High-Throughput Coherence Controllers

Anthony-Trung Nguyen
Microprocessor Research Labs
Intel Corporation
9/30/03

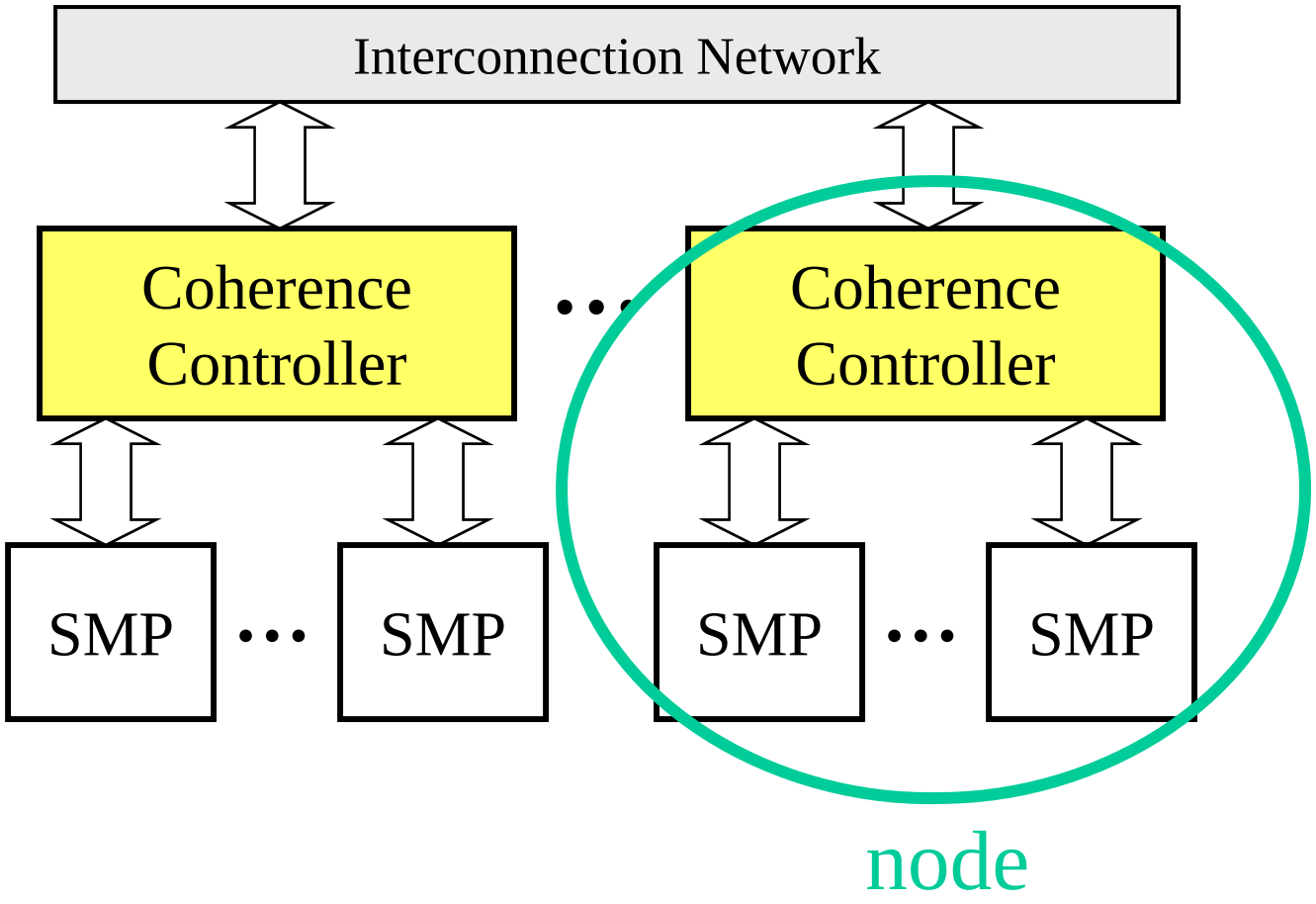
Motivations

- Coherence Controller (CC) throughput is bottleneck of scalable systems.
- CCs need to meet throughput demand to maintain good performance.
- High-throughput CCs allow construction of wide nodes with good performance.
- Microprocessor techniques can be adapted to enhance CC throughput.

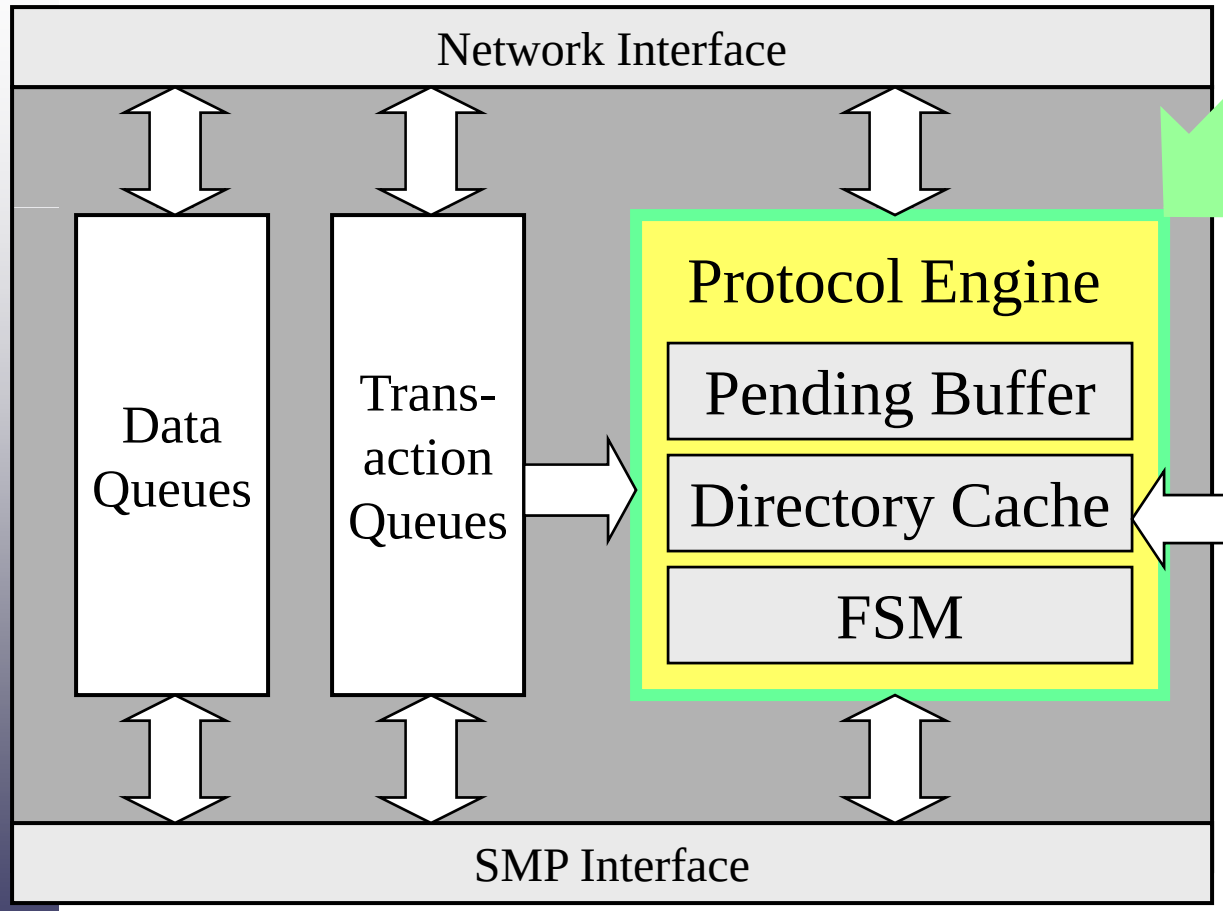
Goals

- Use known and innovative techniques to enhance throughput of hardwired CCs.
- **Six techniques**:
  1. Multiple protocol engines (PEs)
  2. Pipelined PEs
  3. Split request-response streams
  4. Nonblocking PEs
  5. Directory/L3 early fetching
  6. Superpipelined PEs
- Evaluate effectiveness of individual techniques as well as combinations.

Shared-Memory Architecture



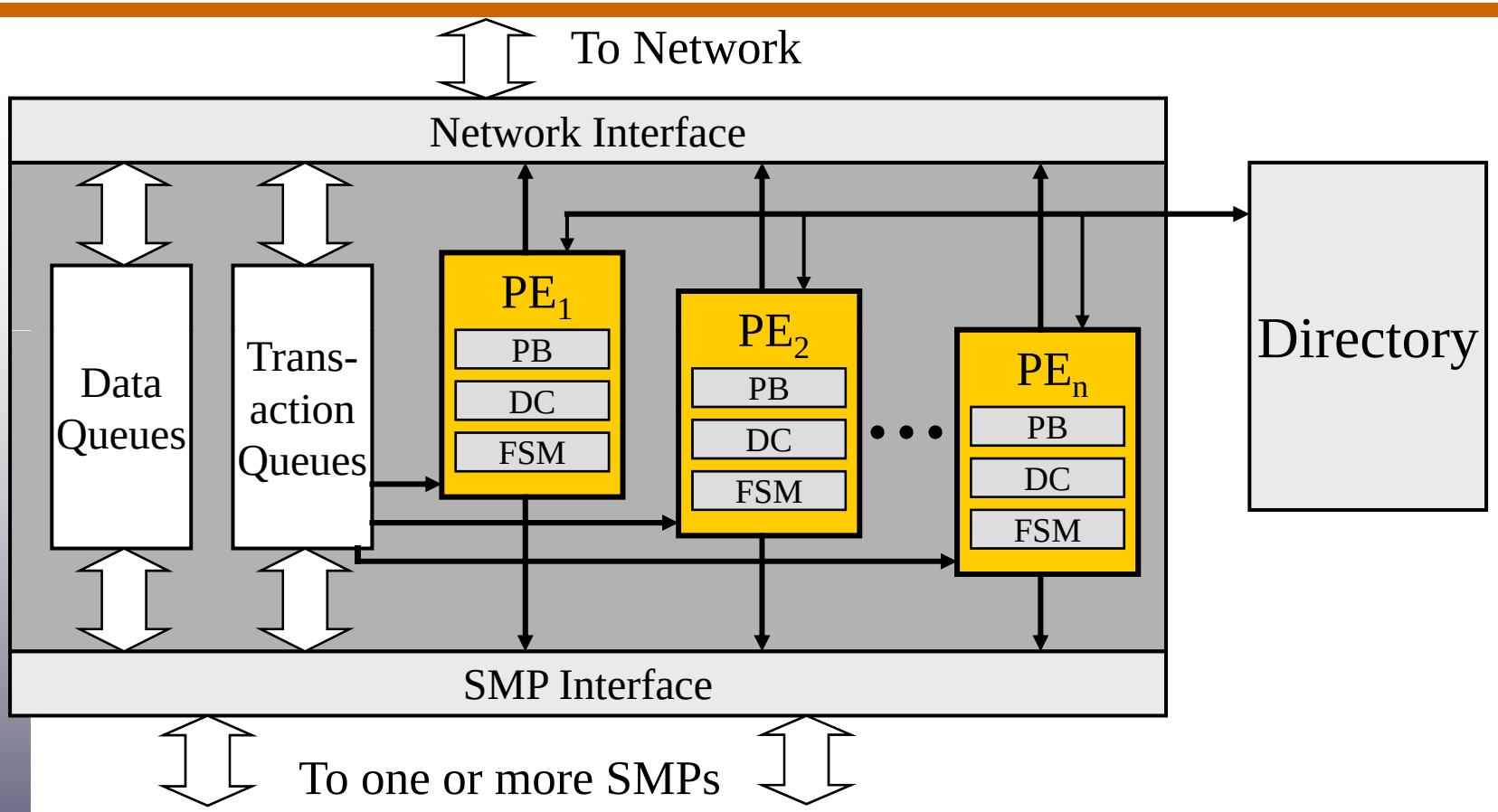
Base CC Architecture



Optimizations

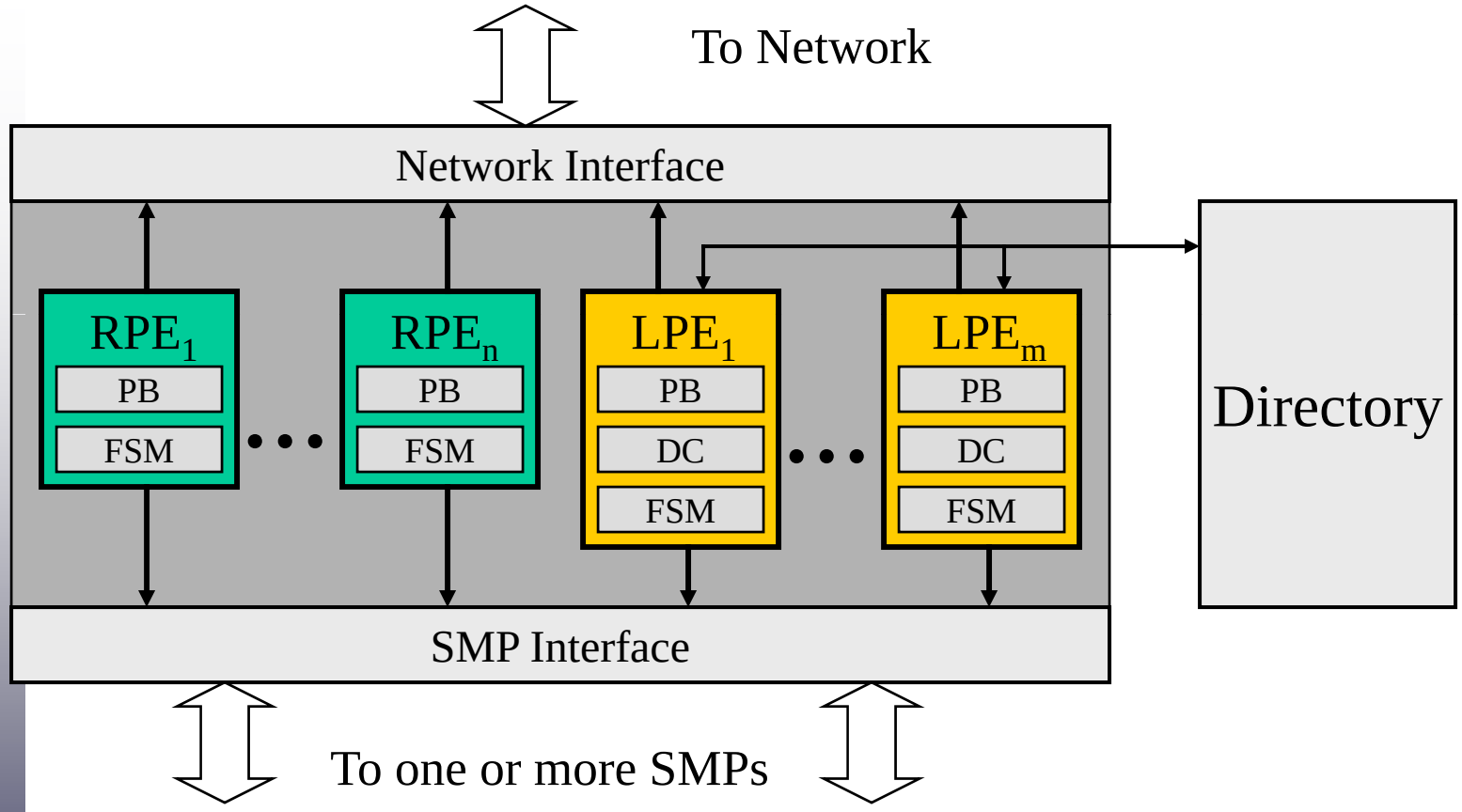
1. Multiple PEs
2. Pipelining
3. Split Streams
4. Nonblocking
5. Dir early fetch
6. Superpipeline

1. Replication of Protocol Engines



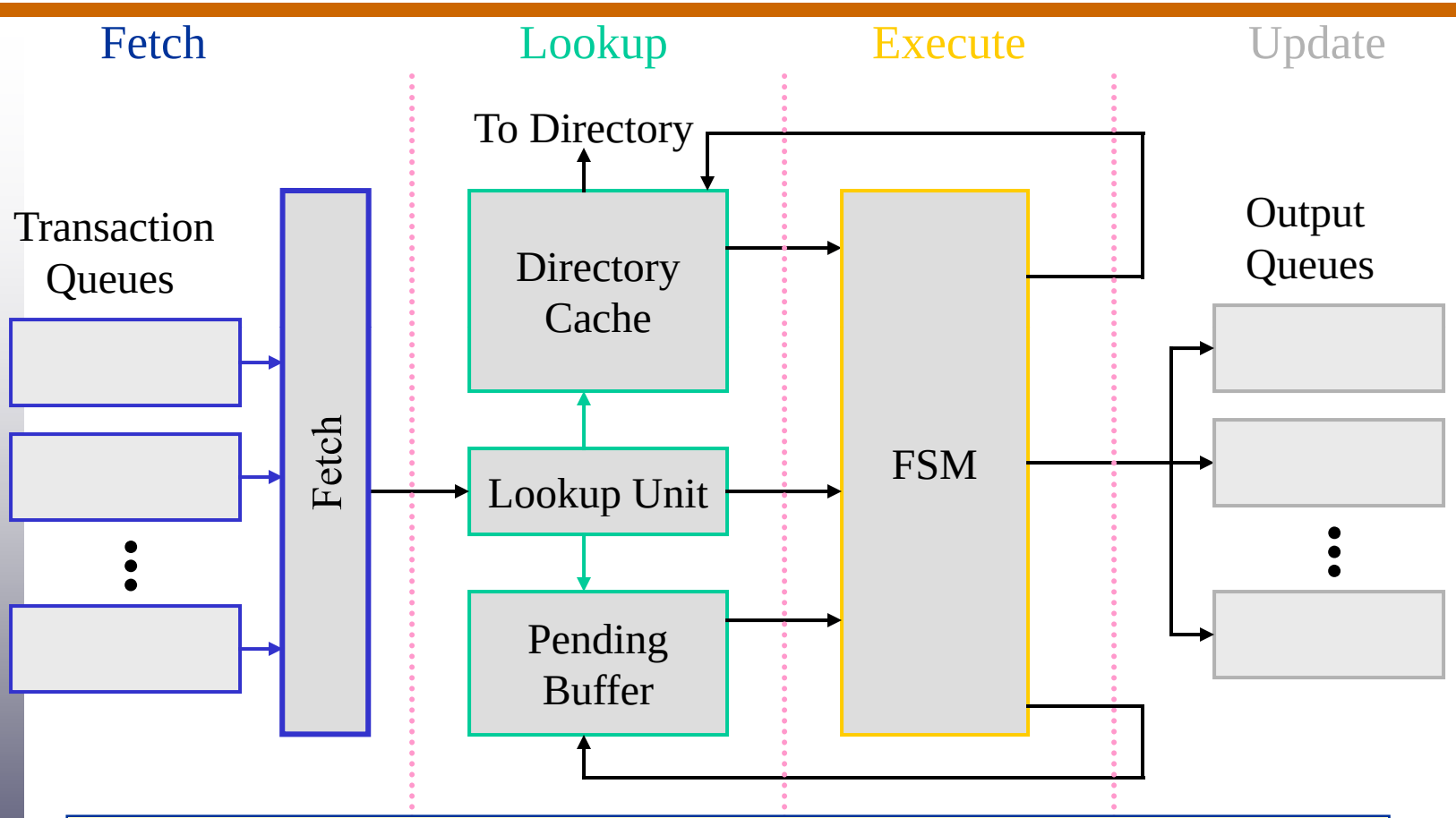
- Assign transactions using low-order bits of memory line id.
- DC misses stall transactions to remote addresses.

1. Local & Remote Protocol Engines



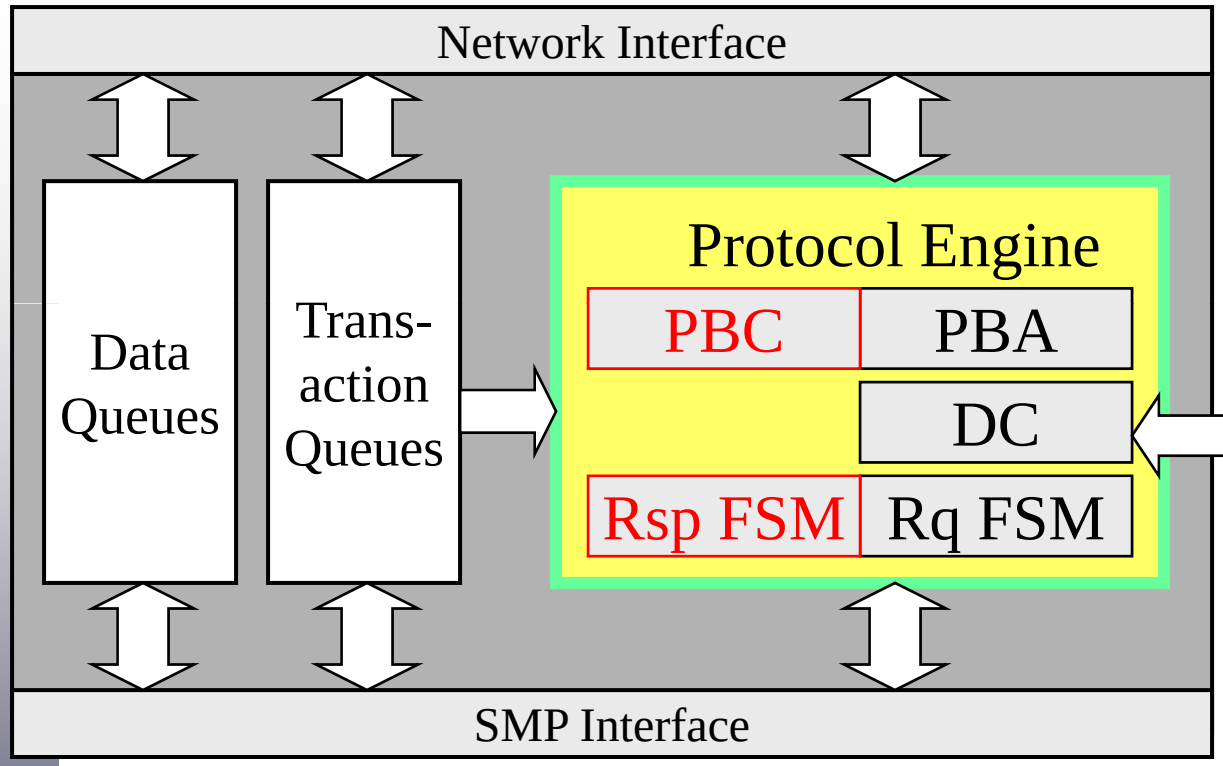
- **No DC for remote addresses.**
- **DC misses in LPEs don't stall remote addresses.**

2. Pipelined Protocol Engine



Requests allocate Pending Buffer entries & lookup DC.
Responses access existing entries – no DC lookup.

3. Split Request-Response Streams



Request

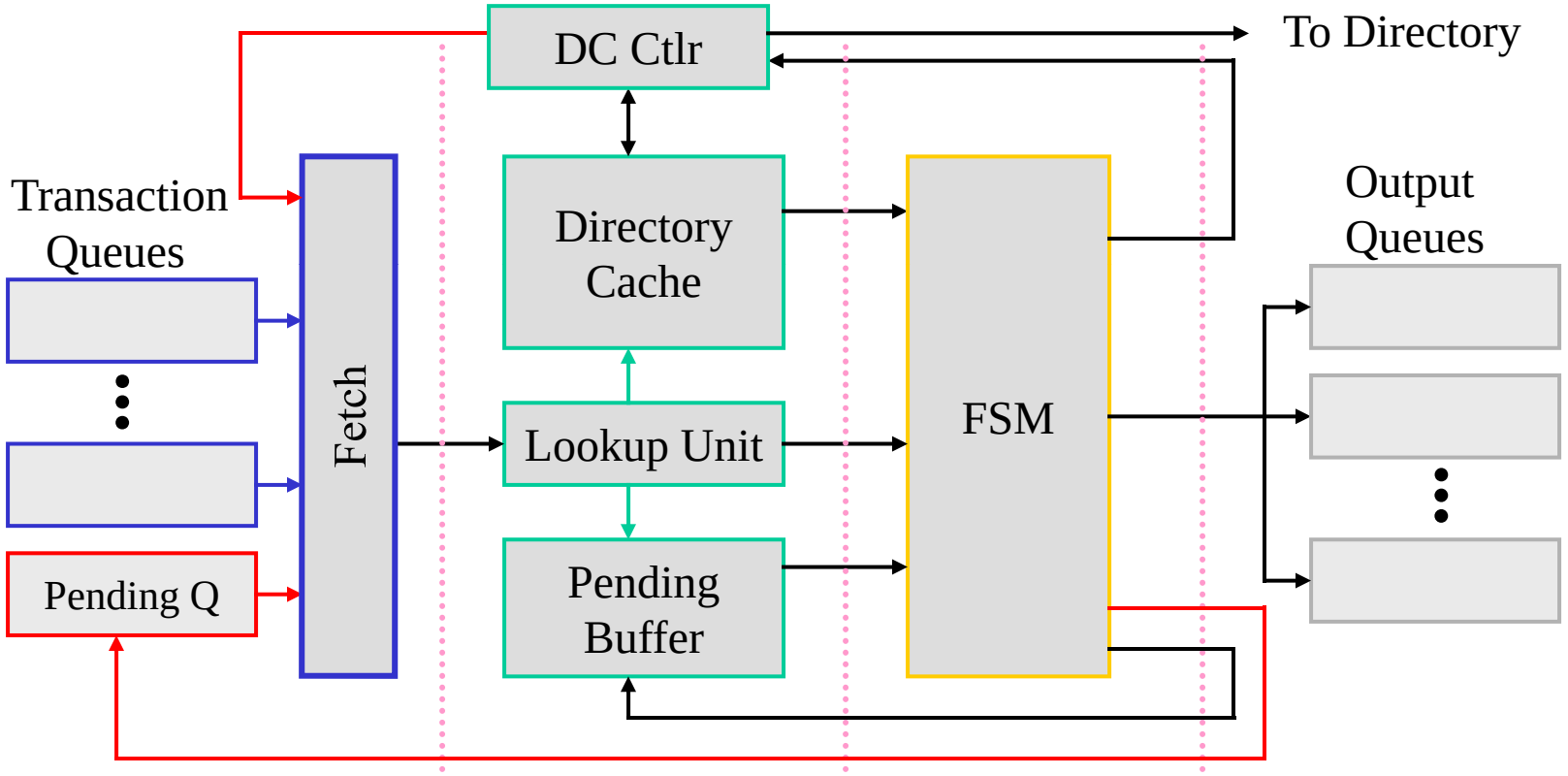
1. PB Addr
2. DC lookup
3. Req FSM

Response

1. PB Content
2. Resp FSM

- **No DC access for response transactions.**
- **Responses are not stalled by previous DC miss.**

4. Nonblocking PE



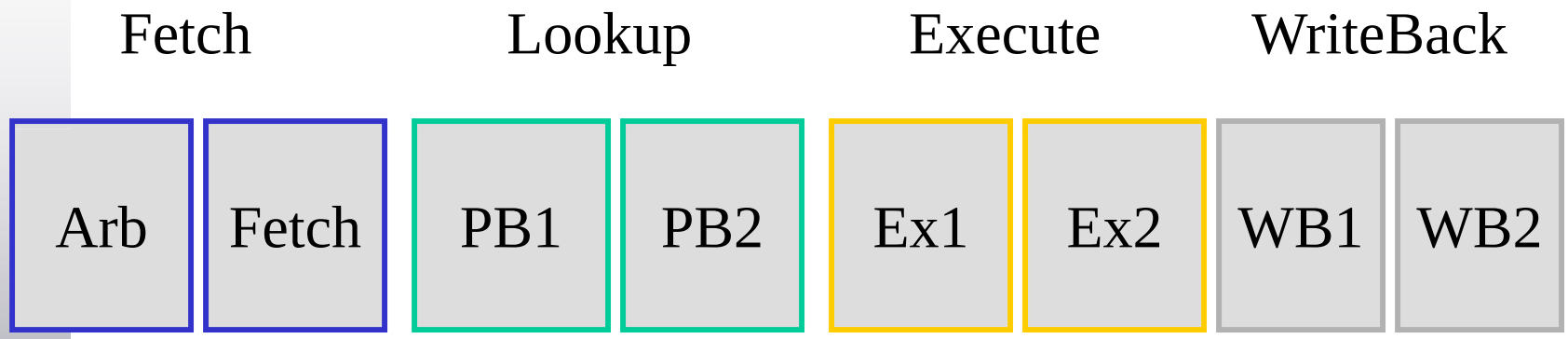
- **Put transaction with DC miss in *Pending* queue.**
- **Allow transactions to different addresses to bypass.**

5. Directory/L3 Early Fetch

When transaction enters CC queue...

- Directory Cache (DC)
  - **Issue early fetch** request to **DC** controller.
  - DC ctlr checks DC & issues early fetch to directory.
- Tag Cache (TC)
  - **Issue early fetch** request to **TC** controller.
  - TC ctlr checks TC & issue early fetch to tag array.

6. Superpipelined Protocol Engine

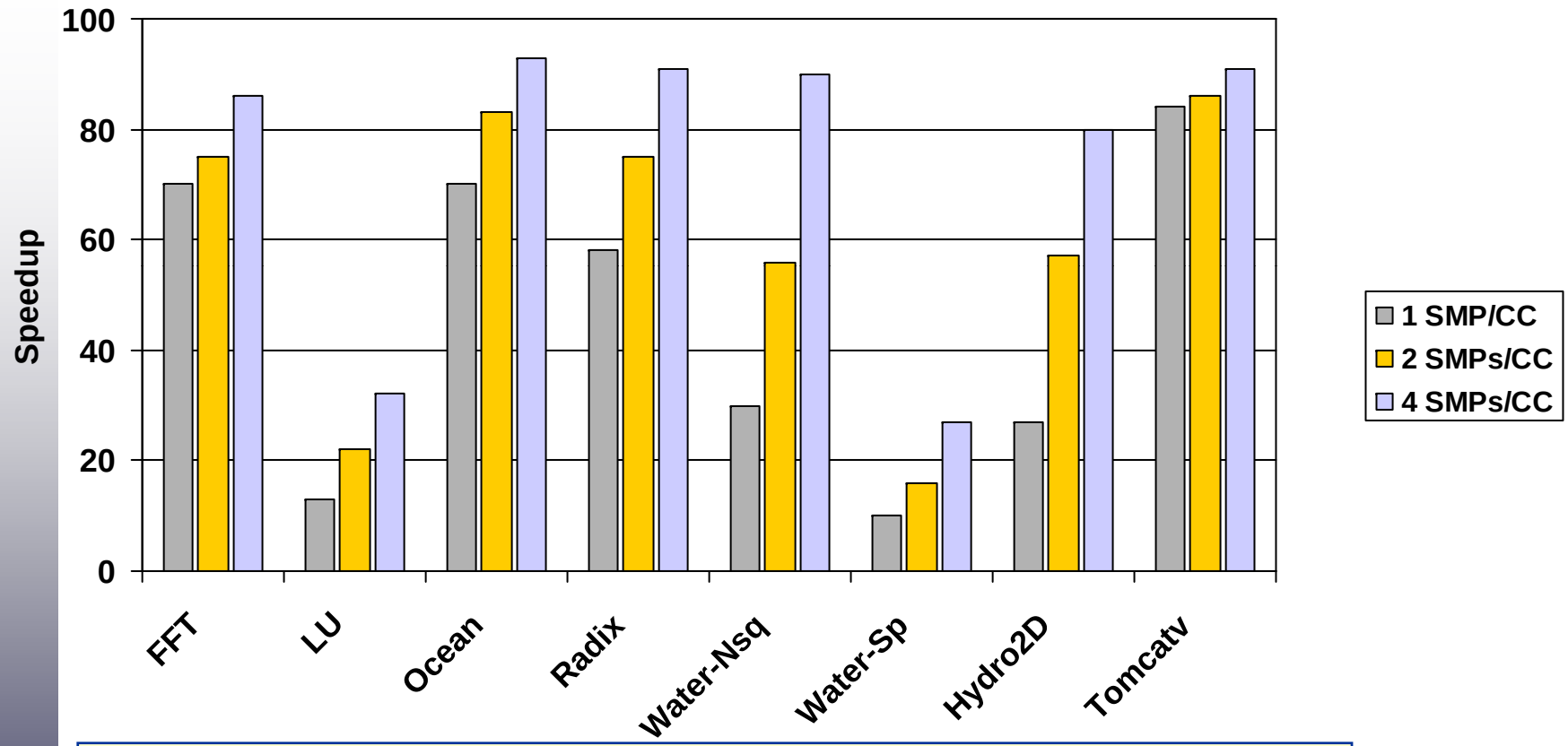


- **Decompose pipelined PE into smaller, faster stages.**
- **No branch misprediction penalty as in CPU.**

Comparisons

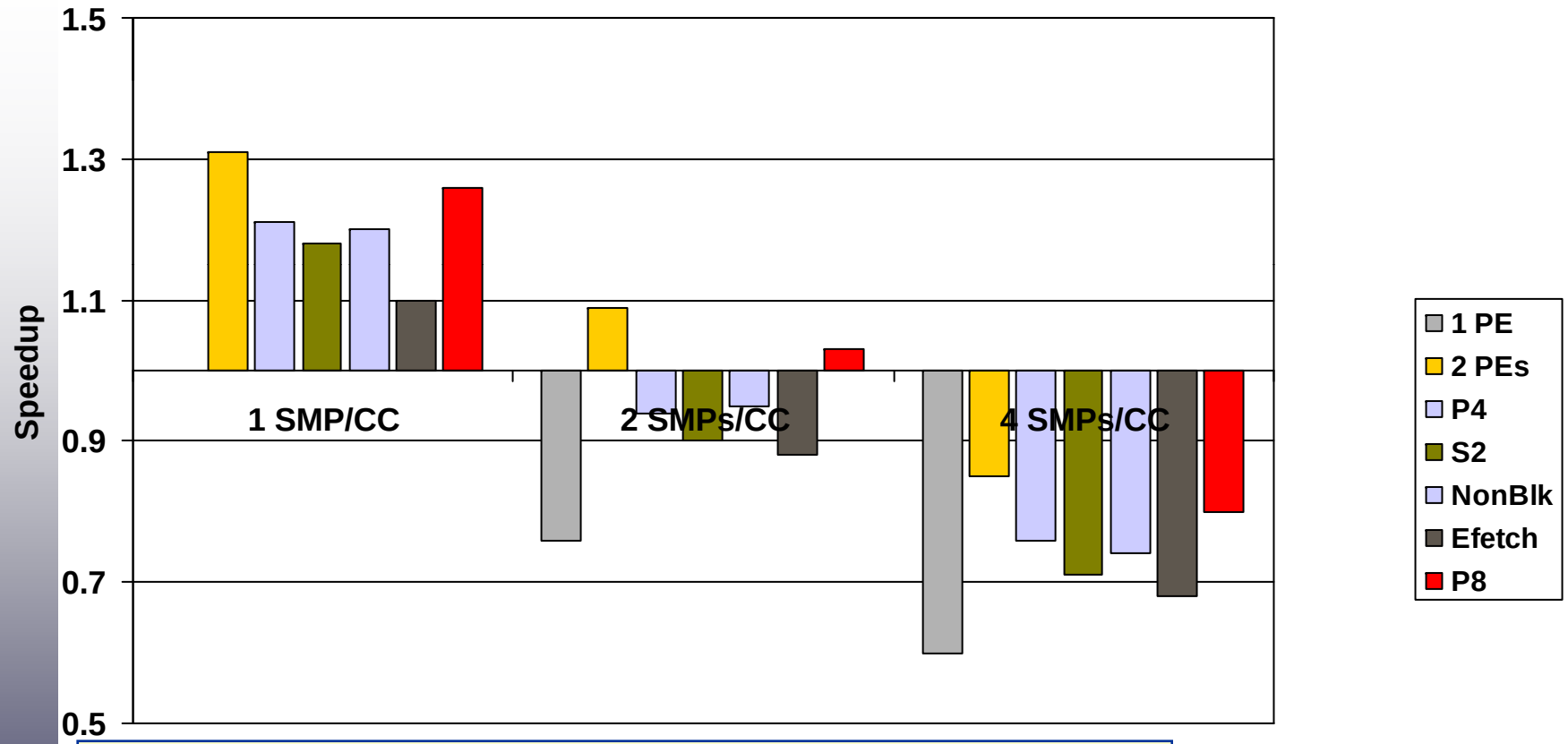
Techniques	Advantages	Disadvantages
Replication	• Simple: design, verify, test, & replicate	• More logic area • Shared path to Directory
Pipeline **Superpipeline**	• Higher throughput • Run at faster clock rate	• Larger overall latency & complexity • Limit directory cache size • Sensitive to dir. cache misses
Split-Streams	• No stall for response • Share PE resources	• Separate pending buffers • Two FSM units
Nonblocking	• Tolerate directory cache miss latency	• Non-blocking dir. cache • Pending queue & wiring
Early Fetch	• Hide dir. cache miss latency	• Extra wiring • Sensitive to timing of prefetches

CC Utilization



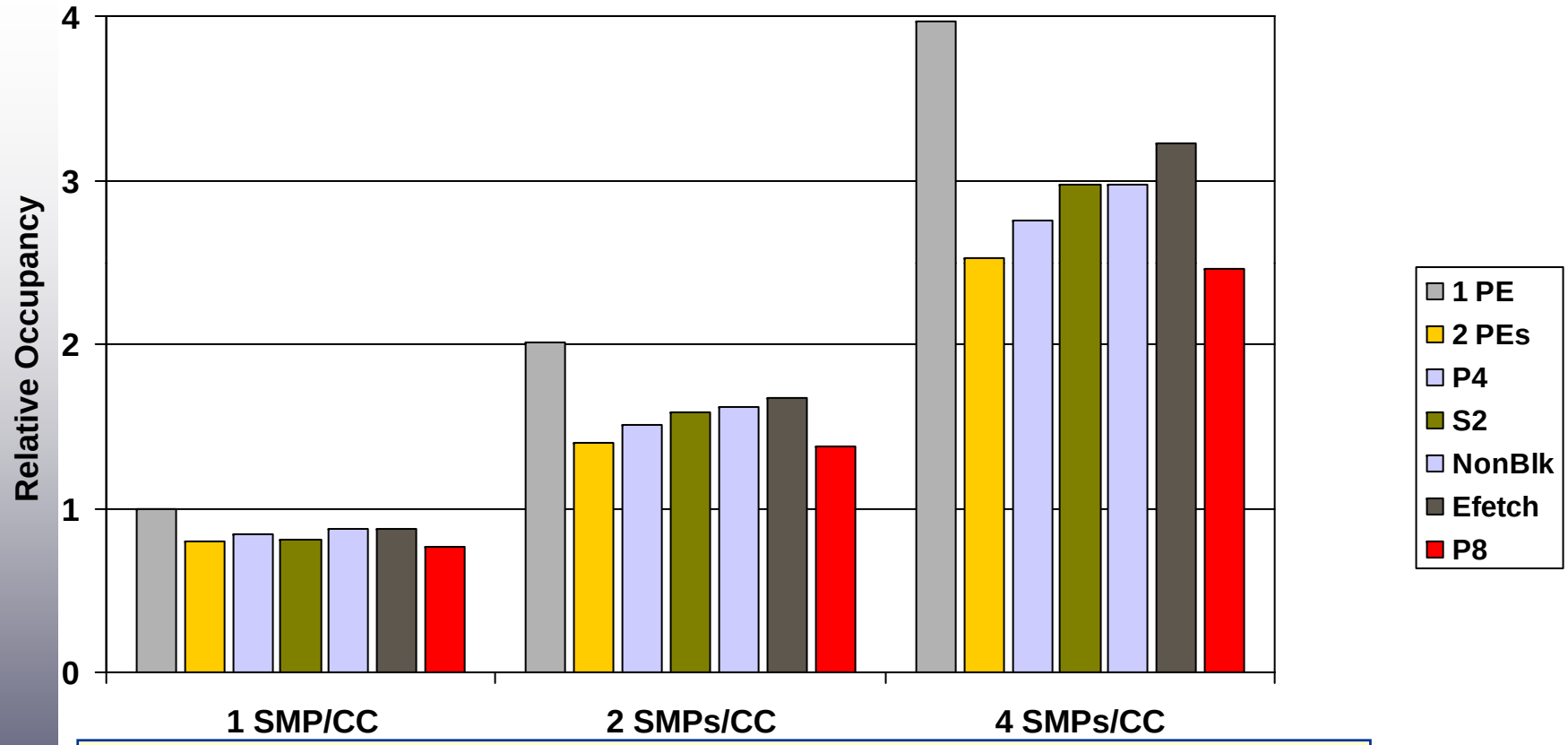
- **Memory-bound apps exhibit high CC utilization**

Single-Optimization: Ave. Speedup



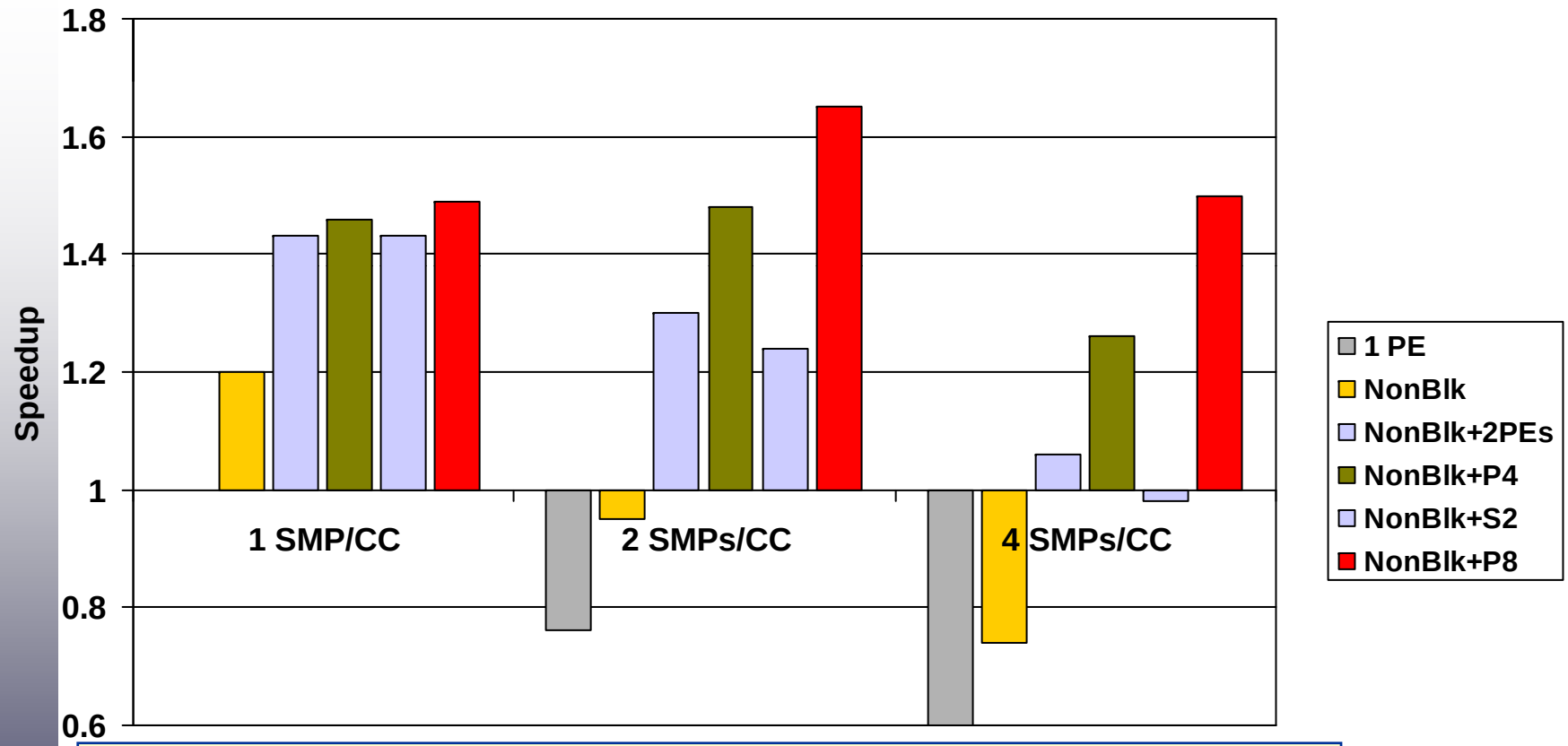
- **Slowdown for wide nodes with base CCs.**
- **2 PEs provide best speedups.**

Single-Optimization: Ave. Occupancy



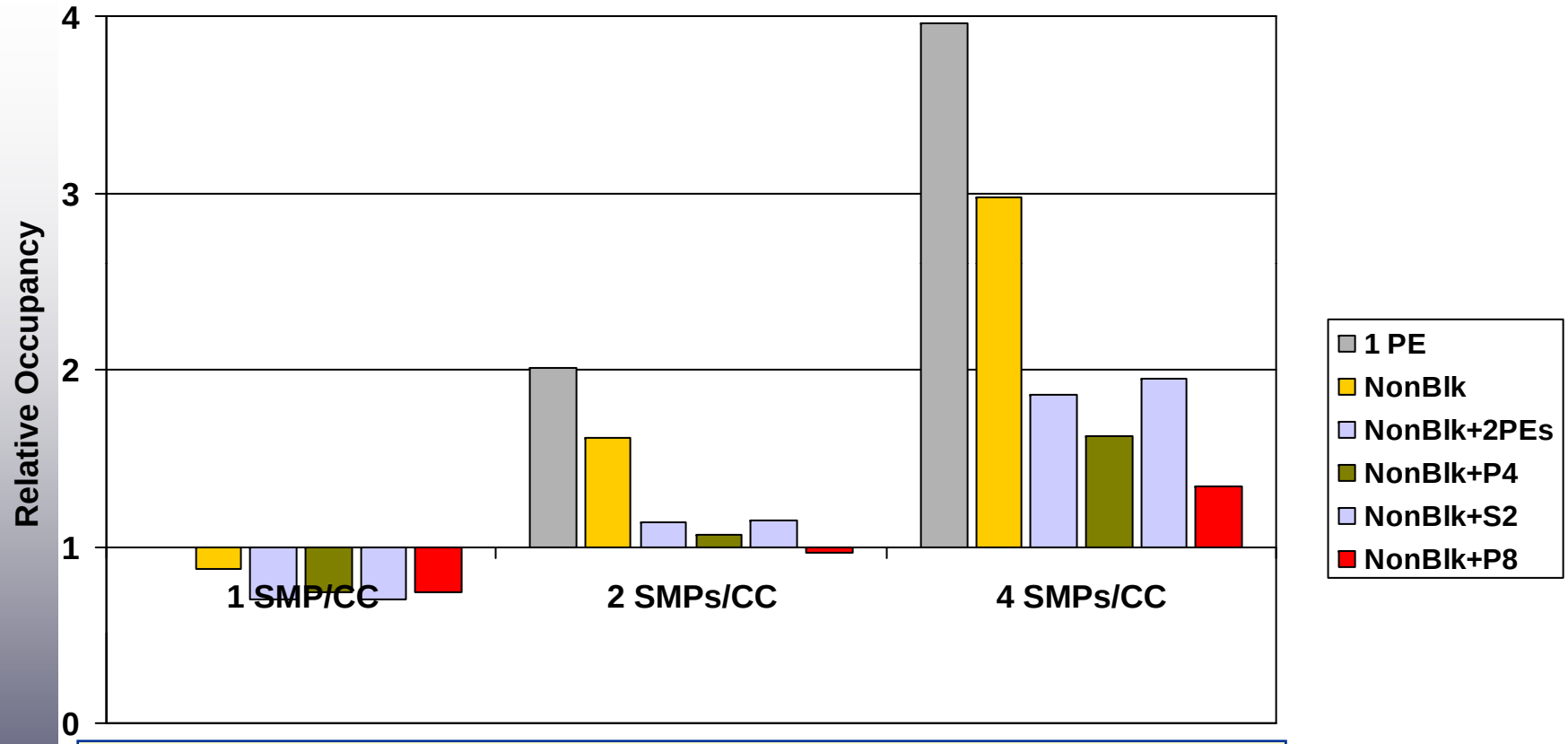
- **Occupancy increases as node size increases.**
- **Single optimization is insufficient for 4 SMPs/CC.**

Nonblock & 1 Opt – Ave. Speedup



- **Superpipeline & pipeline outperforms 2 PEs.**
- **4 SMPs/CC + superpipeline matches 1 SMP/CC.**

Nonblocking & 1 Opt – Ave. Occupancy



- **Superpipeline significantly reduces occupancy.**

Summary

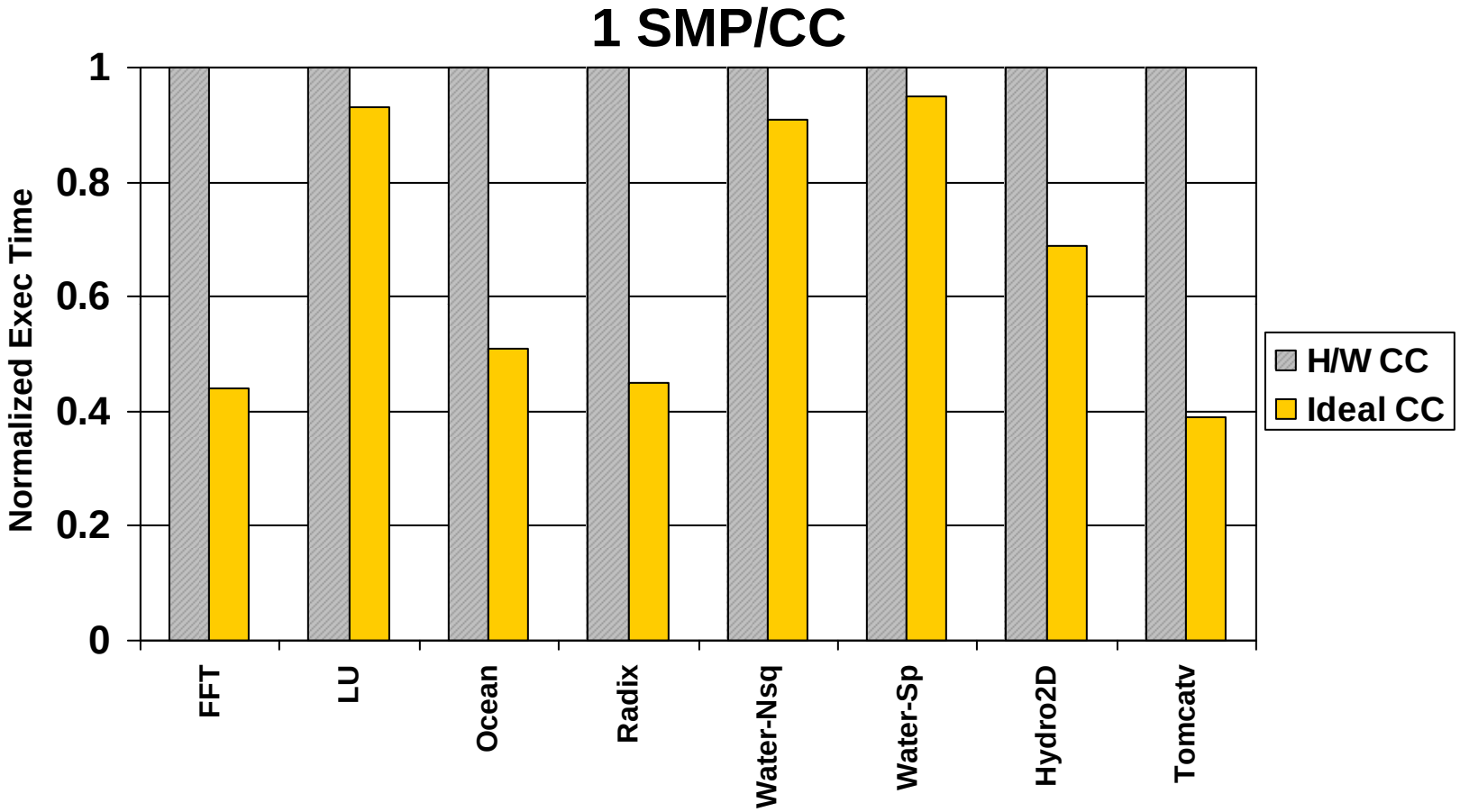
- Six throughput-enhancing techniques:
  - Multiple PEs
  - Pipelined PE
  - Split request-response streams
  - Nonblockingexecution
  - Directory early fetch
  - Superpipelined PE
- Pipeline & superpipeline benefit most from Nonblocking
- *Single Optimization*: Multiple PEs or Superpipelining offers highest speedups
- *Two Optimizations*: Nonblocking+Superpipelining allows wider nodes with competitive performance, lower system cost.

Backup Slides

System Parameters

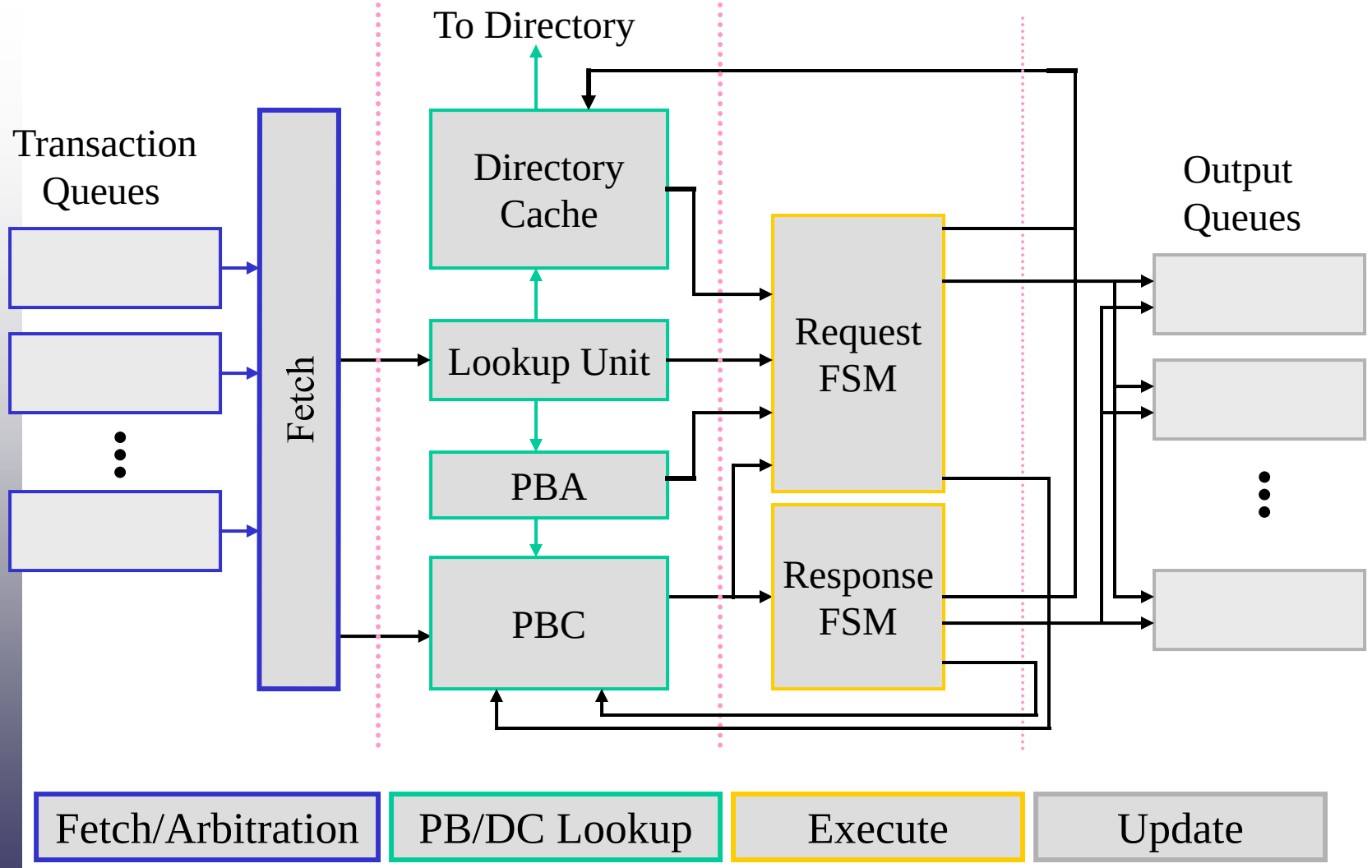
CPU	1GHz, 6-issue, OOO CPUs
Functional Units	3 integer, 3 FP units, 2 unified LD/ST units
Branch Prediction	2K 2-bit saturating counters
L1	16KB, 4-way, LRU, 64B line
L2	64KB, 4-way, LRU, 64B line, multiported
L3	512KB/1MB/2MB, 4-way, LRU, 64B line
SMP Bus	250MHz, 16B wide, fully pipelined, split-transaction, separate address & data buses
Network	250MHz, pipelined, 16B flit
CC	250MHz CC, connecting to 1 to 4 SMP modules
Nodes	16 1-SMP nodes, 8 2-SMP nodes, 4 4-SMP nodes

Hardwired vs. Ideal CCs

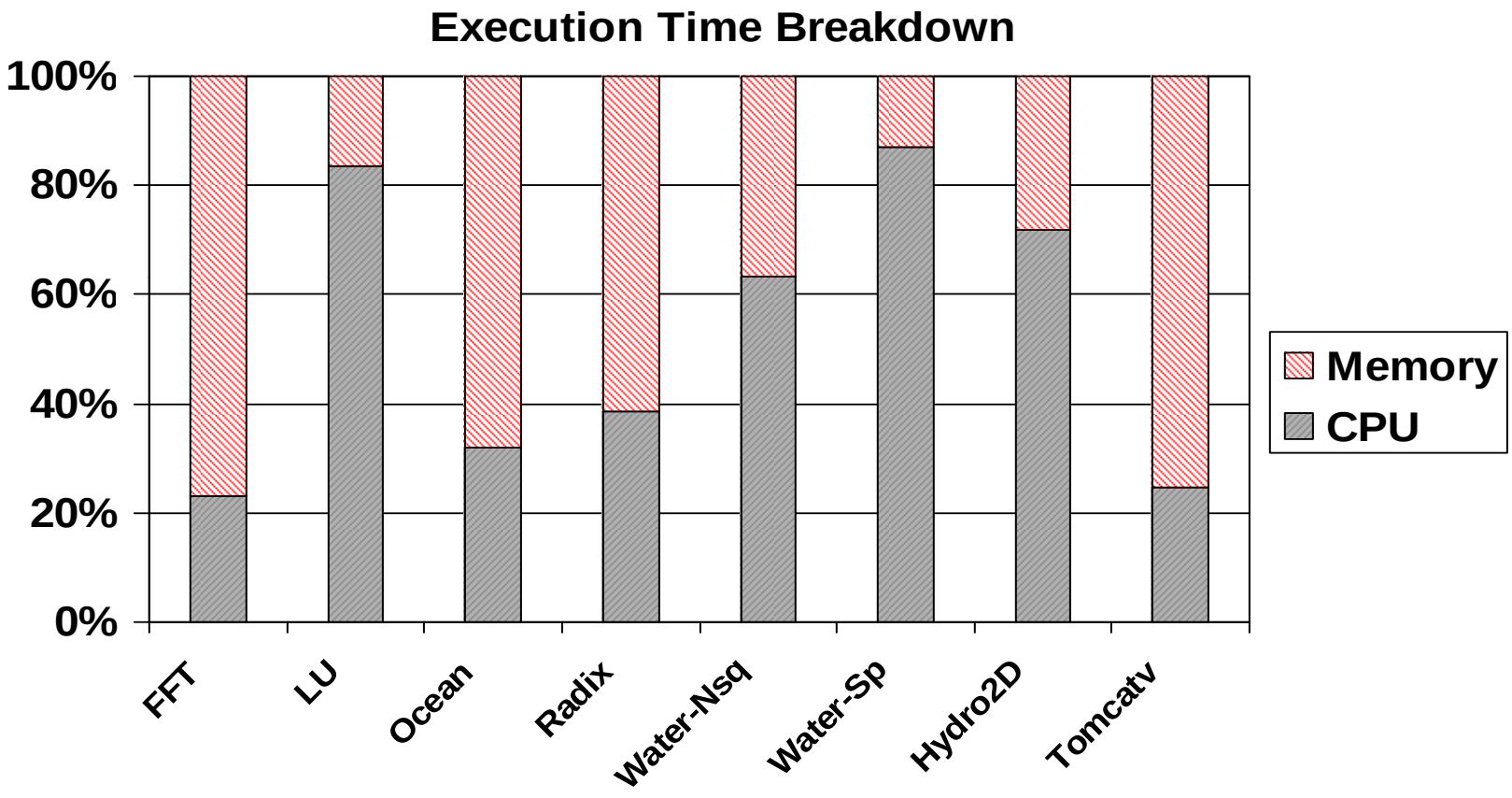


- **Wide disparity in performance for 4 apps.**

Split Streams & Pipelining



Applications



- **CC optimizations will benefit FFT, Ocean, Radix, & Tomcatv.**

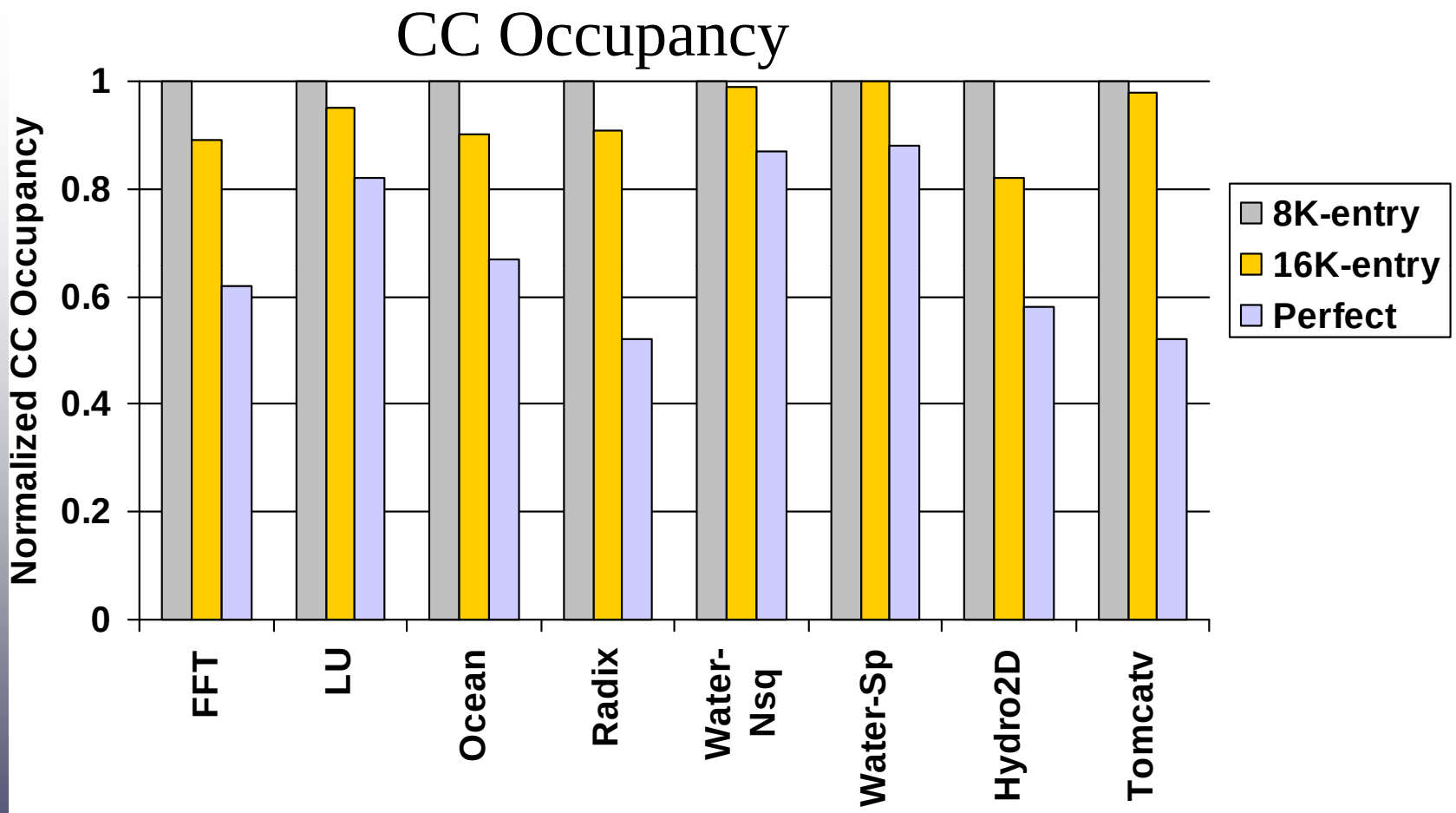
DC Miss Ratio -- 1 SMP/CC

Applications	8K entries	16K entries
FFT	48.66	32.15
LU	4.63	2.73
Ocean	37.88	26.76
Radix	65.87	55.48
Water-Nsq	5.66	4.31
Water-Sp	2.04	1.72
Hydro2D	51.27	27.22
Tomcatv	62.22	55.54

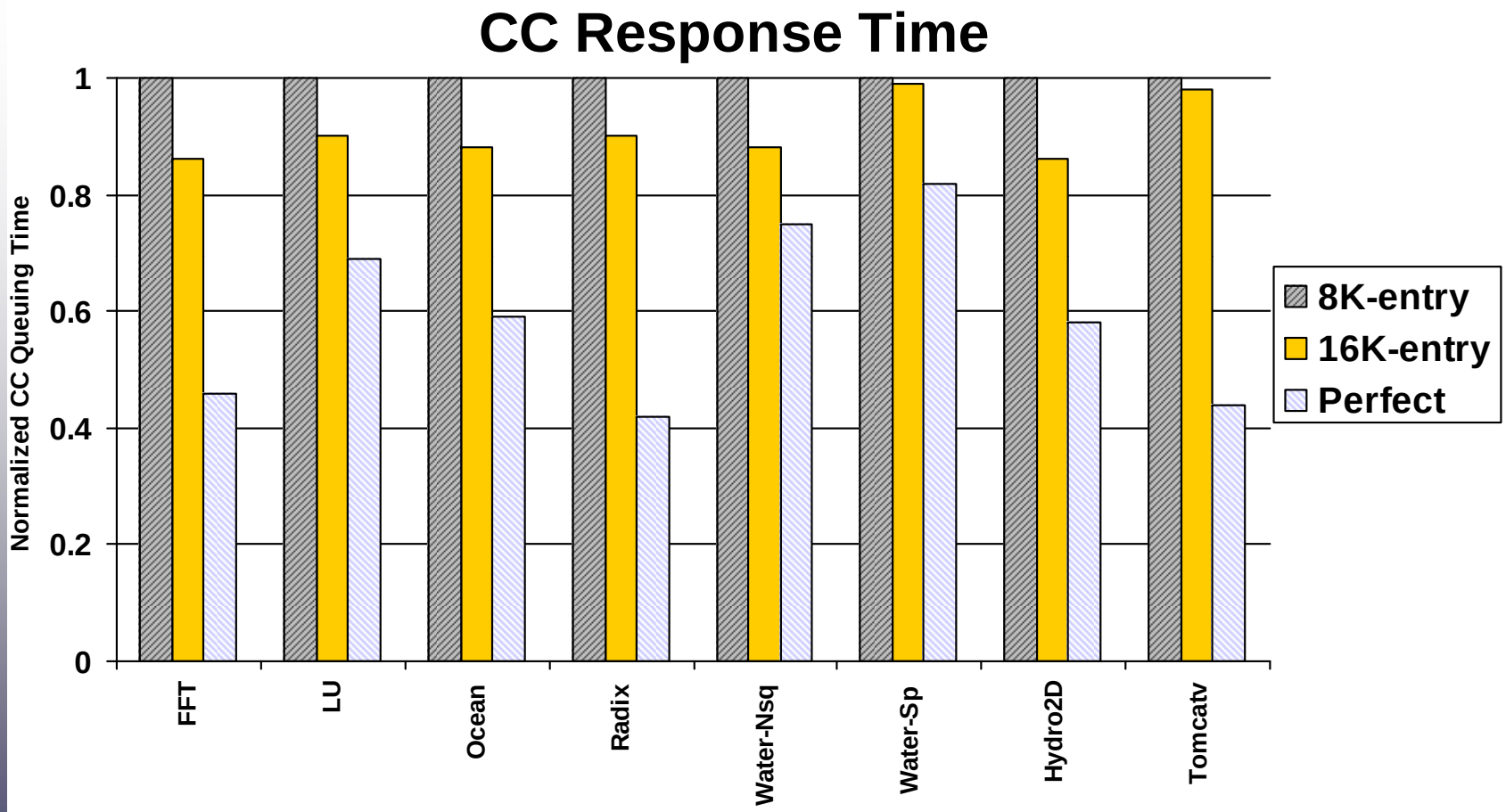
DC Miss Ratio -- 4 SMPs/CC

Applications	8K entries	16K entries
FFT	68.23	45.59
LU	4.83	2.86
Ocean	59.33	39.63
Radix	77.22	64.25
Water-Nsq	2.29	1.82
Water-Sp	2.77	2.41
Hydro2D	69.22	38.65
Tomcatv	68.51	58.84

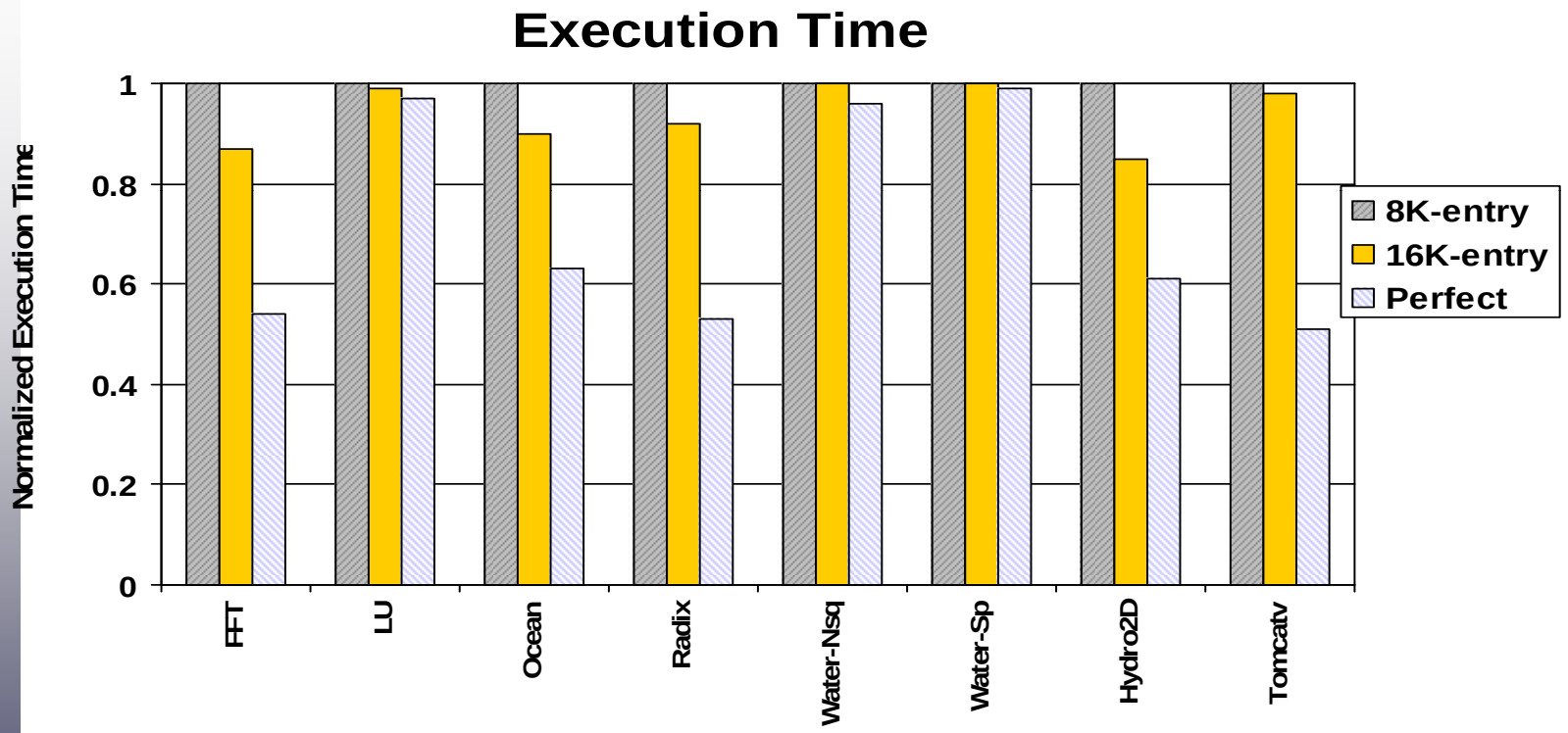
DC sizes & CC Occupancy



DC sizes & CC Queuing Time



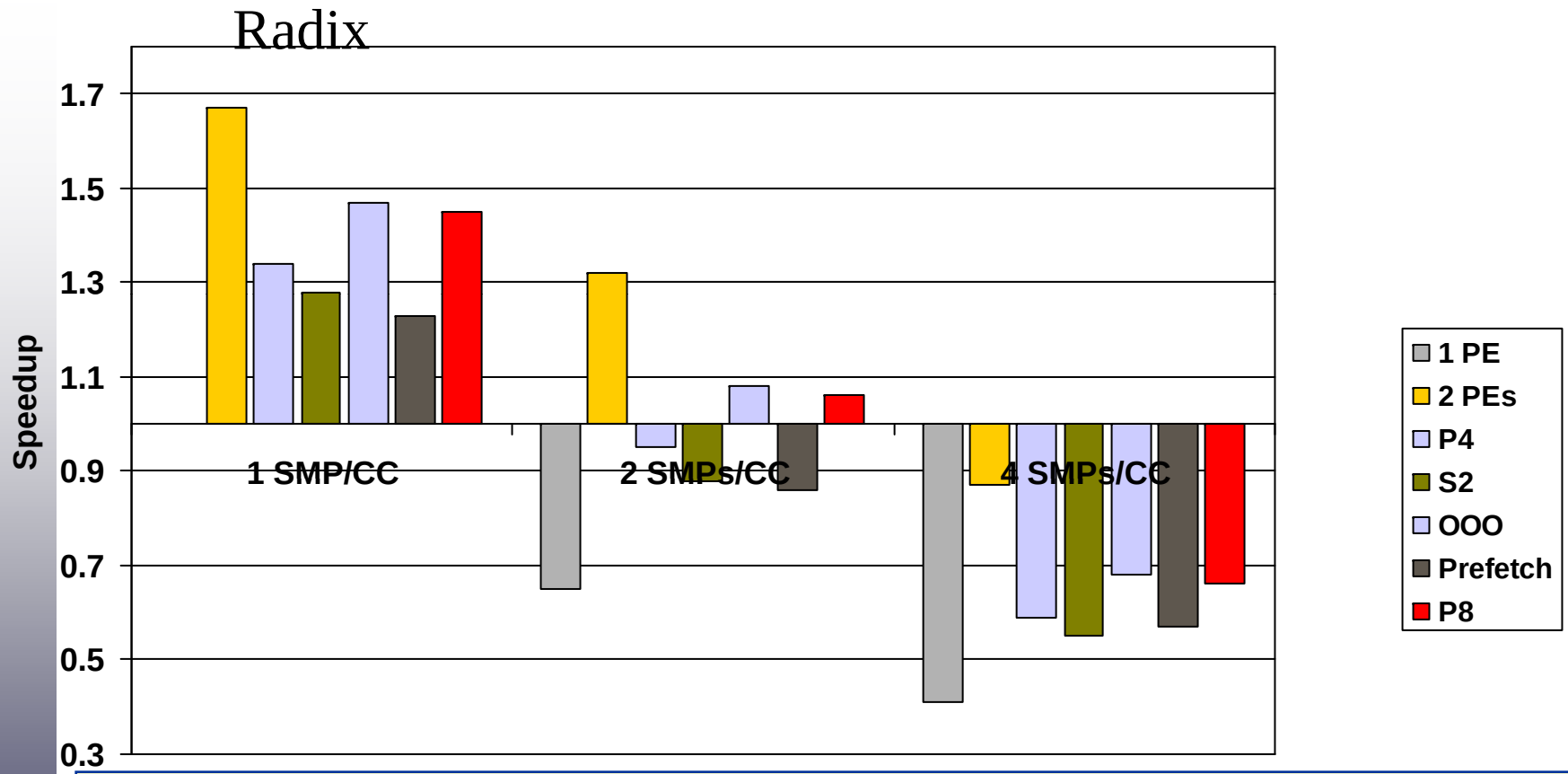
DC Sizes vs. Perfect DC



Publication

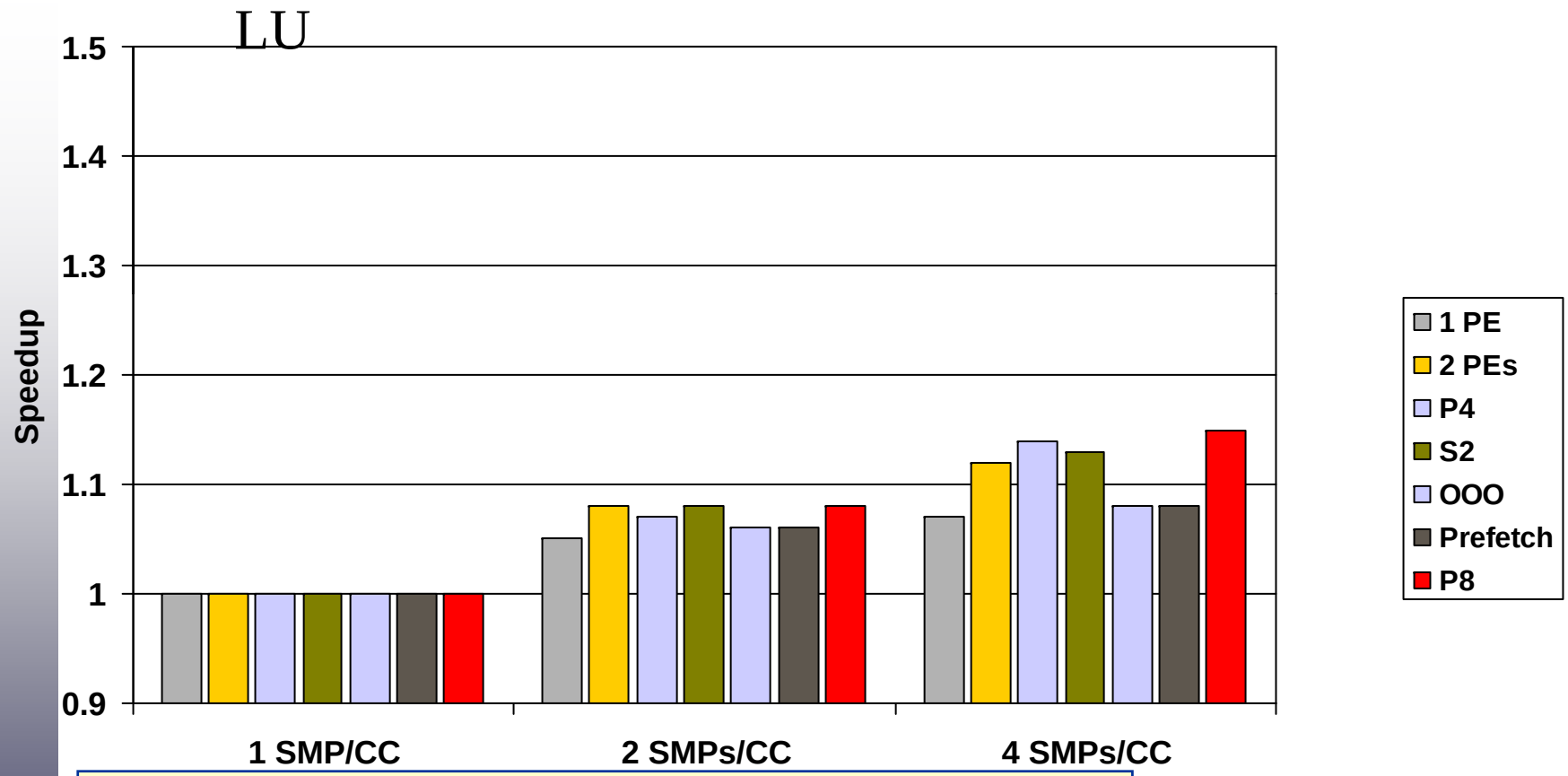
1. A-T. Nguyen, J. Torrellas. *Design Trade-offs in High-Throughput Coherence Controllers*, Submitted for publication, 2003.
2. A-T. Nguyen. *High-throughput Coherence Controllers*, Ph.D. thesis, University of Illinois. Also published as Technical Report UIUCDCS-R-2001-2254. December 2001.
3. A. Nanda, A-T. Nguyen, M. Michael, D. Joseph. *High-throughput Coherence Control and Hardware Messaging in Everest*, IBM Journal of Research and Development, Vol. 45, Number 2, 2001.
4. A. Nanda, A-T. Nguyen, M. Michael, D. Joseph. *High-Throughput Coherence Controllers*, Sixth International Symposium on High-Performance Computer Architecture (HPCA-6), 2000.
5. J. Torrellas, L. Yang, A-T. Nguyen. *Toward a Cost-Effective DSM Organization that Exploits Processor-Memory Integration*, Sixth International Symposium on High-Performance Computer Architecture (HPCA-6), 2000.
6. L. Yang, A-T. Nguyen, J. Torrellas. *How Processor-Memory Integration Affects the Design of DSMs, Workshop on Mixing Logic and DRAM: Chips that Compute and Remember*, 1997.
7. A-T. Nguyen, M. Michael, A. Nanda, K. Ekanadham, P. Bose. *Accuracy and Speedup of Parallel Trace-Driven Architectural Simulation*, Proceedings of the 11th International Parallel Processing Symposium (IPPS'97), 1997.
8. A-T. Nguyen, J-D. Wellman, P. Bose. *PARSIM: A Parallel Trace-Driven Simulation Facility for Fast and Accurate Performance Analysis Studies*, Proceedings of the 1997 IEEE International Performance, Computing, and Communications Conference (IPCCC'97), 1997.
9. A-T. Nguyen, M. Michael, A. Sharma, and J. Torrellas, *The Augmint Multiprocessor Simulation Toolkit for Intel x86 Architectures*, Proceedings of the 1996 IEEE International Conference on Computer Design (ICCD'96), 1996.

Single-Optimization Speedup



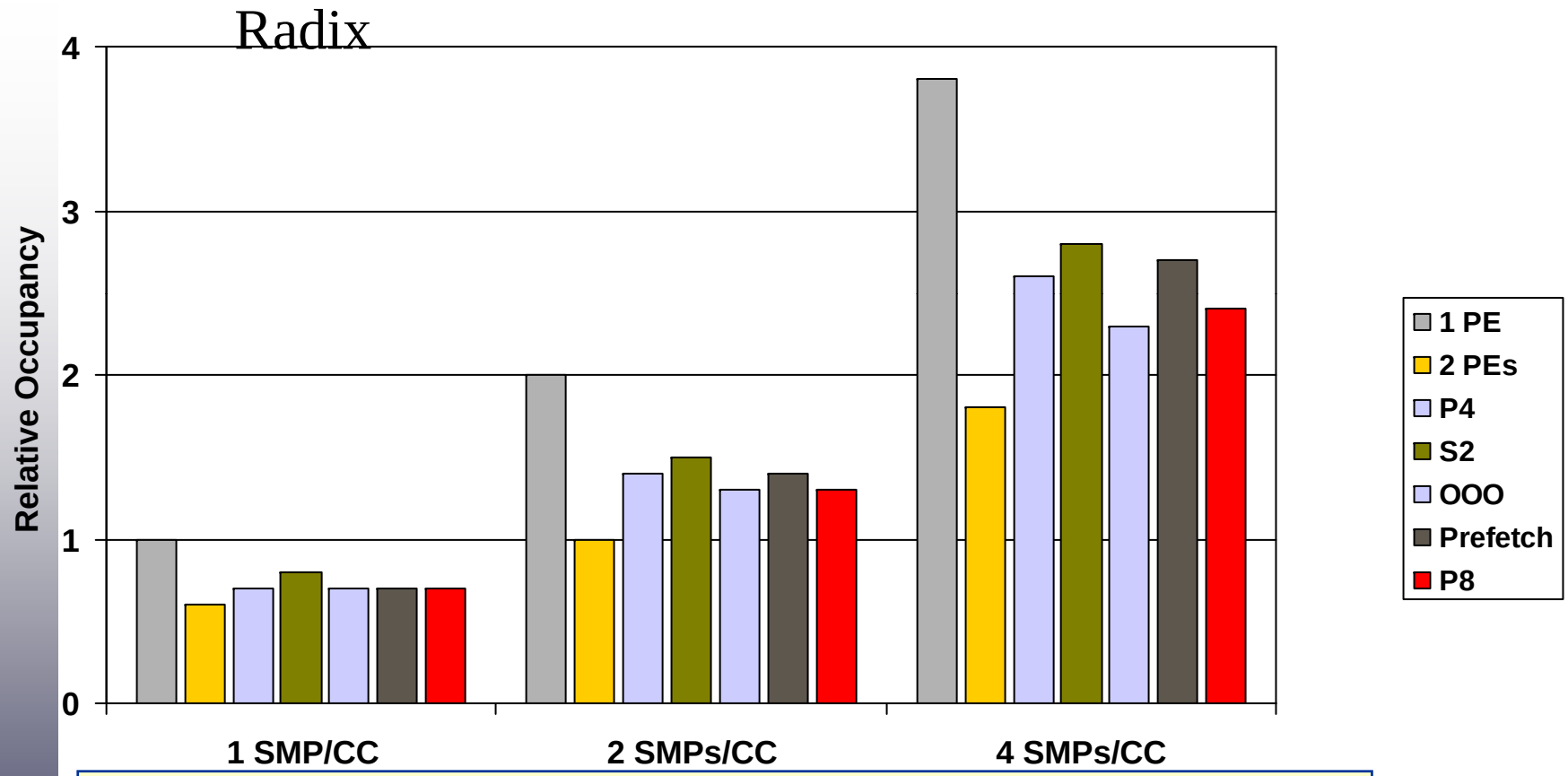
- **Slowdown for wide nodes with base CCs.**
- **Optimizations improve speedups within same organization.**

Single-Optimization Speedup



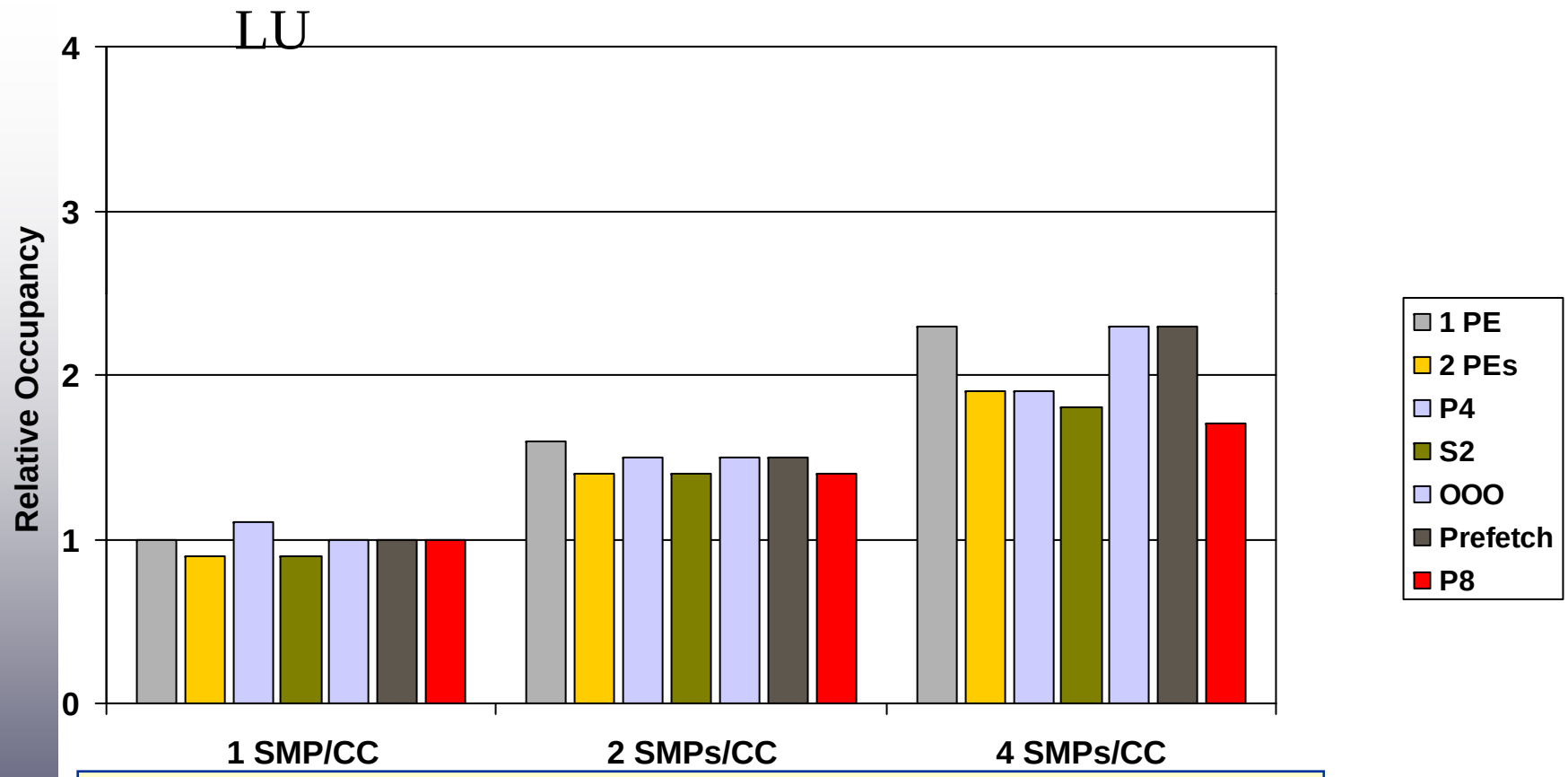
- **Speedup for wide nodes with base CCs.**
- **Single optimization offers no gain.**

Single-Optimization Occupancy



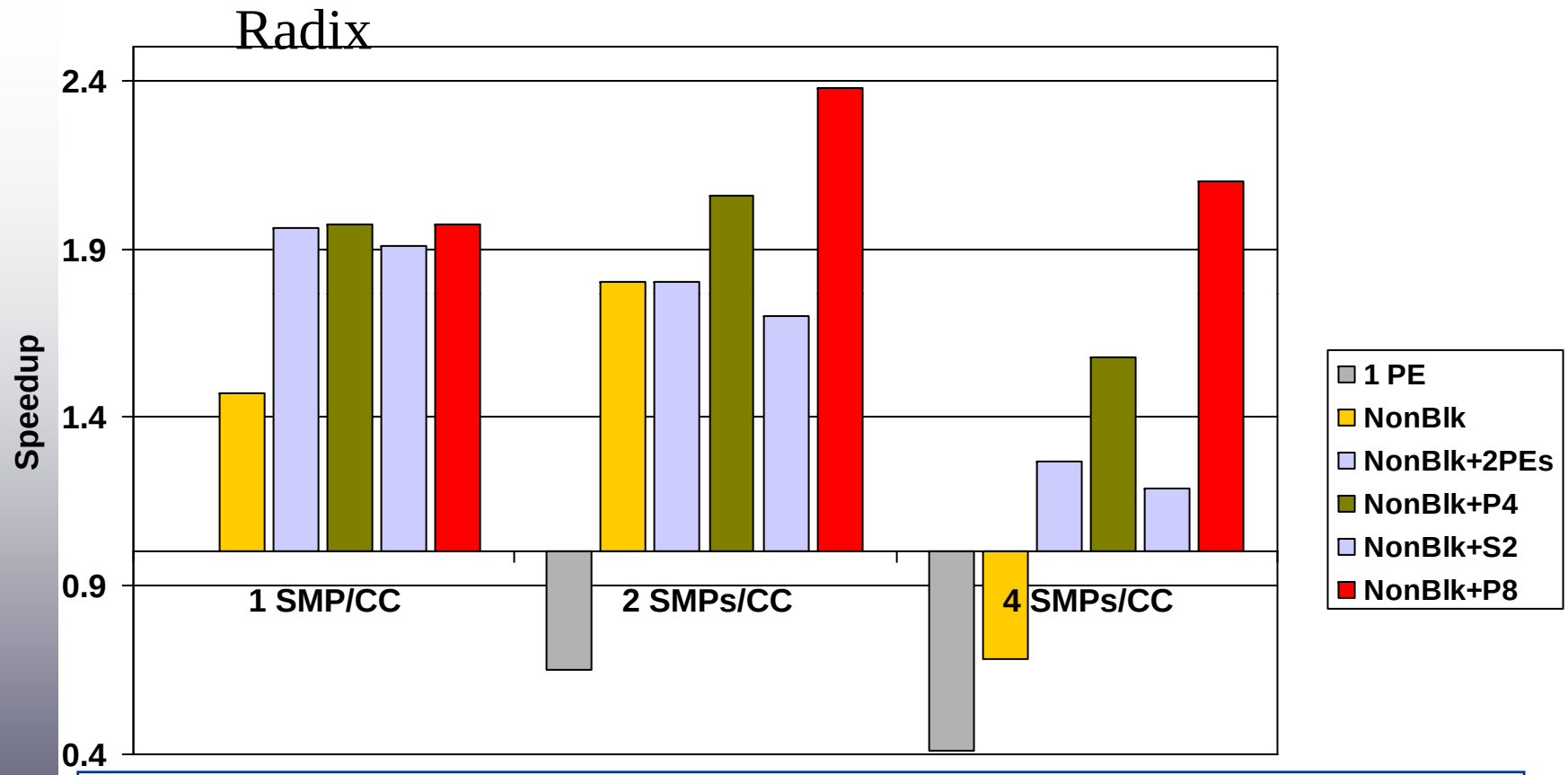
- **Occupancy increases as node size increases.**
- **Single optimization is insufficient for 4 SMPs/CC.**

Single-Optimization Occupancy



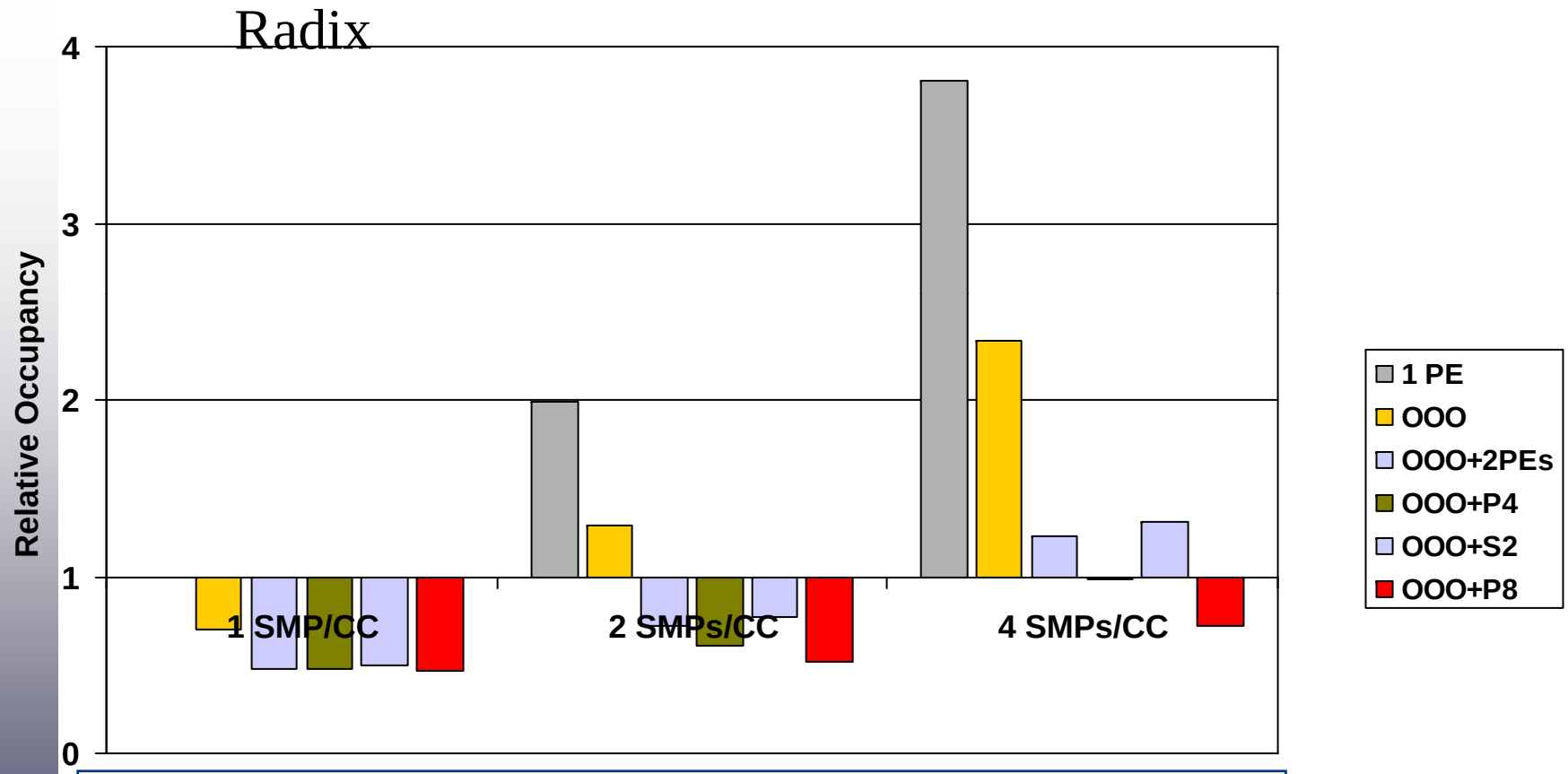
- **Locality of access outweighs increases in traffic.**

Nonblocking & 1 Opt – Speedup



- **Superpipeline & pipeline outperforms 2 PEs.**
- **Wide nodes with superpipeline outperforms 1 SMP/CC.**

Nonblocking & 1 Opt – Occupancy



- **Superpipeline significantly reduces occupancy.**