Balancing DRAM Locality and Parallelism in Shared Memory CMP Systems

Min Kyu Jeong, Doe Hyun Yoon^, Dam Sunwoo*,
Michael Sullivan, Ikhwan Lee, and Mattan Erez

The University of Texas at Austin
Hewlett-Packard Labs^
ARM Inc.*

Executive Summary

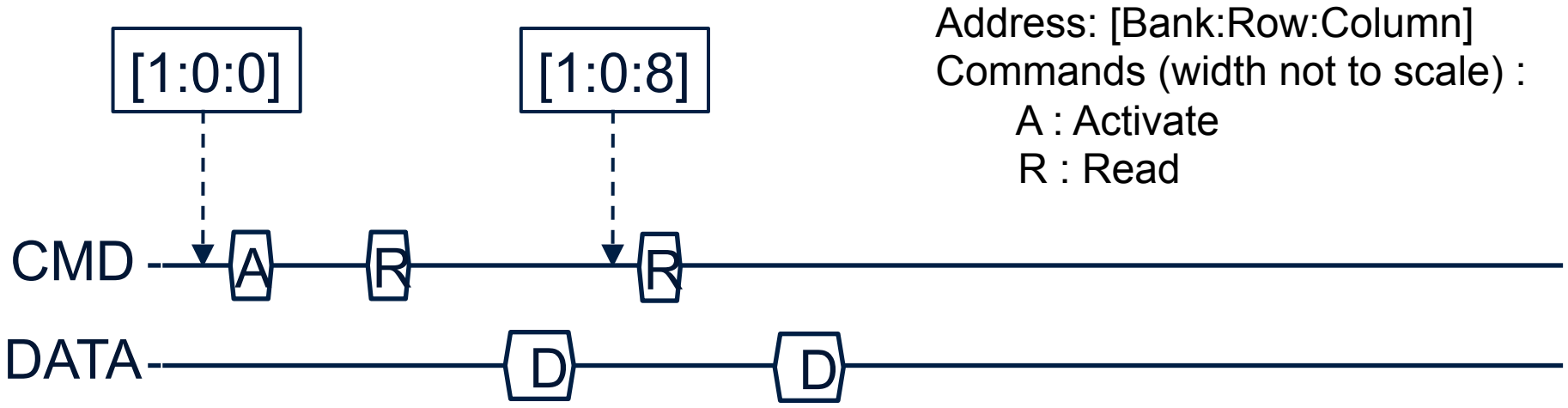
- Spatial locality is lost when independent access streams from many cores are interleaved
- To preserve the locality, we propose to isolate streams to exclusive set of DRAM banks
- Partitioning banks reduces bank-level parallelism available to each thread
- To compensate for lost BLP, we increase effective bank count with sub-ranking
- Our combined approach simultaneously improves performance and efficiency, while maintaining fairness

Outline

1. Motivation - Locality Interference
2. Locality - Bank-partitioning
3. Parallelism - Sub-ranking
4. Experimental Results
5. Conclusion

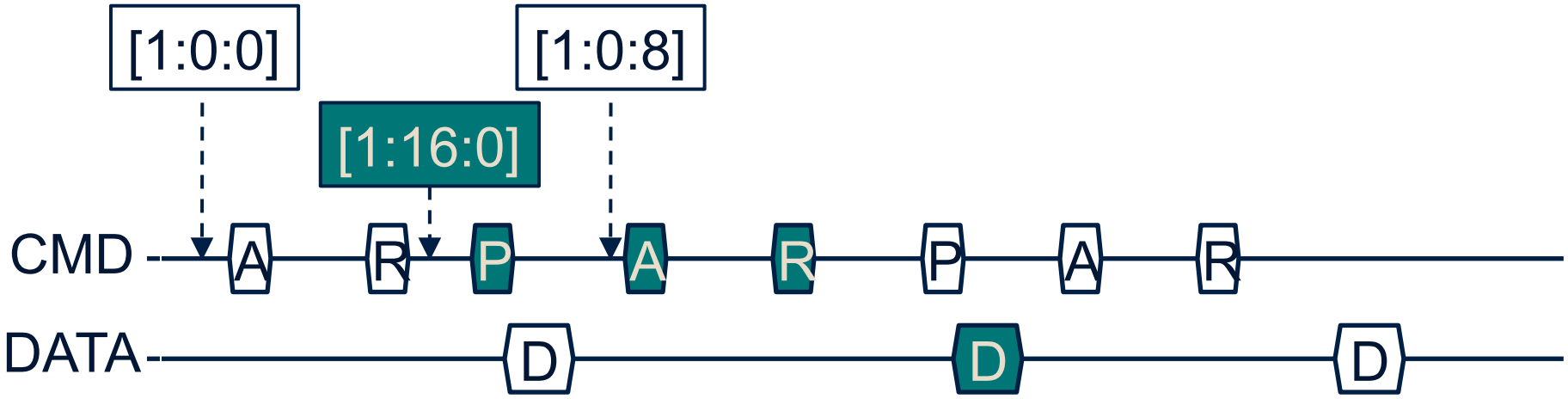
Spatial Locality in DRAM

- Many applications exhibit spatial locality
- Modern memory systems are designed to exploit spatial locality to deliver performance cost effectively (e.g. Row-Buffer Hits)



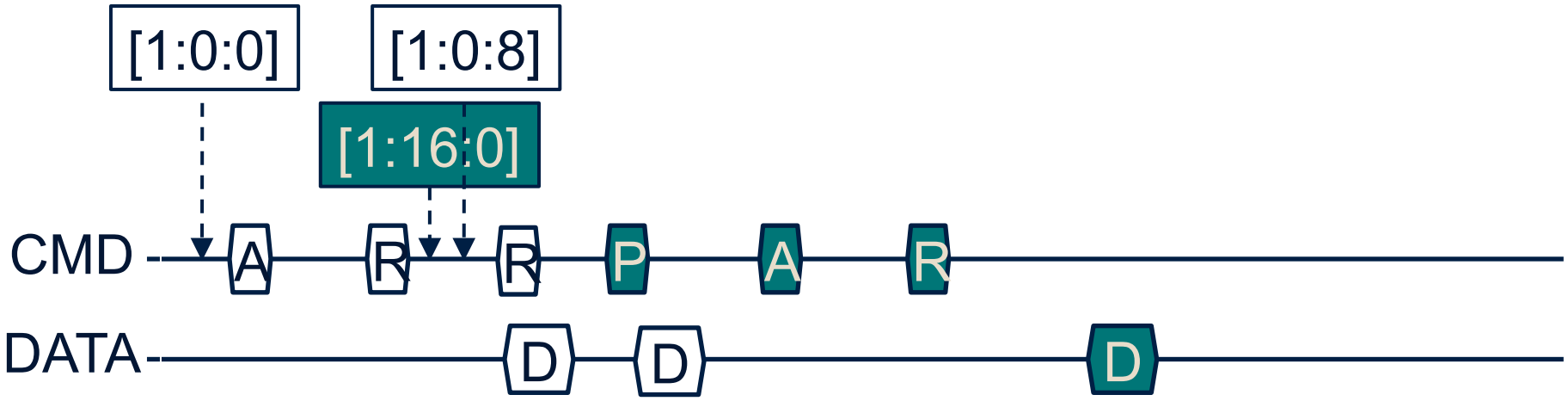
Loss of Opportunity

- However, in chip-multiprocessor systems, spatial locality is lost as independent access streams from multiple cores are interleaved
- Result: Lower performance and energy efficiency



Prior Work

- Out-of-order scheduling
  - Reduces the number of back-and-forth row swapping
  - Arrival interval should be short enough
  - Limited by the scheduling queue size
  - Delaying certain streams hurts performance and fairness



Prior Work

- MP fairness-aware scheduling
  - Maximizing bandwidth != system performance
  - Optimize for system fairness and performance

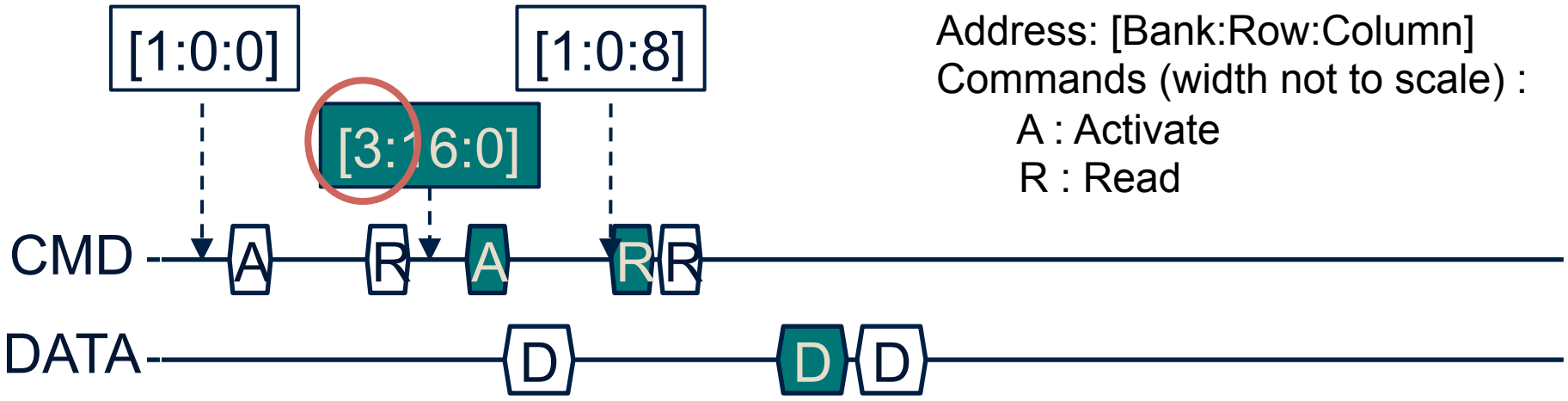
- **All still pay the cost of bank conflicts**

Outline

1. Motivation - Locality Interference
2. **Locality - Bank-partitioning**
3. Parallelism - Sub-ranking
4. Experimental Results
5. Conclusion

Eliminate Inter-Process Bank Conflicts

- Make different cores to use different DRAM banks



- Modify the physical frame allocation algorithm of an OS

Virtual to Physical to DRAM Address

Bit index	…	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Virtual Address	Virtual Page Number														Page Offset												
Physical Address	Physical Frame Number														Frame Offset												
DRAM Address Bit-mask	Row													Bank	Column												

Address Translation Map

Page Table P0

Page #	Frame #
0	x00
1	x01
2	x02
3	x03

Page Table P1

Page #	Frame #
0	x04
1	x05
2	x42
3	x43

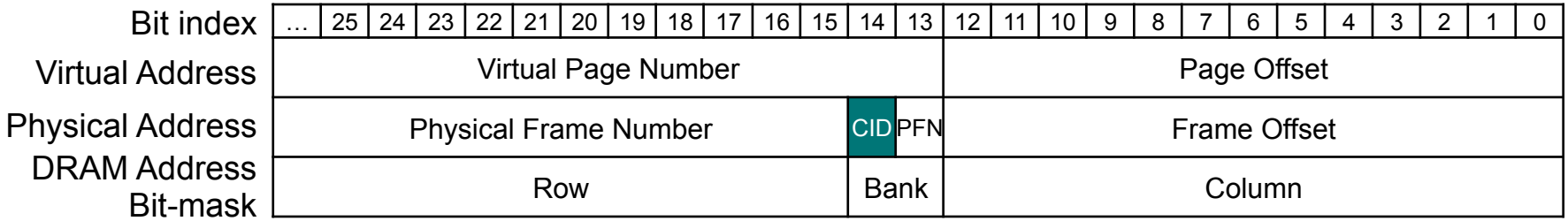
Frame Table

Frame #	DRAM Addr
x00	Bank 0, Row 0
x01	Bank 1, Row 0
x02	Bank 2, Row 0
x03	Bank 3, Row 0
x04	Bank 0, Row 1
x05	Bank 1, Row 2
…	….
x40	Bank 0, Row 16
x41	Bank 1, Row 16
x42	Bank 2, Row 16
x43	Bank 3, Row 16

Physical Frame Layout in DRAM

Bank 0	Bank 1	Bank 2	Bank 3	
x00	x01	x02	x03	Row 0
x04	x05	x06	x07	Row 1
				…
x40	x41	x42	x43	Row 16

Bank-partitioning Frame Allocation



Address Translation Map

Page Table P0

Page #	Frame #
0	x00
1	x01
2	x04
3	x05

Page Table P1

Page #	Frame #
0	x02
1	x03
2	x42
3	x43

Frame Table

Frame #	DRAM Addr
x00	Bank 0, Row 0
x01	Bank 1, Row 0
x02	Bank 2, Row 0
x03	Bank 3, Row 0
x04	Bank 0, Row 1
x05	Bank 1, Row 2
…	….
x40	Bank 0, Row 16
x41	Bank 1, Row 16
x42	Bank 2, Row 16
x43	Bank 3, Row 16

Physical Frame Layout in DRAM

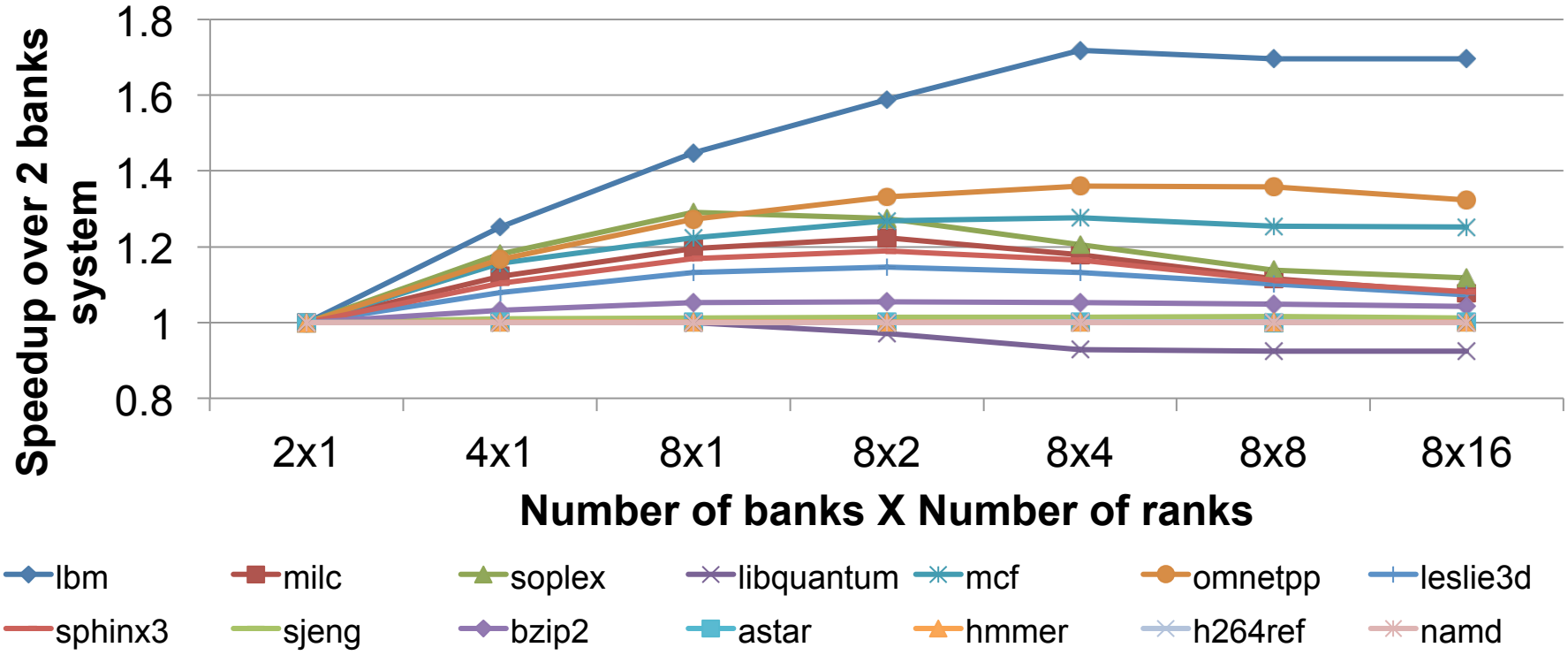
Bank 0	Bank 1	Bank 2	Bank 3	
x00	x01	x02	x03	Row 0
x04	x05	x06	x07	Row 1
				…
x40	x41	x42	x43	Row 16

Outline

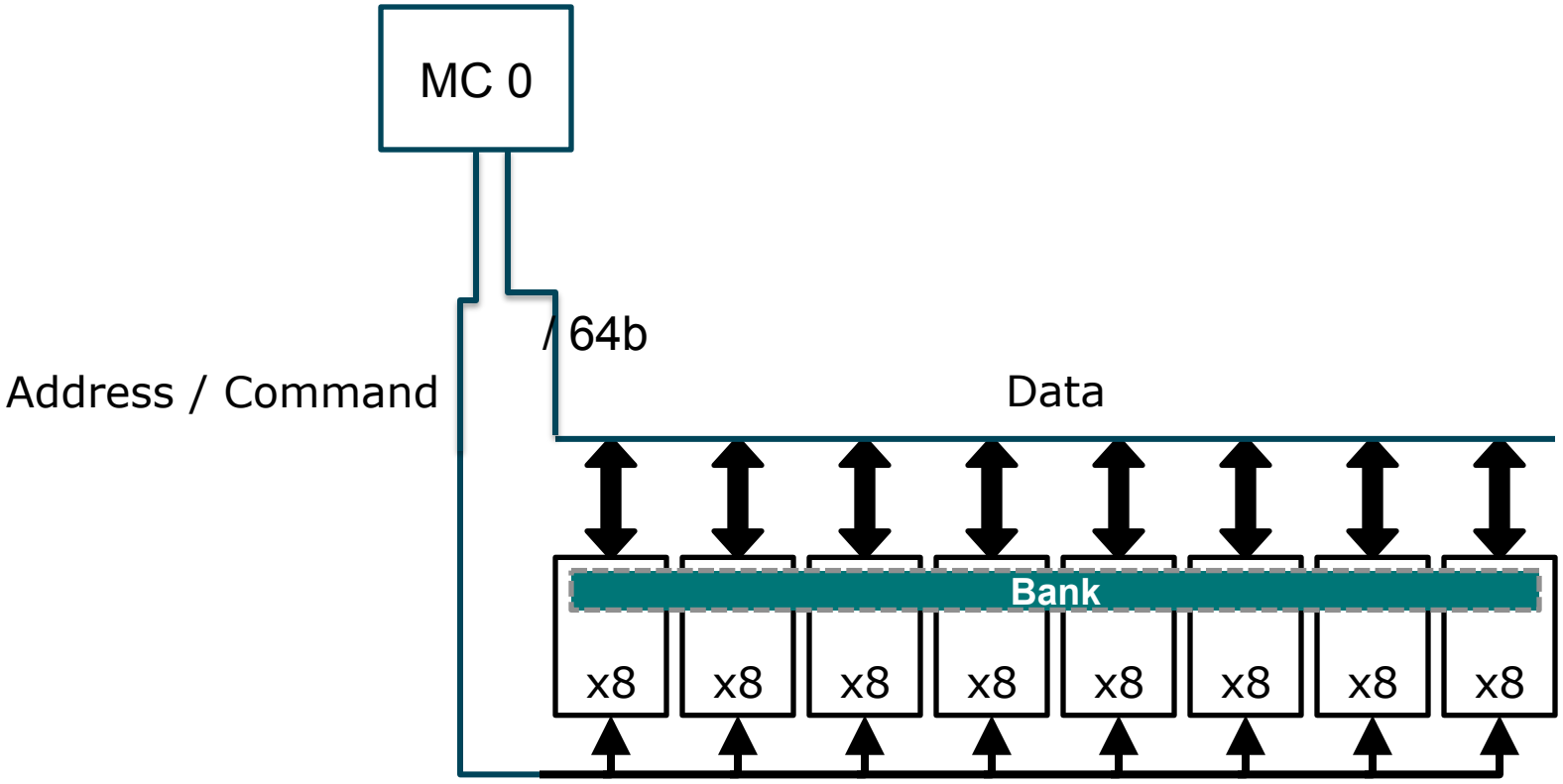
1. Motivation - Locality Interference
2. Locality - Bank-partitioning
3. **Parallelism - Sub-ranking**
4. Experimental Results
5. Conclusion

Bank-Level Parallelism

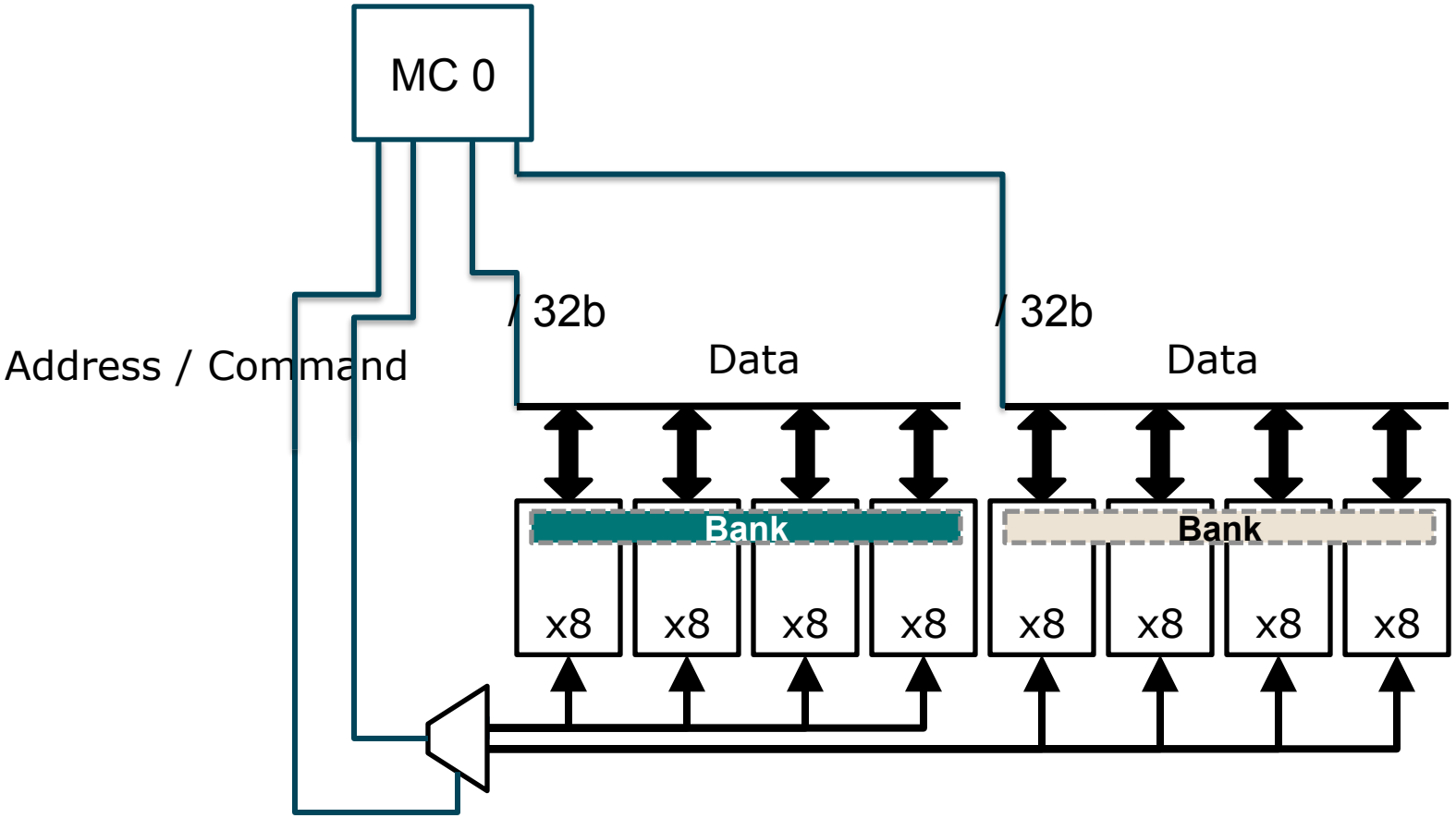
- Bank-partitioning reduces the number of banks per thread
- Applications with low spatial locality needs many banks to overlap long latency accesses
- How many do we need?



Conventional Rank Structure



Sub-ranking


MC 0
Address / Command
/ 32b
Data
/ 32b
Data
Bank
Bank
x8
x8
x8
x8
x8
x8
x8
x8

Trading off Parallelism and Locality

- Bank Partitioning
  - Isolate streams to preserve *locality*
  - Good for applications with high spatial locality
- Sub-ranking
  - Controls subsets of rank independently, increases *BLP*
  - Good for applications with low spatial locality

- The two techniques complement each other and improve synergistically

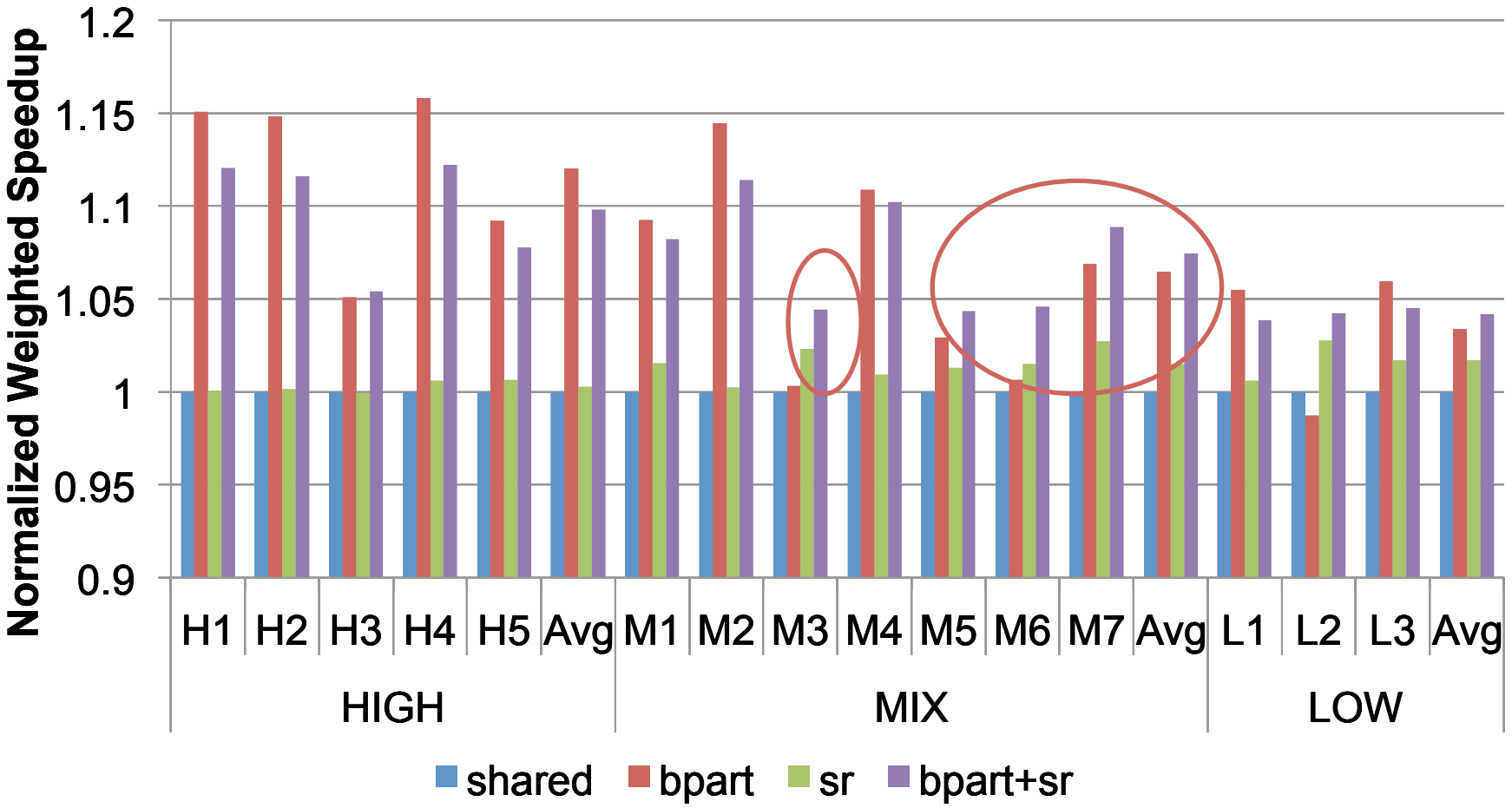
Outline

1. Motivation - Locality Interference
2. Locality - Bank-partitioning
3. Parallelism - Sub-ranking
4. **Experimental Results**
5. Conclusion

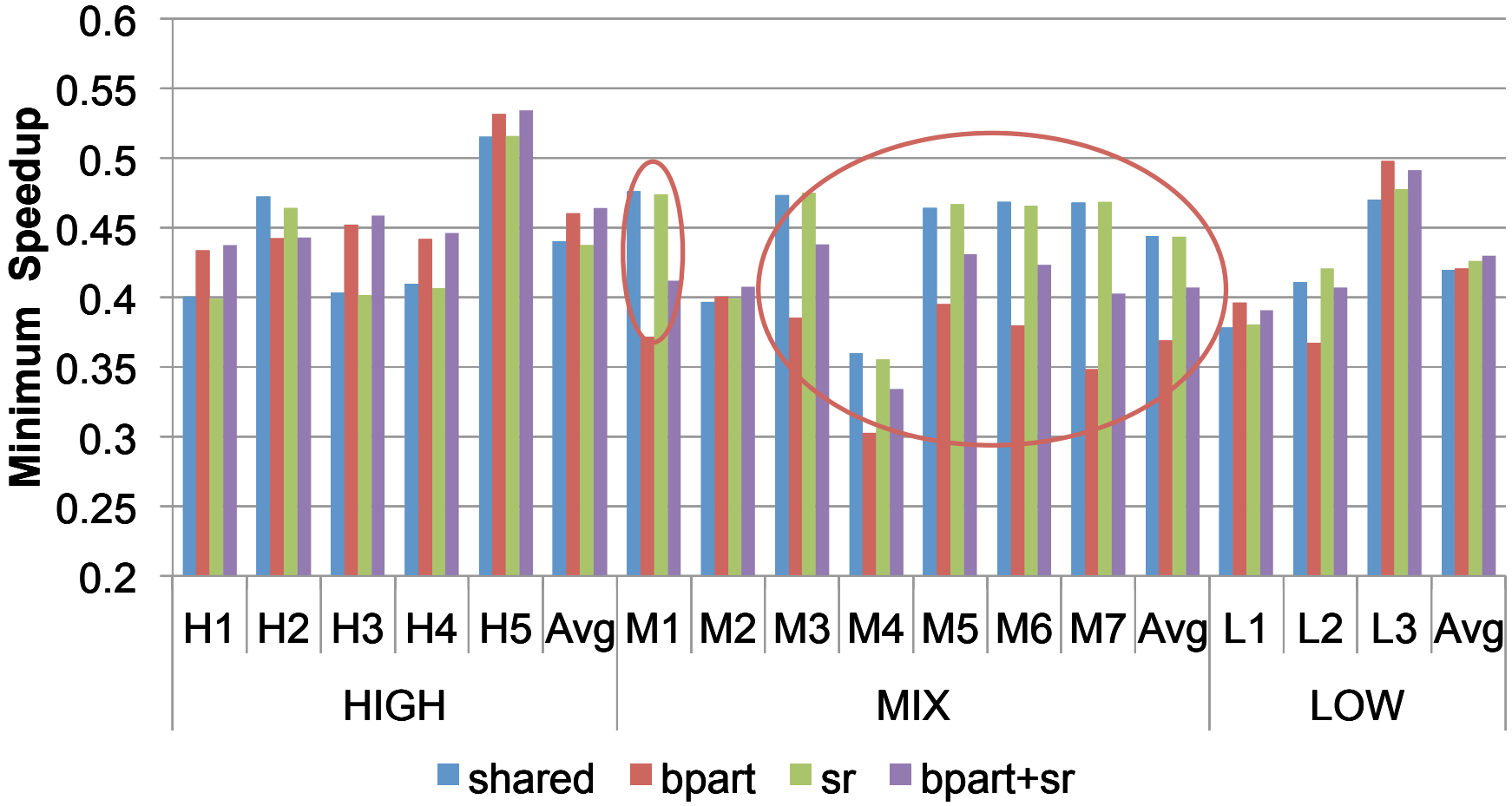
Evaluation Setup

- Simulator configuration (Zesto)
  - 4GHz x86 out-of-order 8-core processor
  - Private 32KB I/D L1, 256KB L2, next-line prefetcher
  - Shared 8MB L3, stream prefetcher
  - Syscall-emulated. Frame-allocation code modified
  - 2 channels, 2 ranks/channel, 8 banks/rank
    DDR-1600. FR-FCFS
- Workloads
  - Multi-programmed workloads consisting of
    memory intensive benchmarks from SPEC CPU2006
  - 4 workload groups:
    HIGH, MIX, LOW (Spatial Locality), and LOW_BW
  - 200 million instructions SimPoint

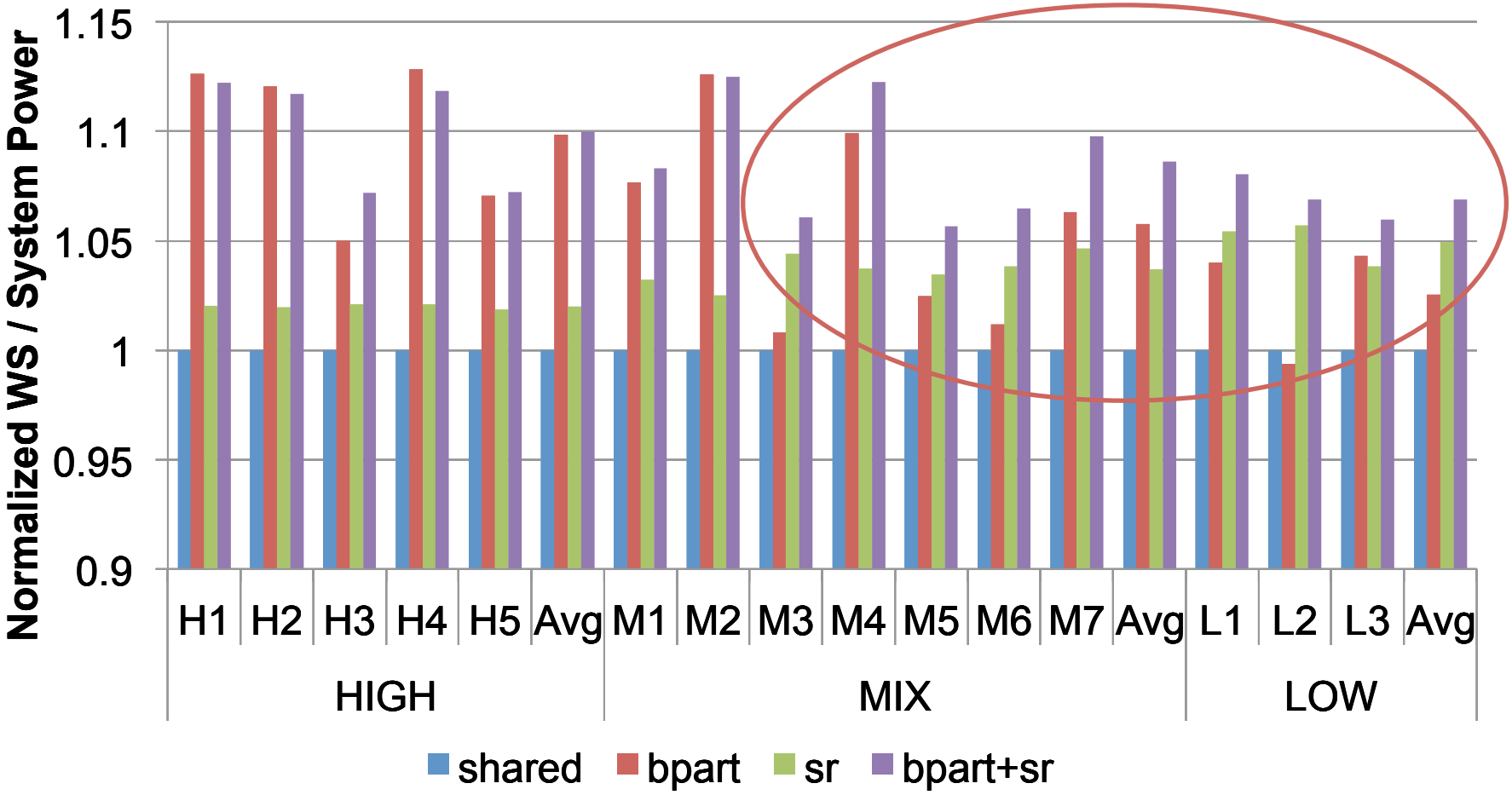
System Throughput



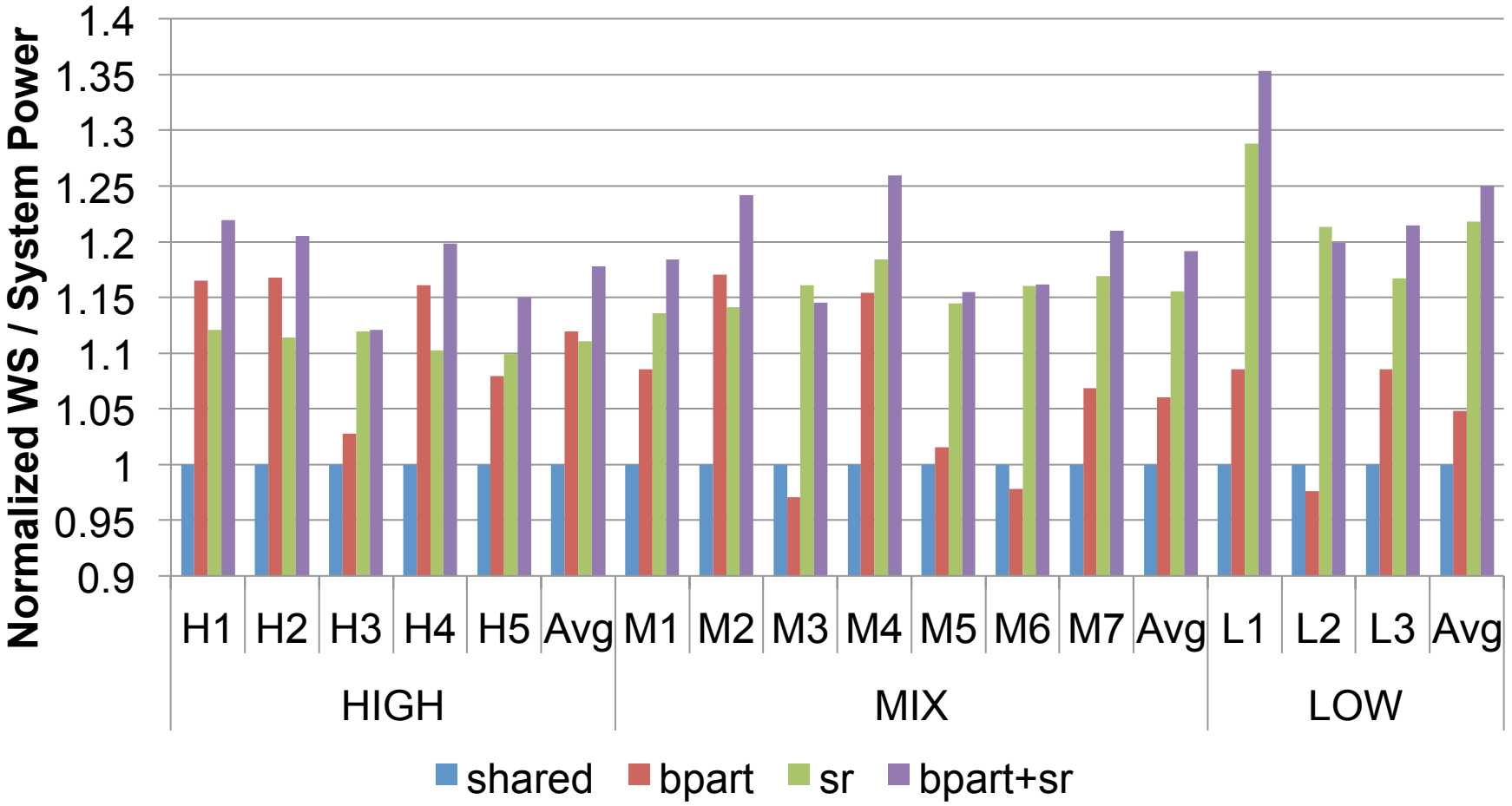
Fairness


Minimum Speedup
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
H1
H2
H3
H4
H5
Avg
M1
M2
M3
M4
M5
M6
M7
Avg
L1
L2
L3
Avg
HIGH
MIX
LOW
shared
bpart
sr
bpart+sr

System Efficiency



System Eff. Of Bank-Limited System



Conclusion

- Combination of bank partitioning and sub-ranking balances locality and parallelism
- It boosts performance and efficiency of the system simultaneously while maintaining fairness
  - 10%, 7%, and 5% throughput gain for HIGH, MIX, and LOW
  - 10%, 9%, and 6% efficiency gain
  - 21.4% DRAM Power reduction on average
  - 15% fairness gain over bank-partitioning only in MIX
- Larger improvements for systems with higher core/ bank ratio