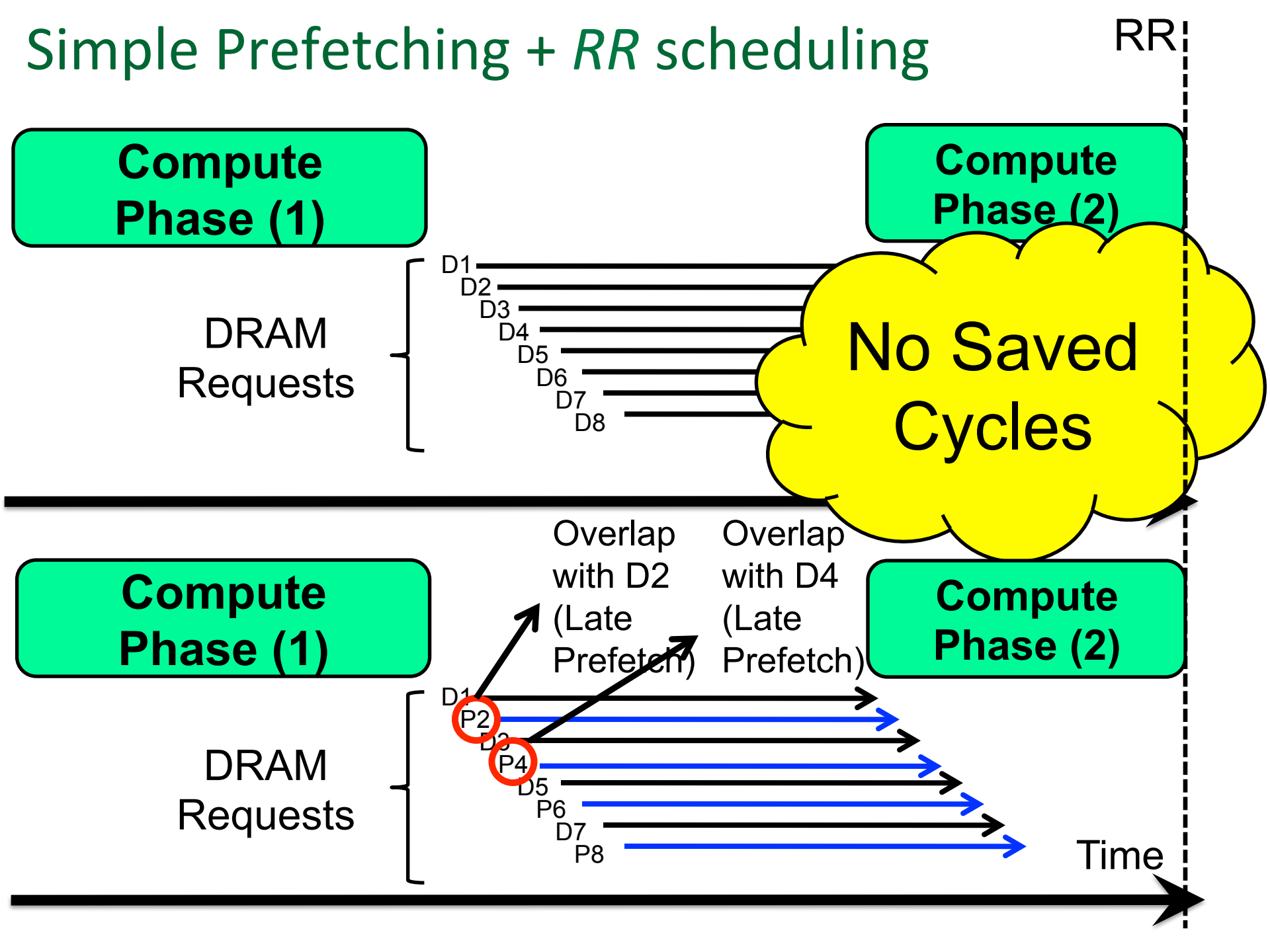
Our Goal

- Keep the prefetcher simple, yet get the performance benefits of a sophisticated prefetcher.

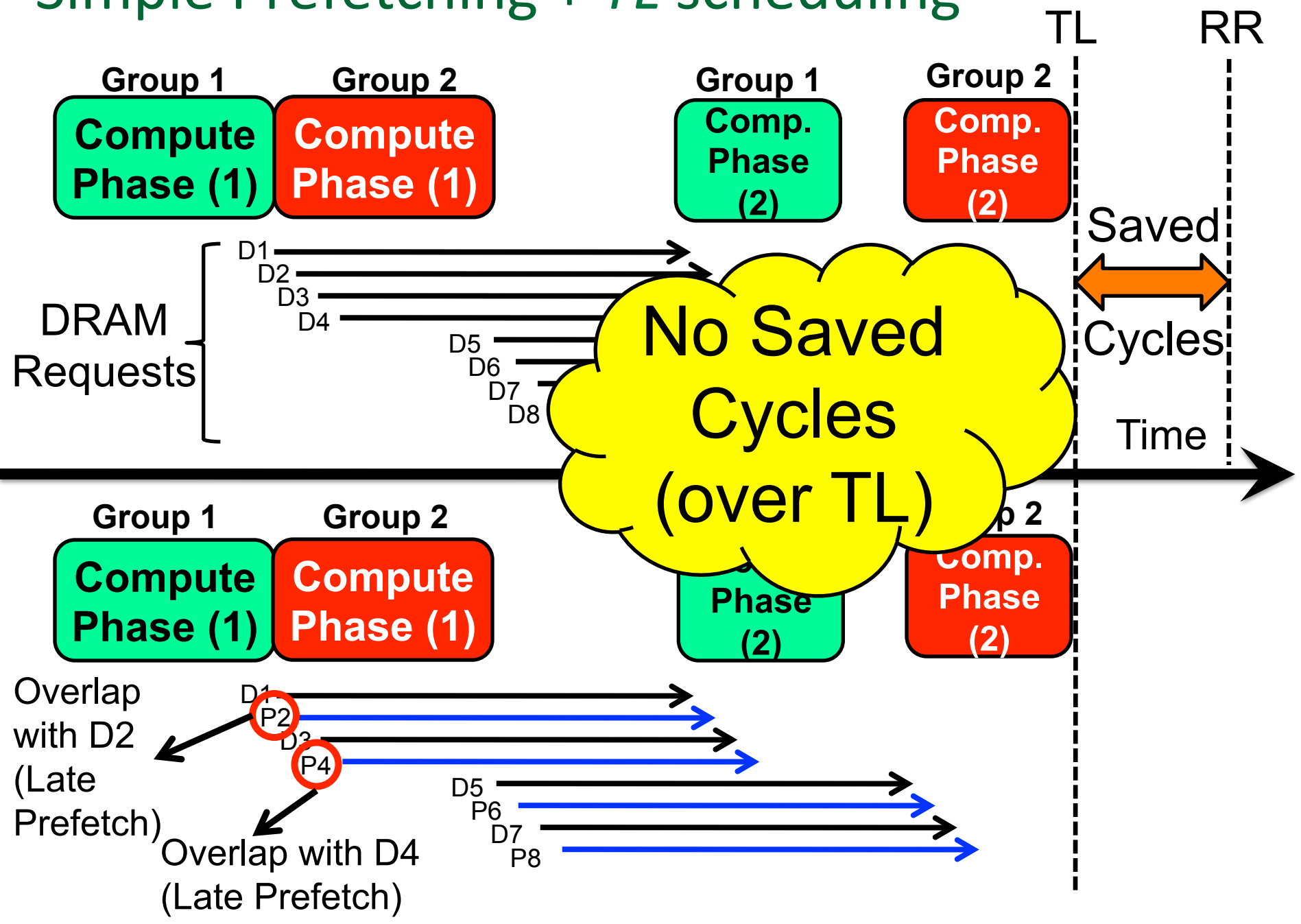
To this end, we will design a prefetch-aware warp scheduling policy Why?

A simple prefetching does not improve performance with existing scheduling policies.

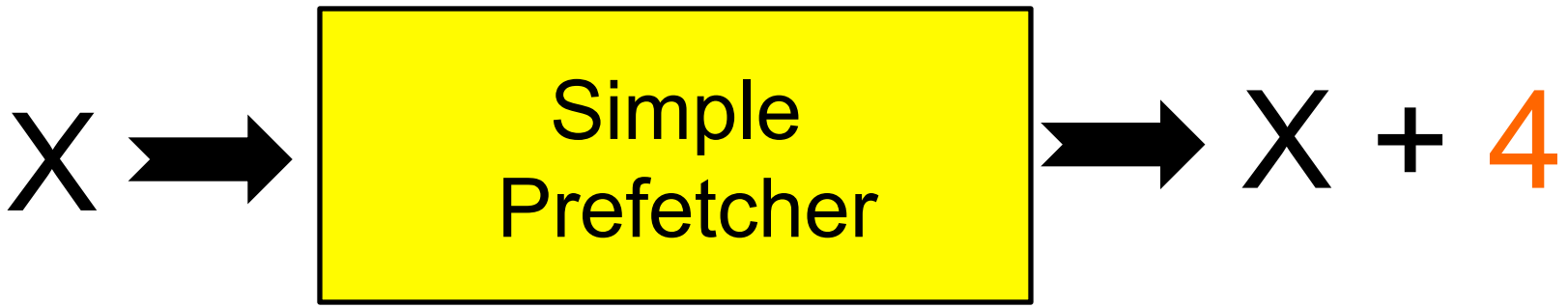
Simple Prefetching + *RR* scheduling



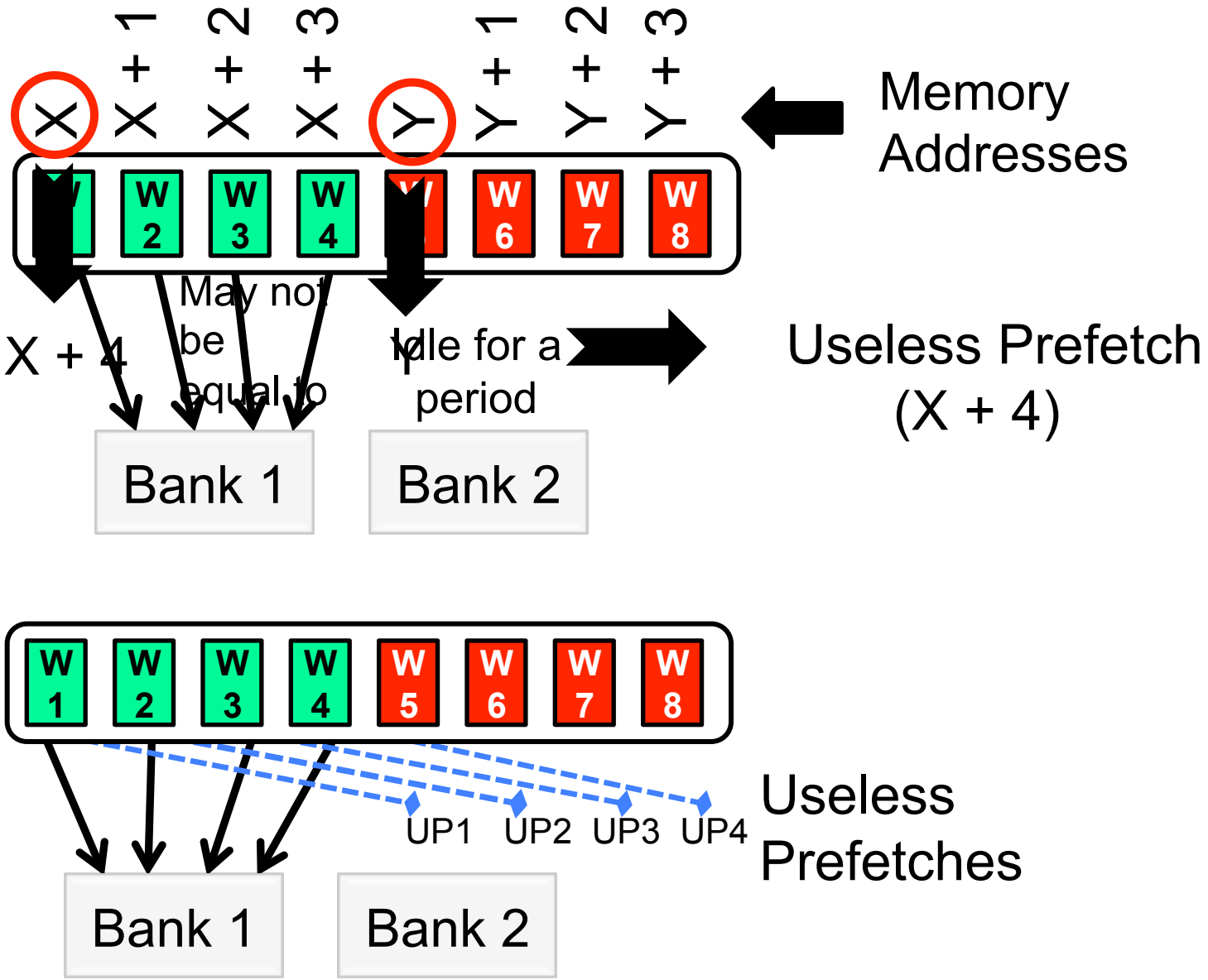
Simple Prefetching + *TL* scheduling



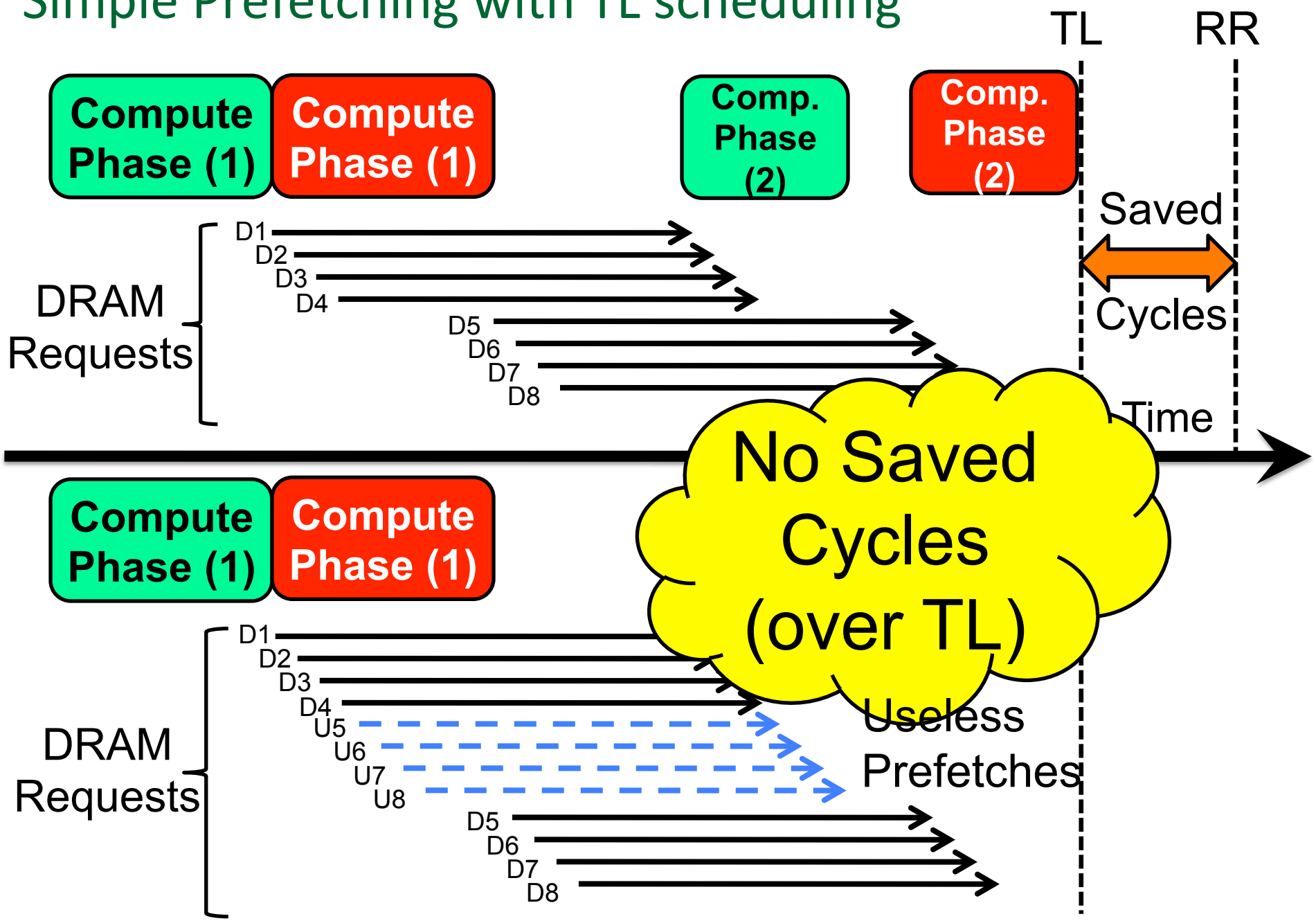
Let's Try…



Simple Prefetching with TL scheduling



Simple Prefetching with TL scheduling



Warp Scheduler Perspective (Summary)

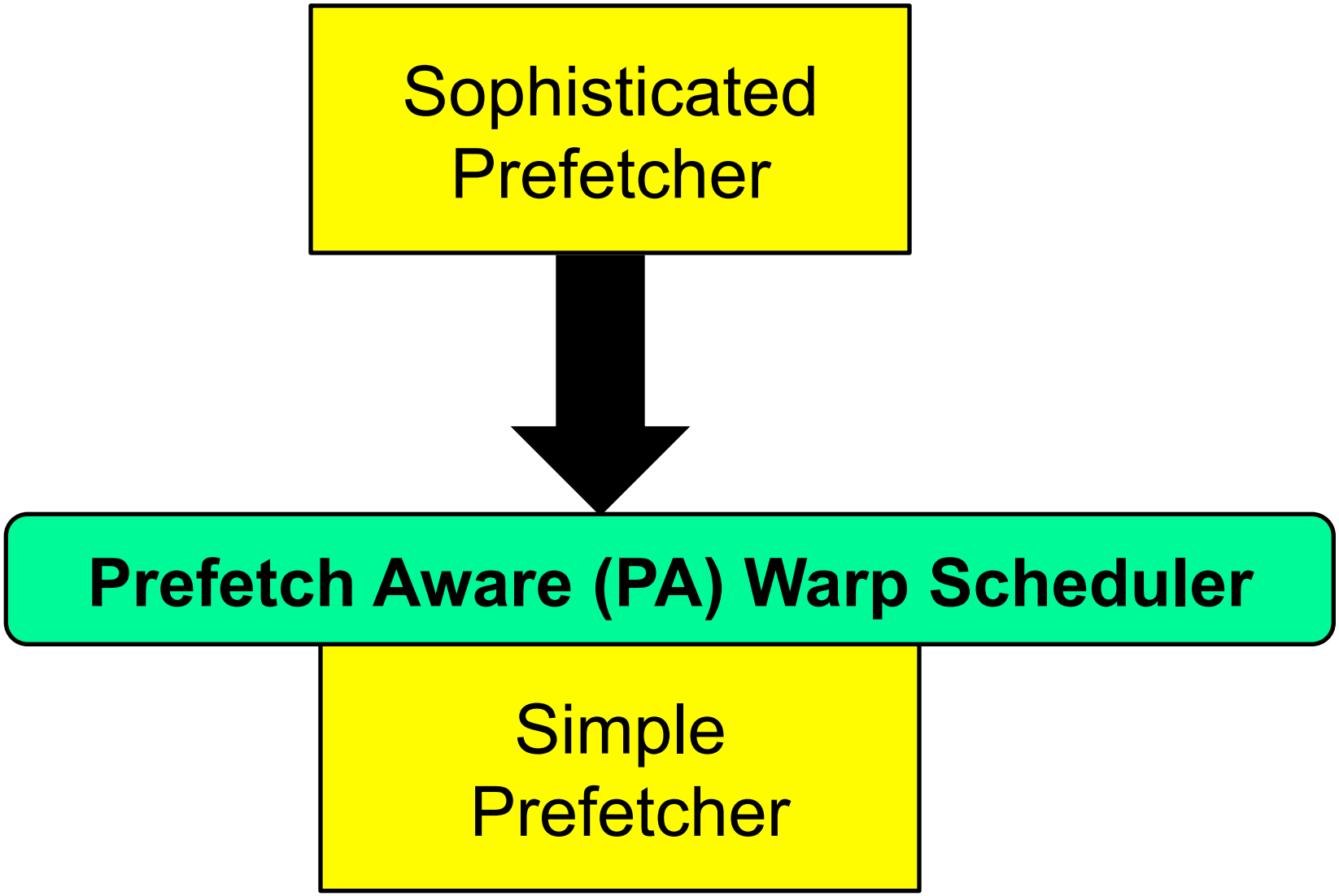
Warp Scheduler	Forms Multiple Warp Groups?	Simple Prefetcher Friendly?	DRAM Bandwidth Utilization	
			Bank Level Parallelism	Row Buffer Locality
Round-Robin (RR)	✖	✖	✔	✔
Two-Level (TL)	✔	✖	✖	✔

Our Goal

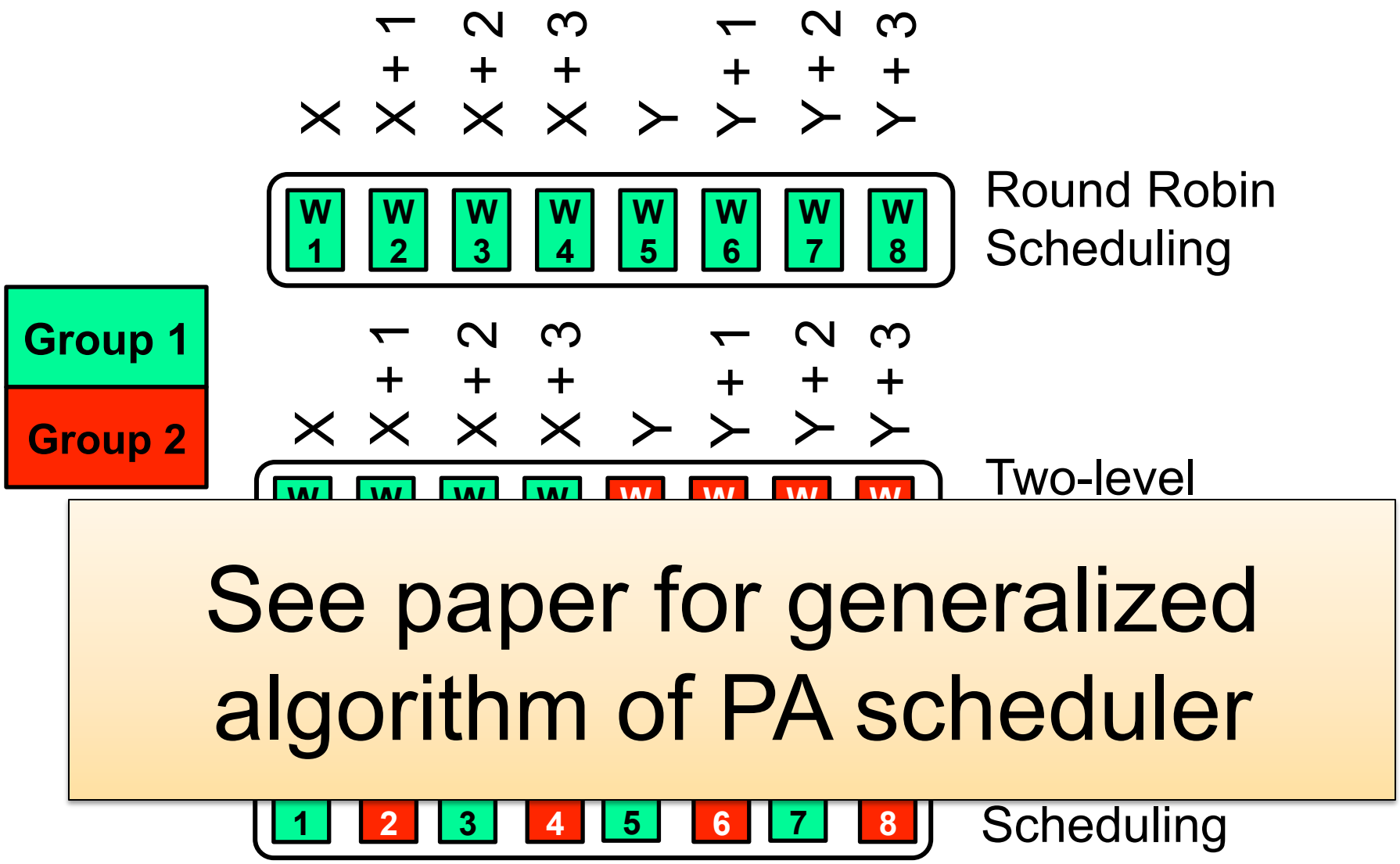
- Keep the prefetcher simple, yet get the performance benefits of a sophisticated prefetcher.

To this end, we will design a prefetch-aware warp scheduling policy

A simple prefetching does not improve performance with existing scheduling policies.


Sophisticated Prefetcher
Prefetch Aware (PA) Warp Scheduler
Simple Prefetcher

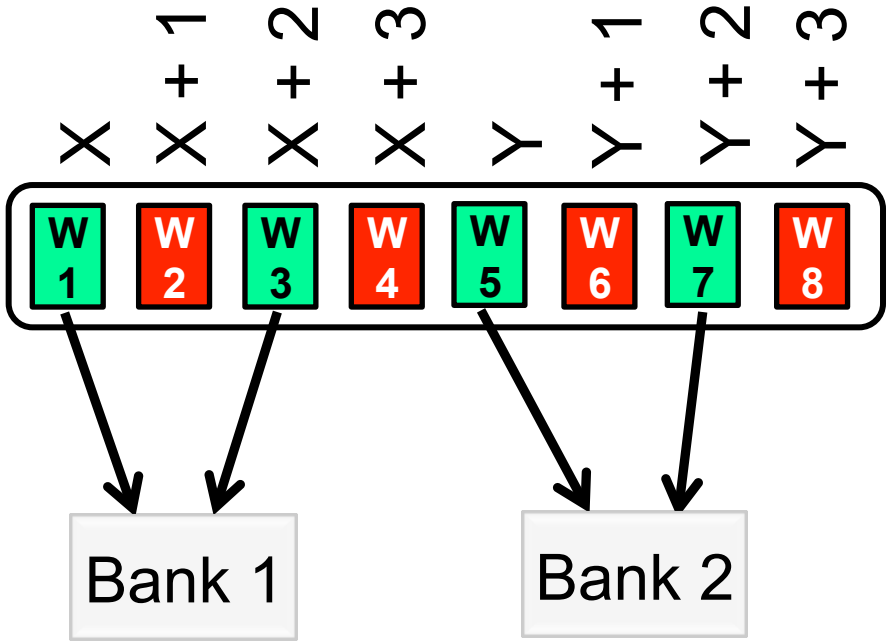
Prefetch-aware (PA) warp scheduling



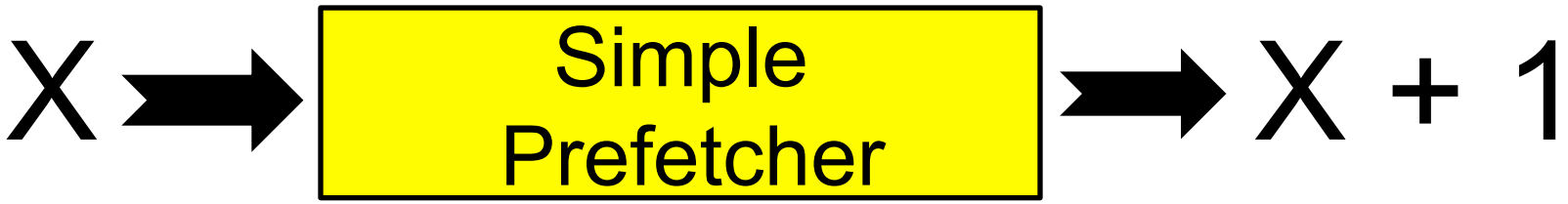
See paper for generalized algorithm of PA scheduler

Non-consecutive warps are associated with one group

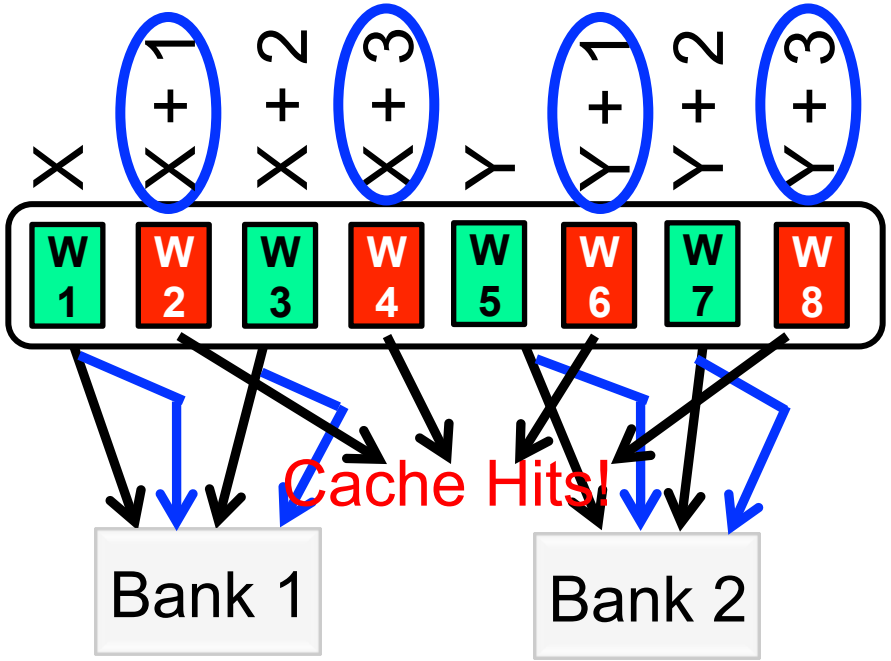
Simple Prefetching with PA scheduling

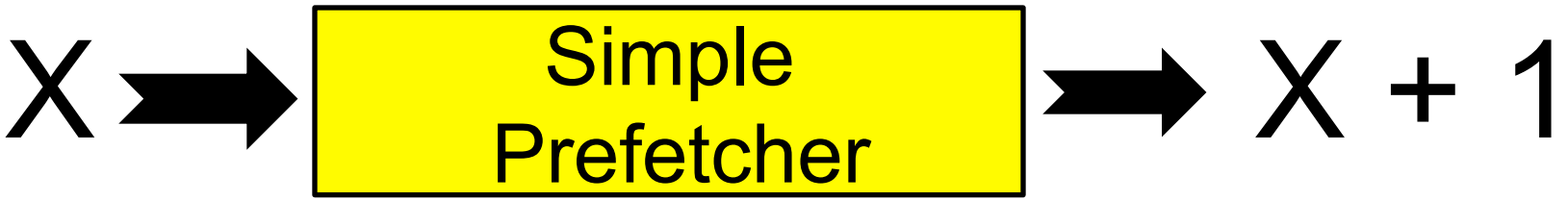


Reasoning of non-consecutive warp grouping is that groups can (simple) prefetch for each other (green warps can prefetch for red warps using simple prefetcher)

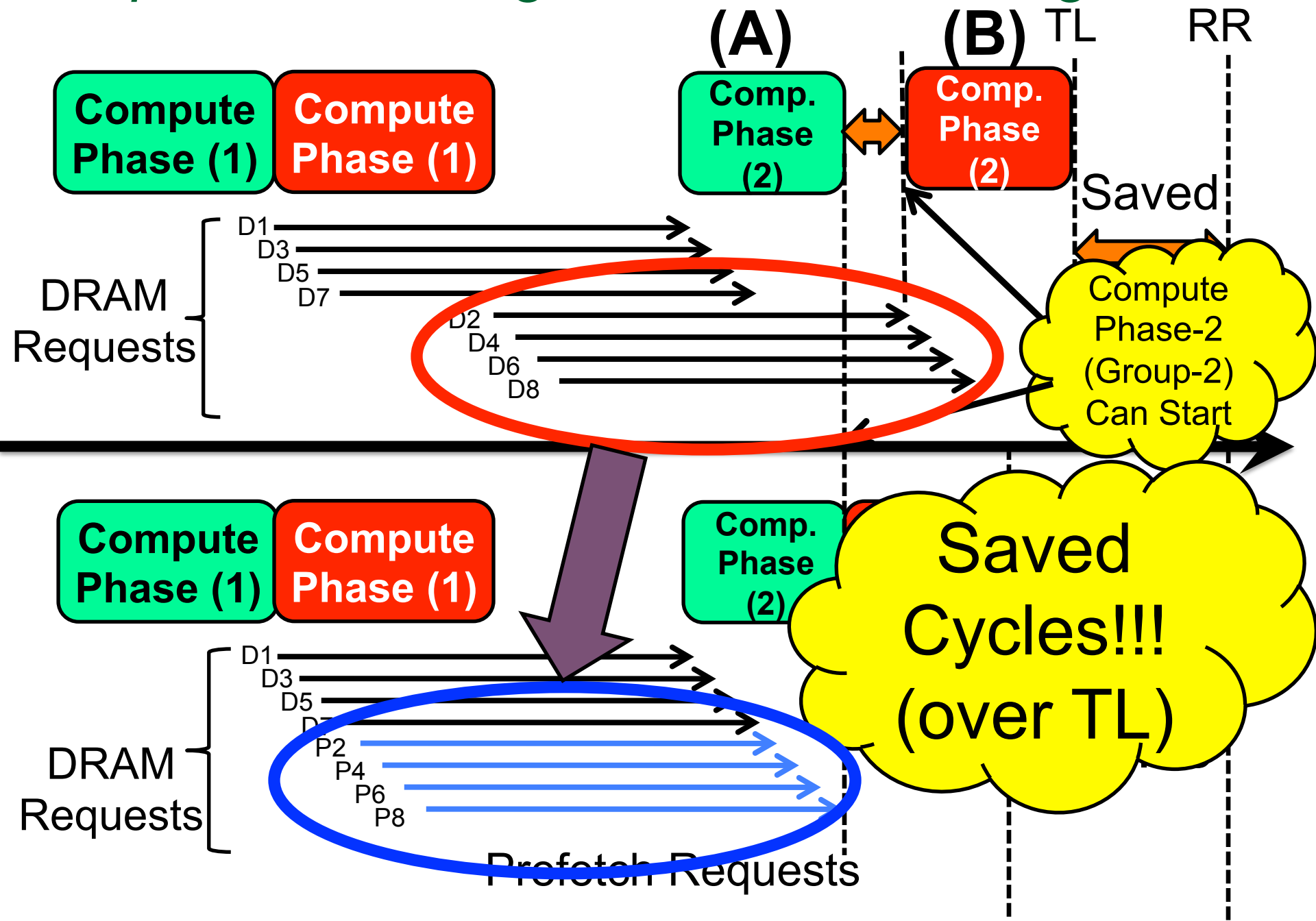


Simple Prefetching with PA scheduling

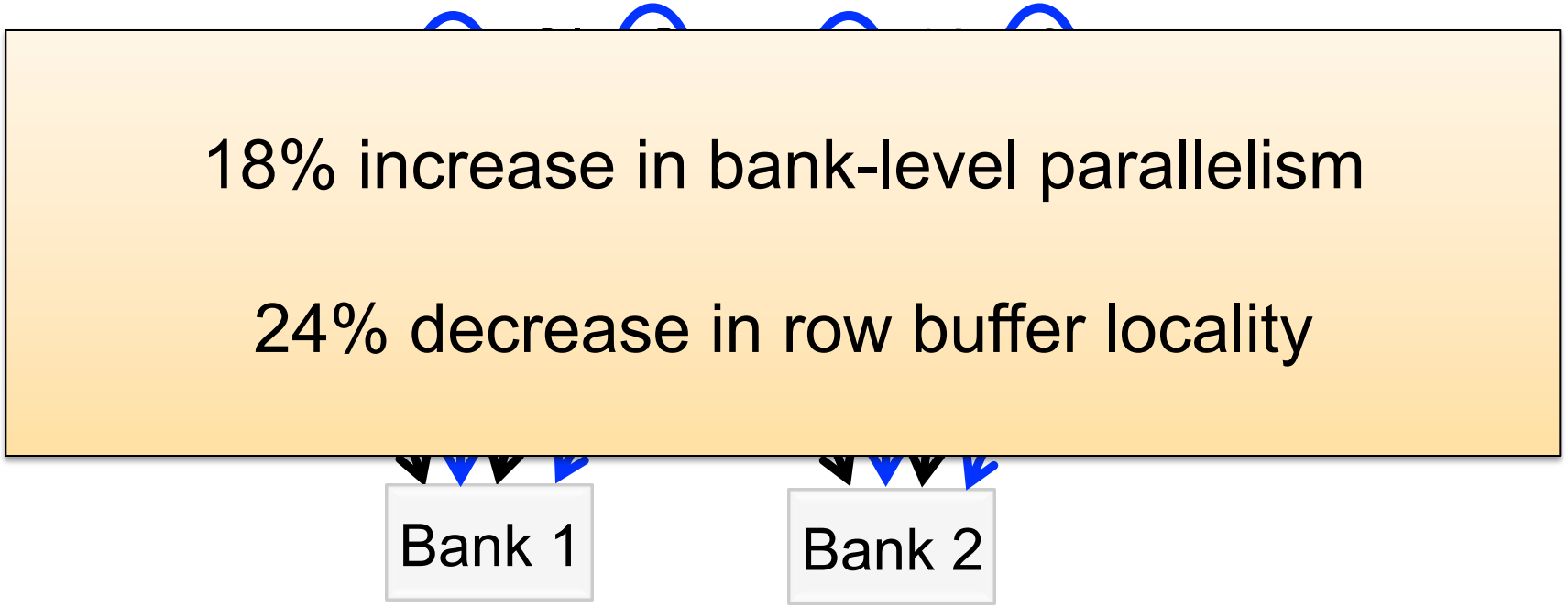




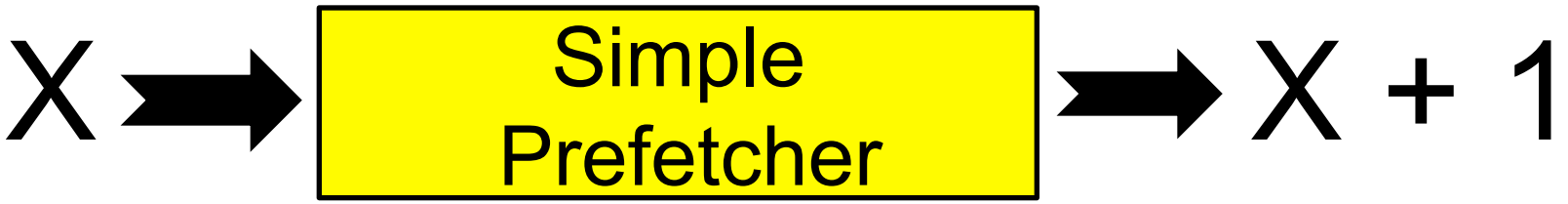
Simple Prefetching with PA scheduling



DRAM Bandwidth Utilization



High Bank-Level Parallelism
High Row Buffer Locality



Warp Scheduler Perspective (Summary)

Warp Scheduler	Forms Multiple Warp Groups?	Simple Prefetcher Friendly?	DRAM Bandwidth Utilization	
			Bank Level Parallelism	Row Buffer Locality
Round-Robin (RR)	✖	✖	✔	✔
Two-Level (TL)	✔	✖	✖	✔
Prefetch-Aware (PA)	✔	✔	✔	✔ (with prefetching)

Outline

- Proposal
- Background and Motivation
- Prefetch-aware Scheduling
- Evaluation
- Conclusions

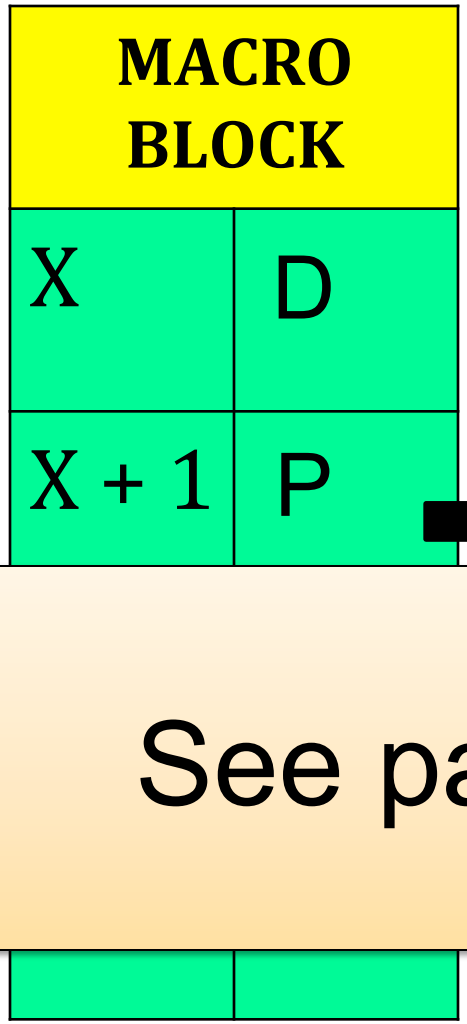
Evaluation Methodology

- Evaluated on GPGPU-Sim, a cycle accurate GPU simulator

- Baseline Architecture
  - 30 SMs, 8 memory controllers, crossbar connected
  - 1300MHz, SIMT Width = 8, Max. 1024 threads/core
  - 32 KB L1 data cache, 8 KB Texture and Constant Caches
  - L1 Data Cache Prefetcher, GDDR3@1100MHz

- Applications Chosen from:
  - Mapreduce Applications
  - Rodinia – Heterogeneous Applications
  - Parboil – Throughput Computing Focused Applications
  - NVIDIA CUDA SDK – GPGPU Applications

Spatial Locality Detector based Prefetching



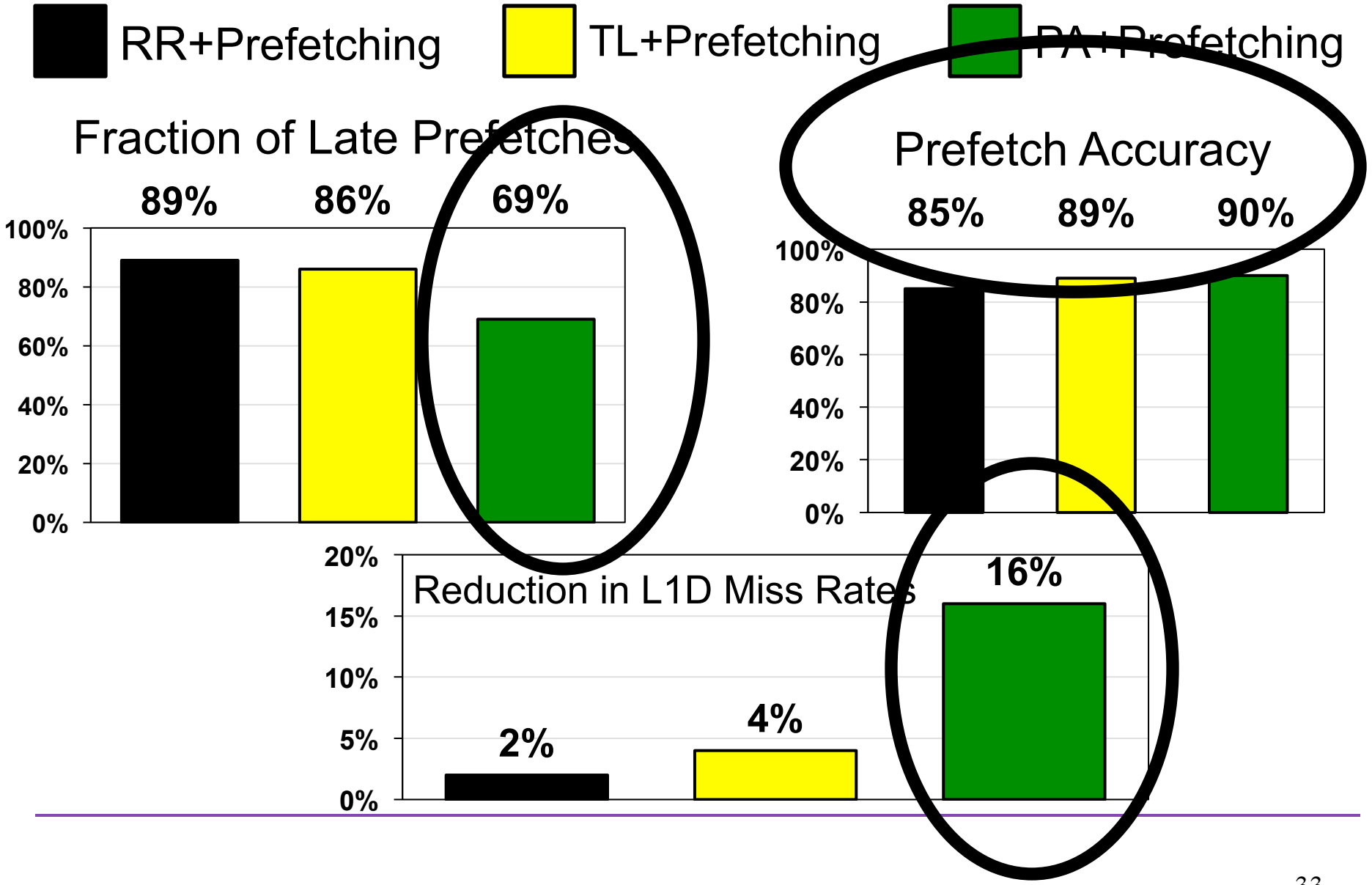
Prefetch:- Not accessed (demanded) Cache Lines

Prefetch-aware Scheduler

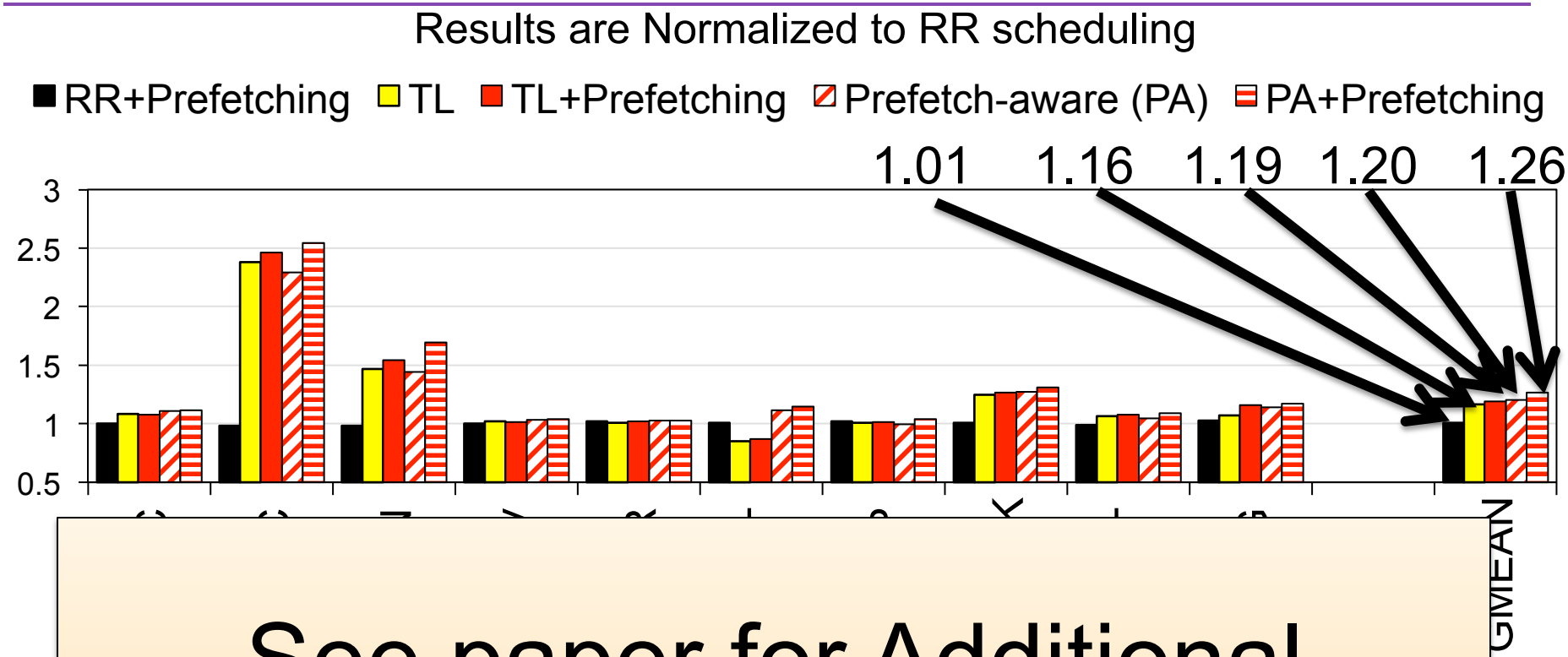
See paper for more details

D = Demand, P = Prefetch

Improving Prefetching Effectiveness



Performance Evaluation



See paper for Additional Results

Conclusions

- Existing warp schedulers in GPGPUs cannot take advantage of simple prefetchers
  - Consecutive warps have good spatial locality, and can prefetch well for each other
  - But, existing schedulers schedule consecutive warps closeby in time → prefetches are too late
- We proposed prefetch-aware (PA) warp scheduling
  - Key idea: group consecutive warps into different groups
  - Enables a simple prefetcher to be timely since warps in different groups are scheduled at separate times
- Evaluations show that PA warp scheduling improves performance over combinations of conventional (RR) and the best previous (TL) warp scheduling and prefetching policies
  - Better orchestrates warp scheduling and prefetching decisions

THANKS!

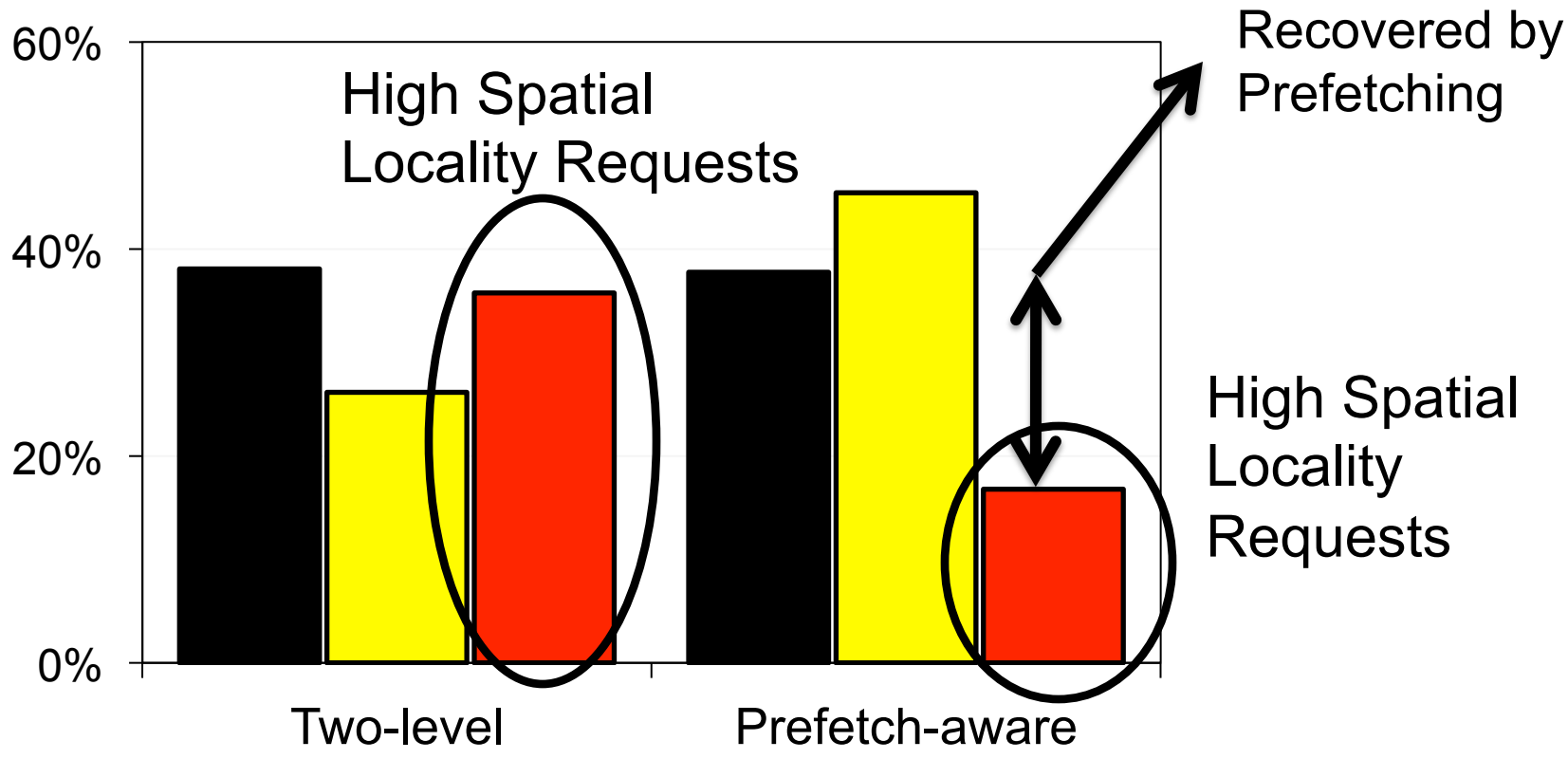
QUESTIONS?

BACKUP

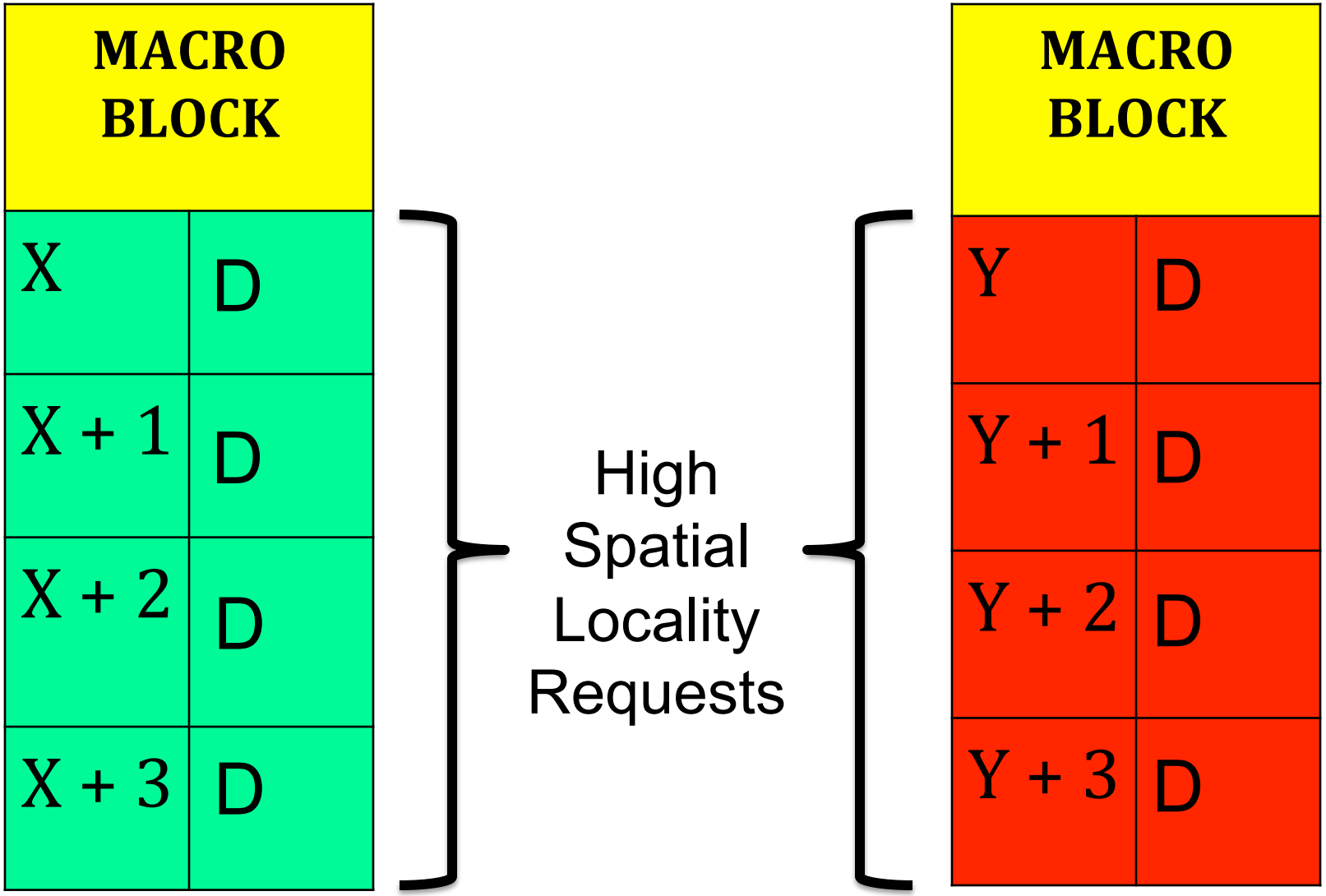
Effect of Prefetch-aware Scheduling

Percentage of DRAM requests (averaged over group) with:

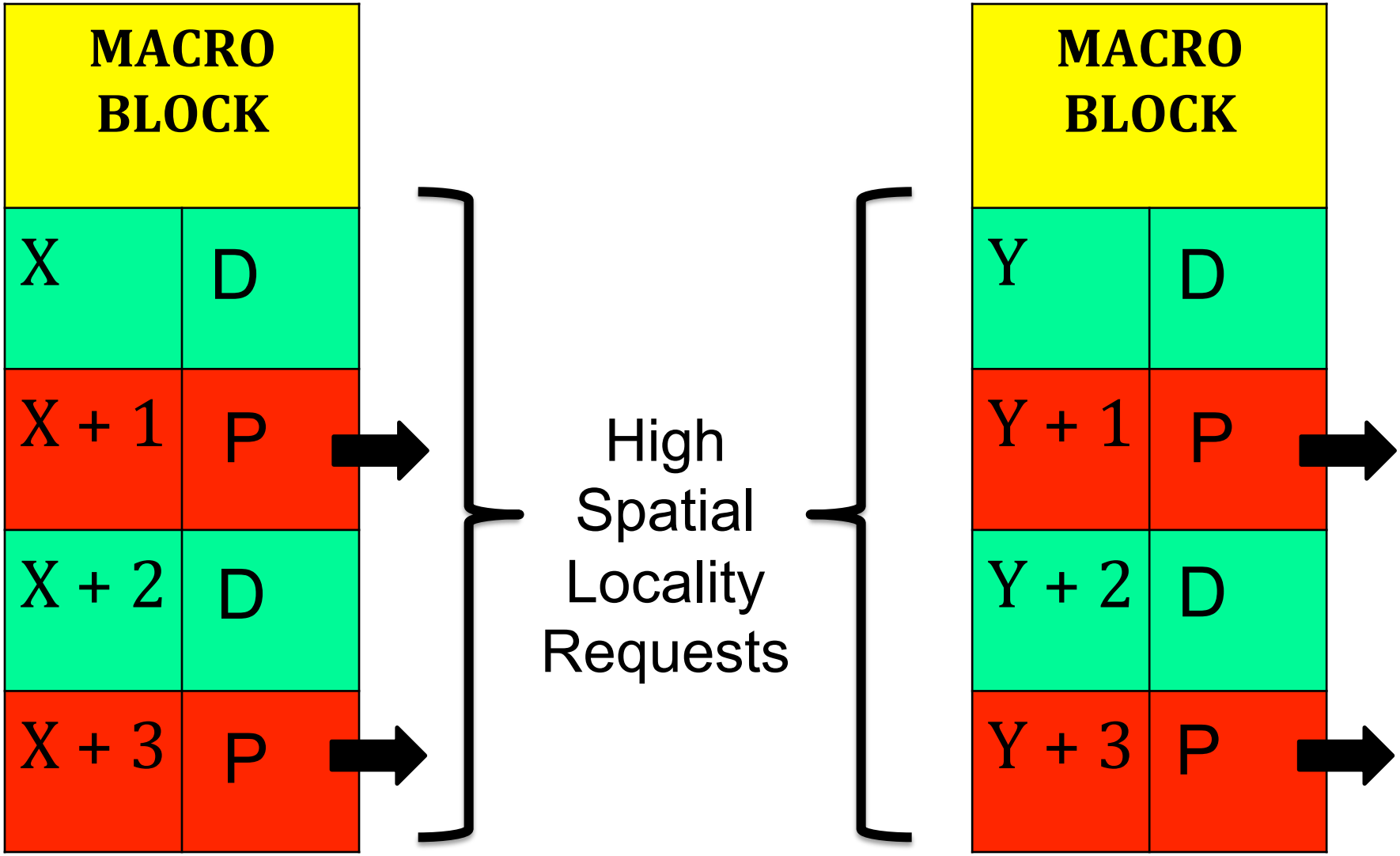
■ 1 miss ■ 2 misses ■ 3-4 misses to a macro-block



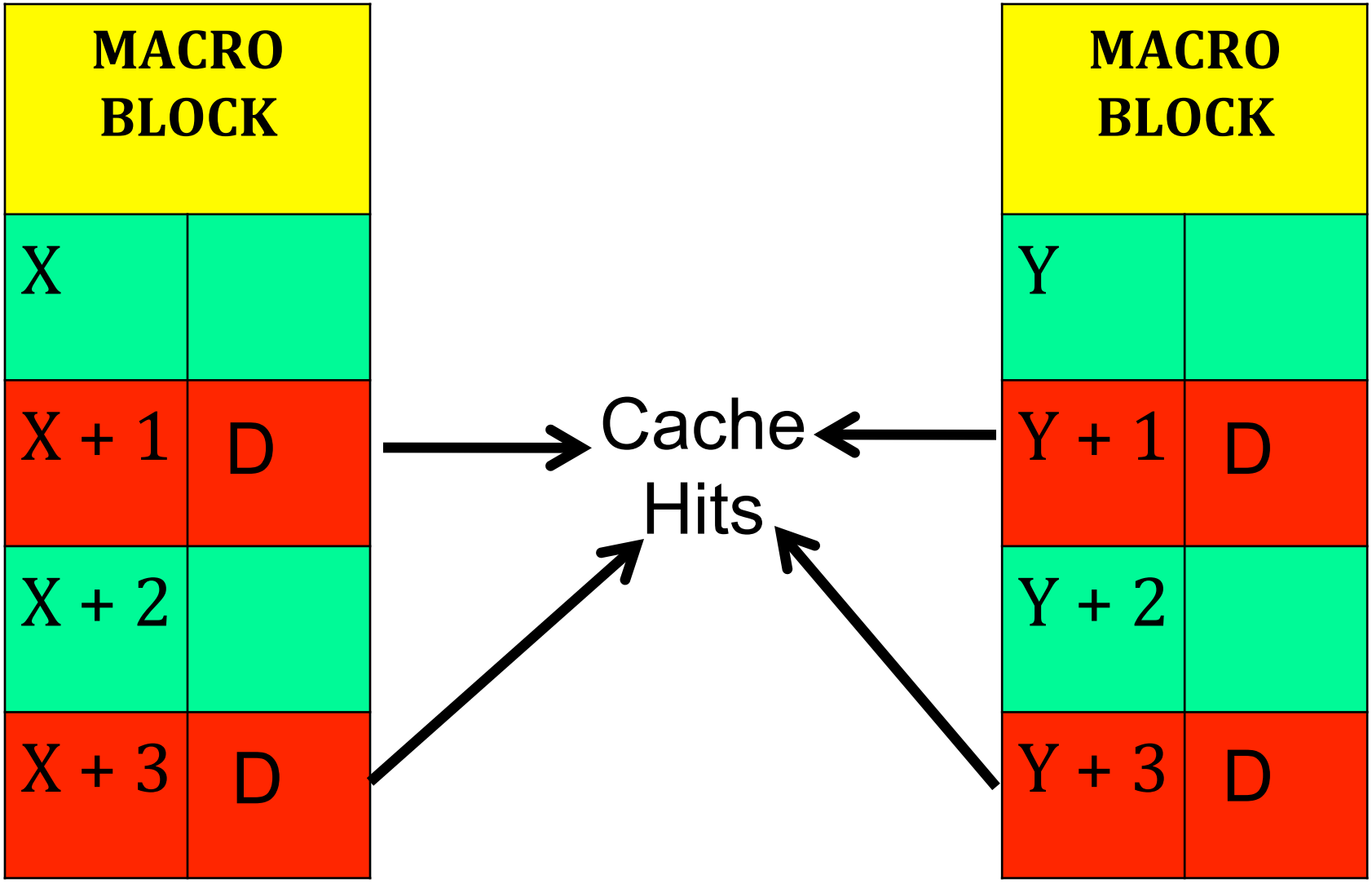
Working (With Two-Level Scheduling)



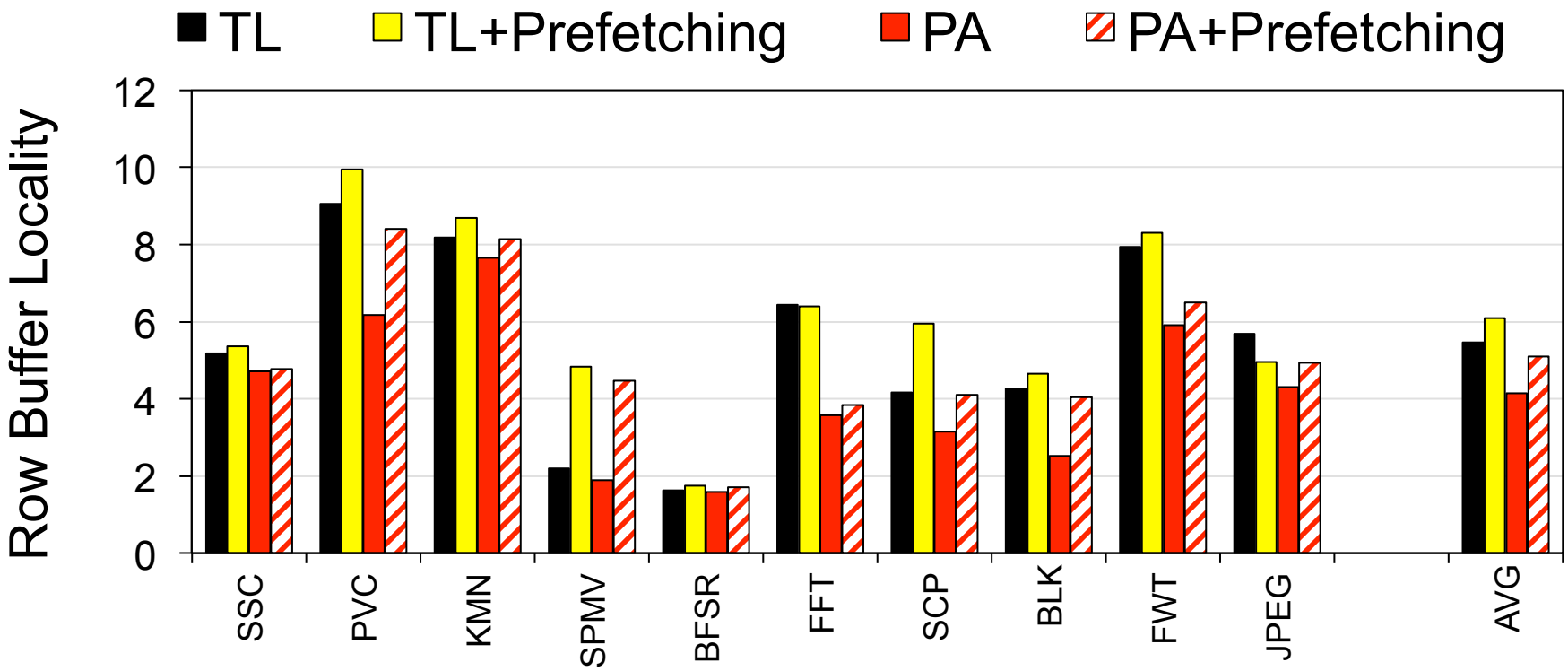
Working (With Prefetch-Aware Scheduling)



Working (With Prefetch-Aware Scheduling)

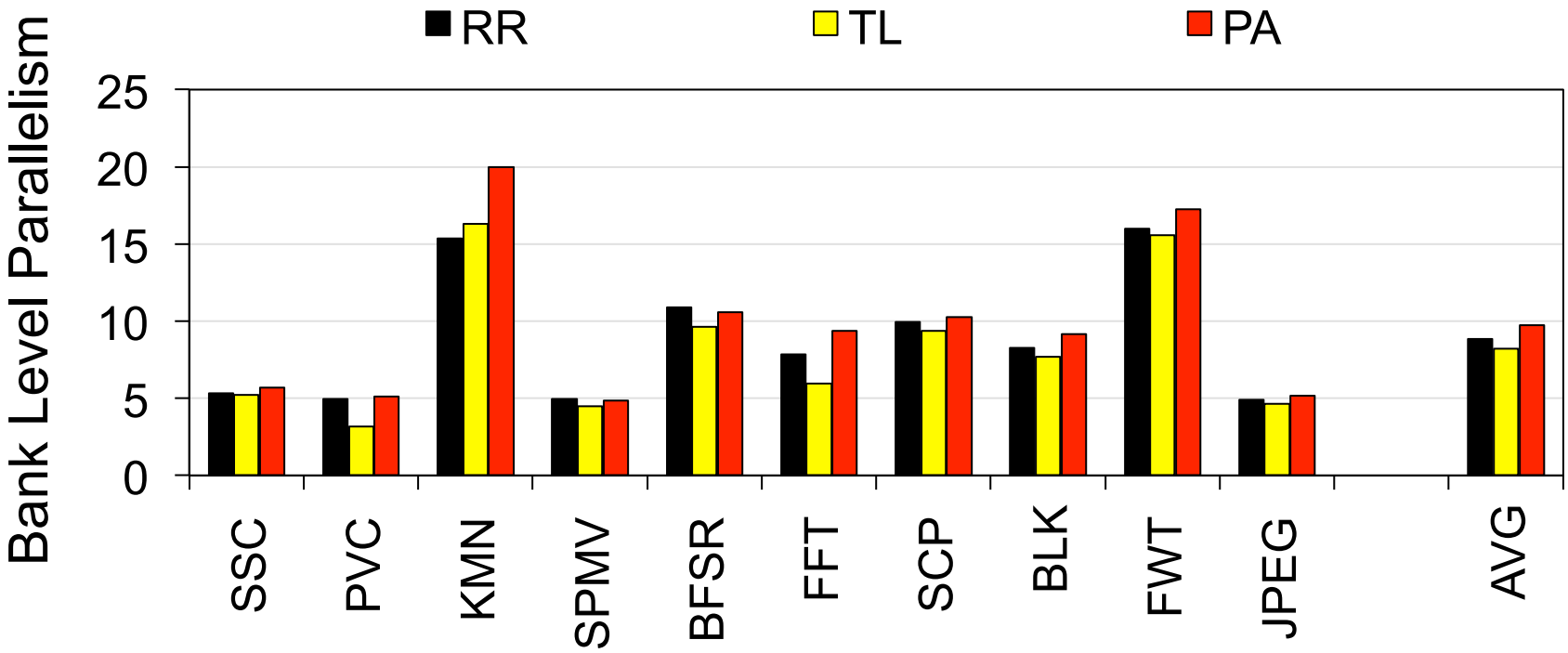


Effect on Row Buffer locality



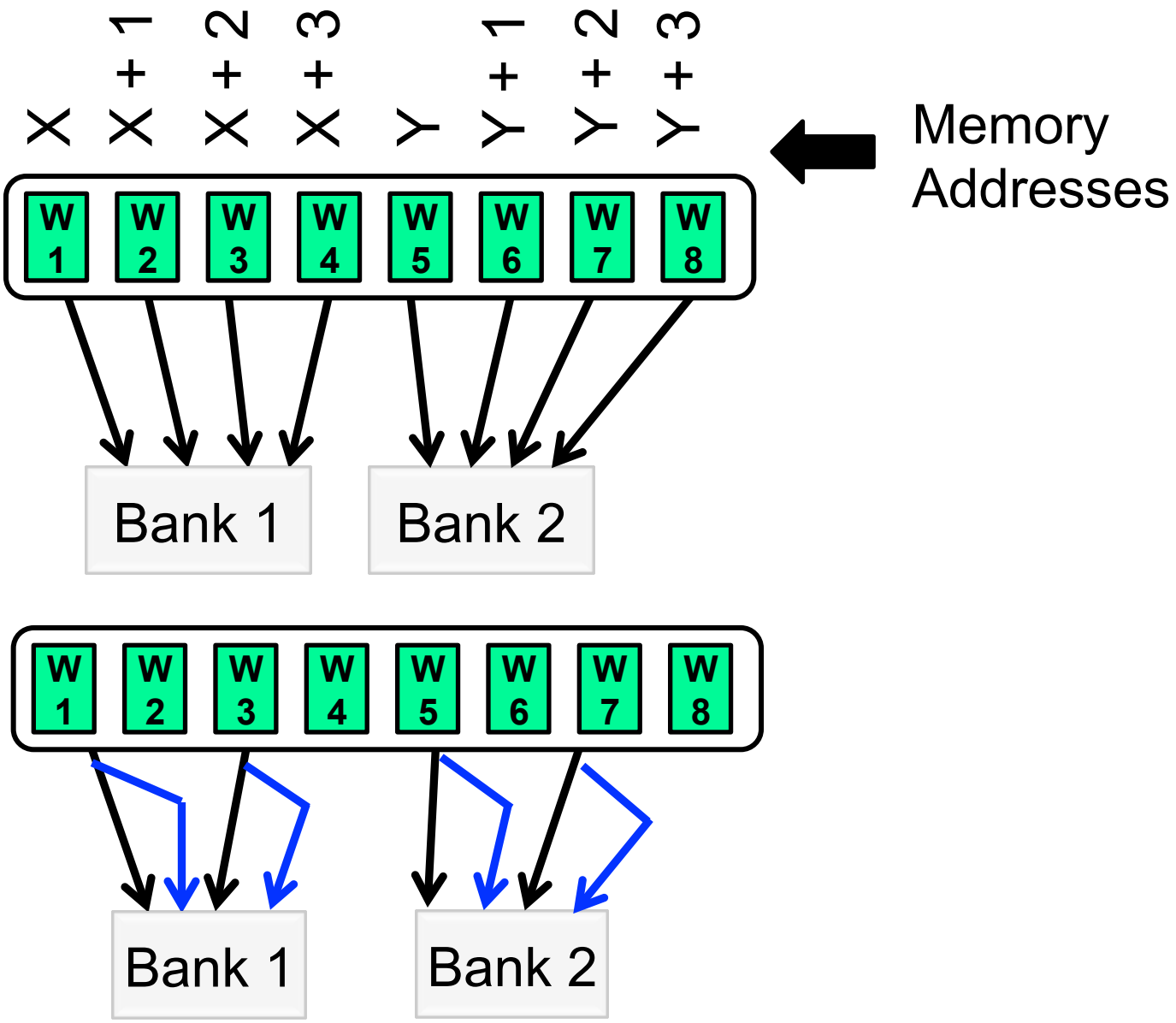
24% decrease in row buffer locality over TL

Effect on Bank-Level Parallelism

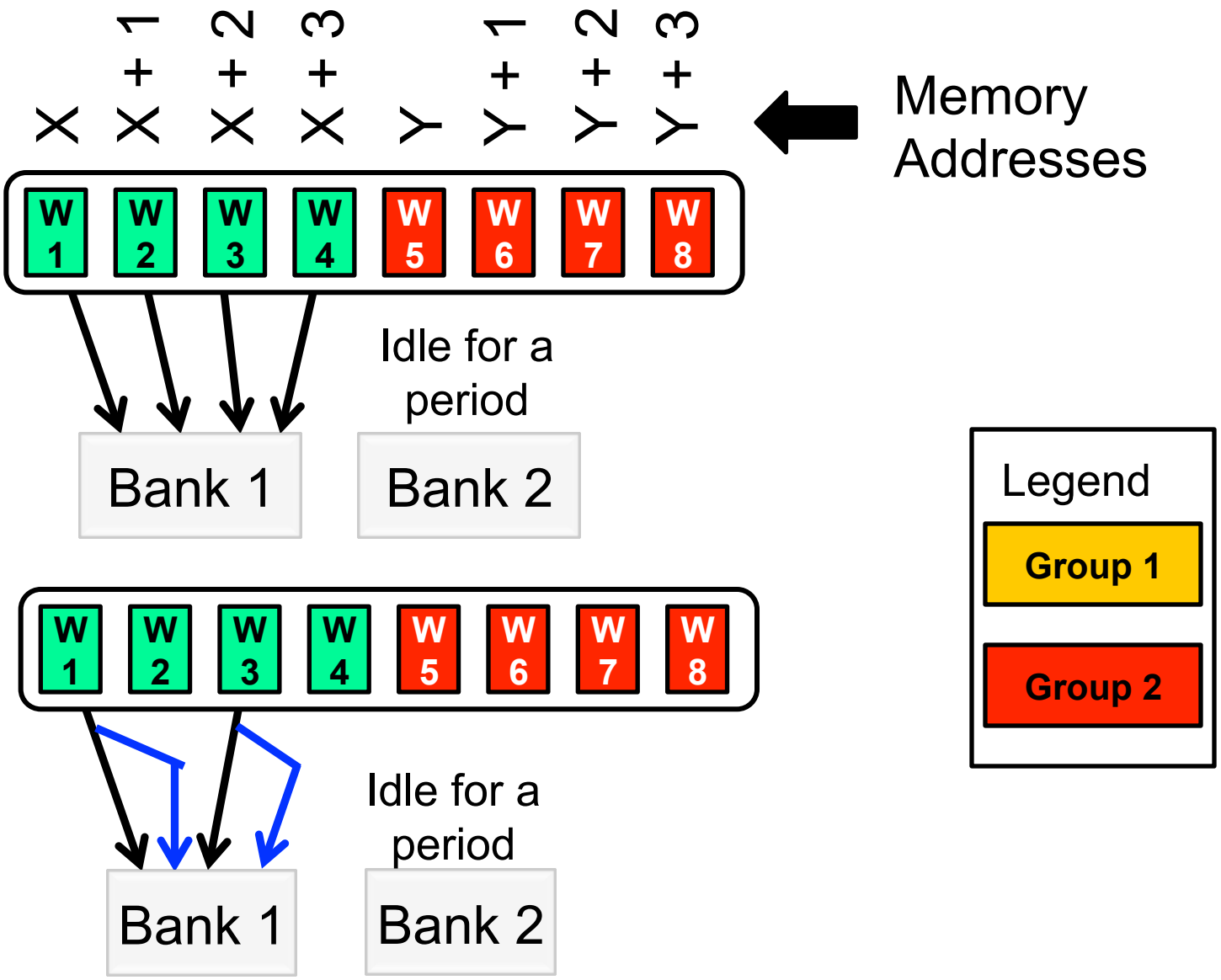


18% increase in bank-level parallelism over TL

Simple Prefetching + *RR* scheduling



Simple Prefetching with *TL* scheduling



CTA-Assignment Policy (Example)