
Taming Shared Work To Maximize Query Throughput

Ryan Johnson

Nikos Hardavellas, Ippokratis Pandis, Stavros
Harizopoulos**, *Anastasia Ailamaki, *Babak Falsafi

*EPFL

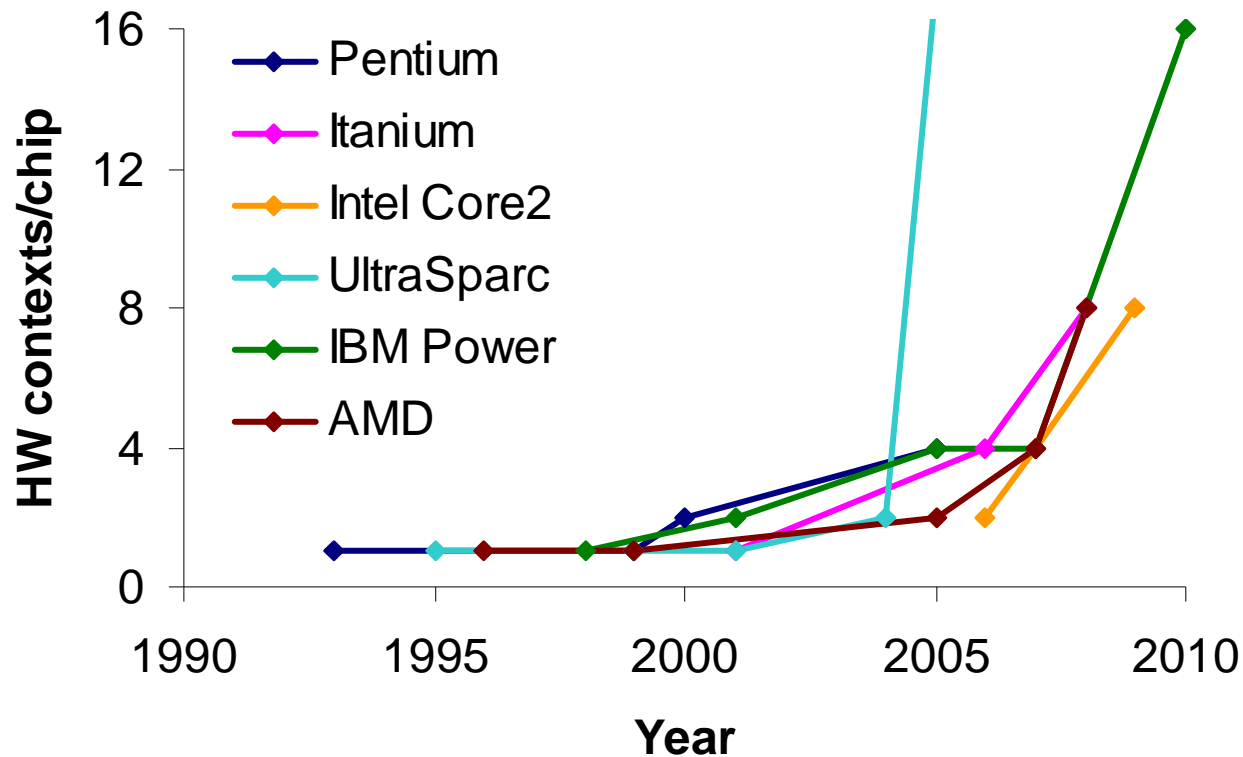
Carnegie Mellon University

**HP LABS

CarnegieMellon

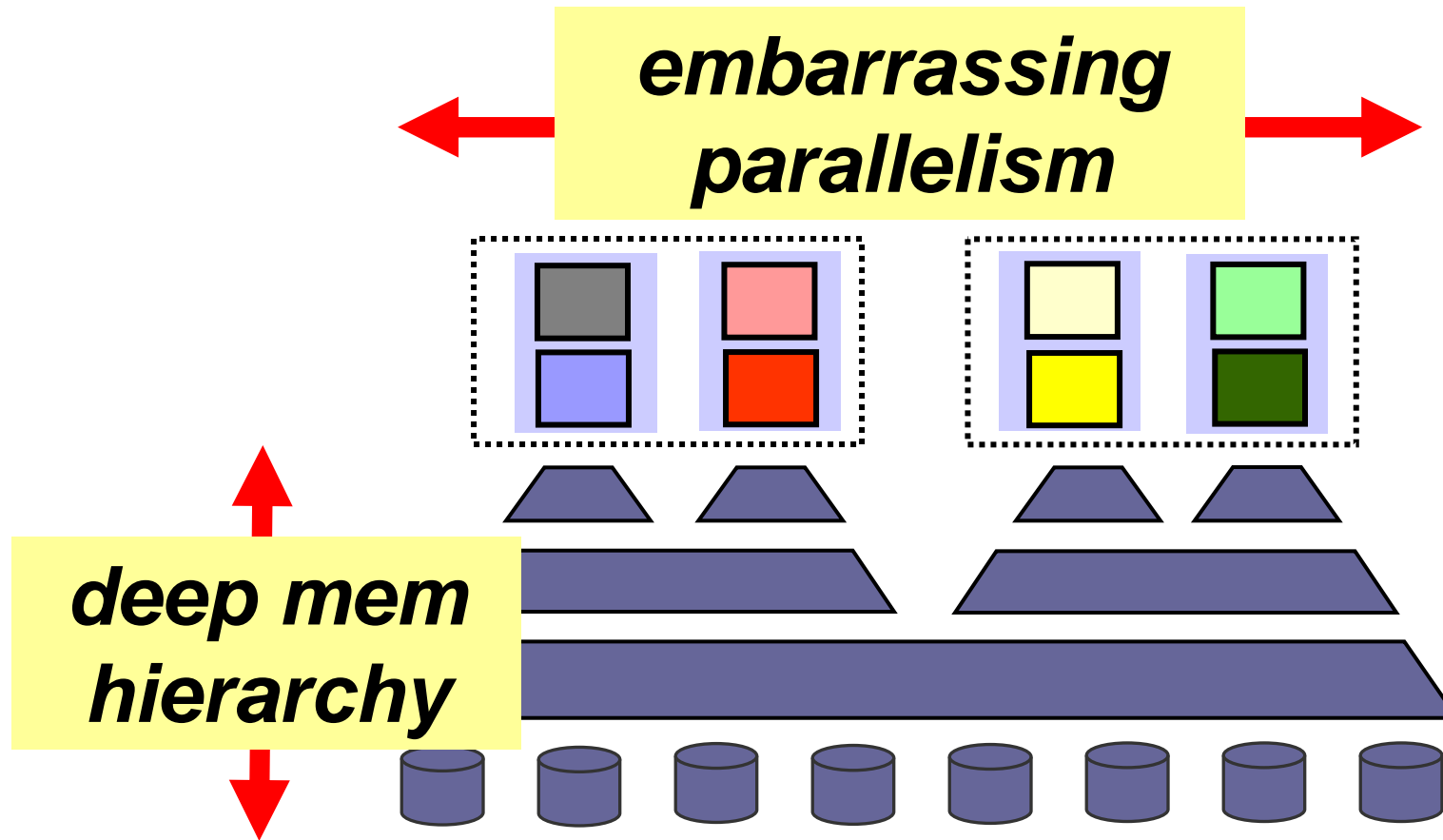


Multi-core Has Arrived

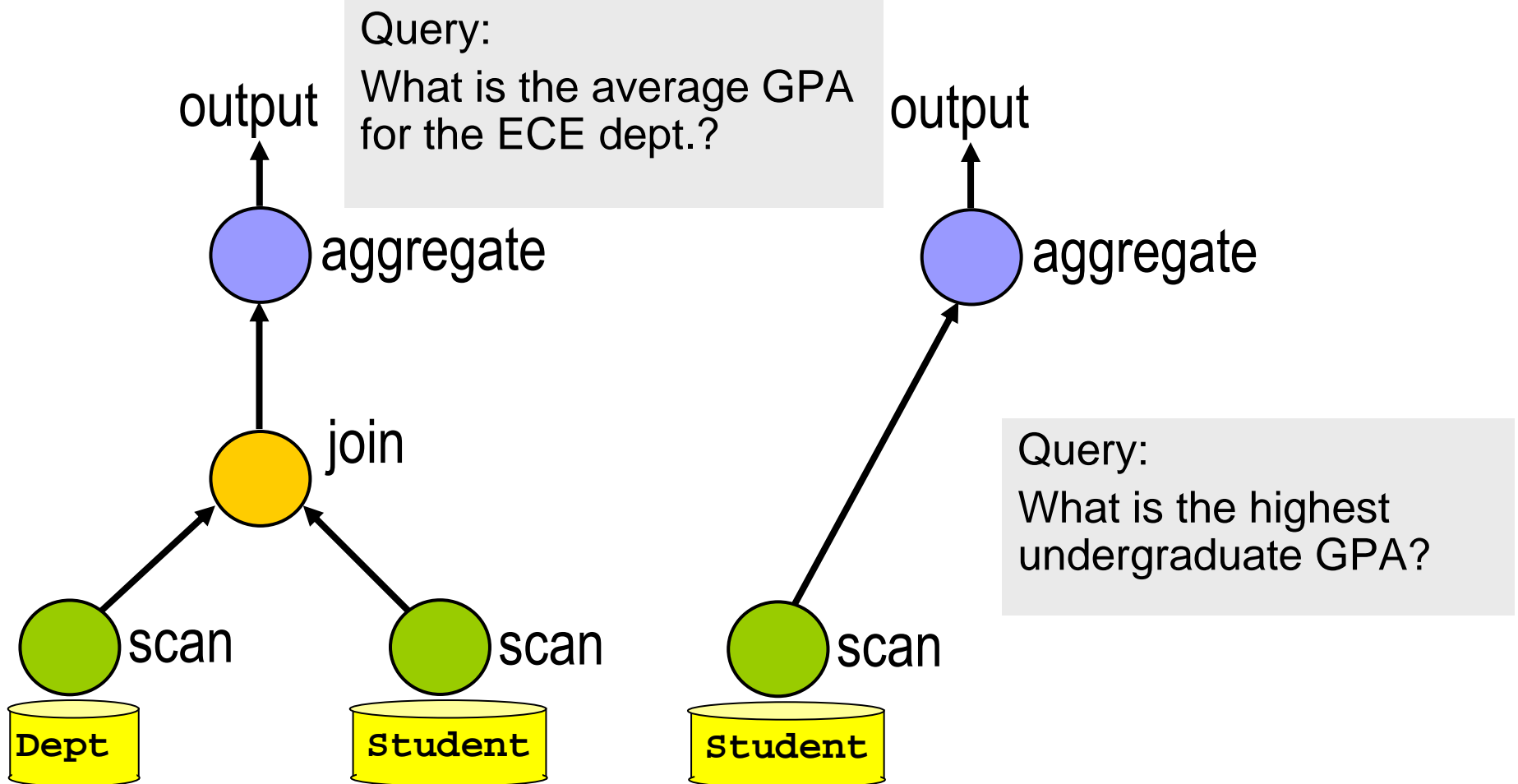


- Moore's Law giving us cores, not performance
 - Shifts performance burden to software

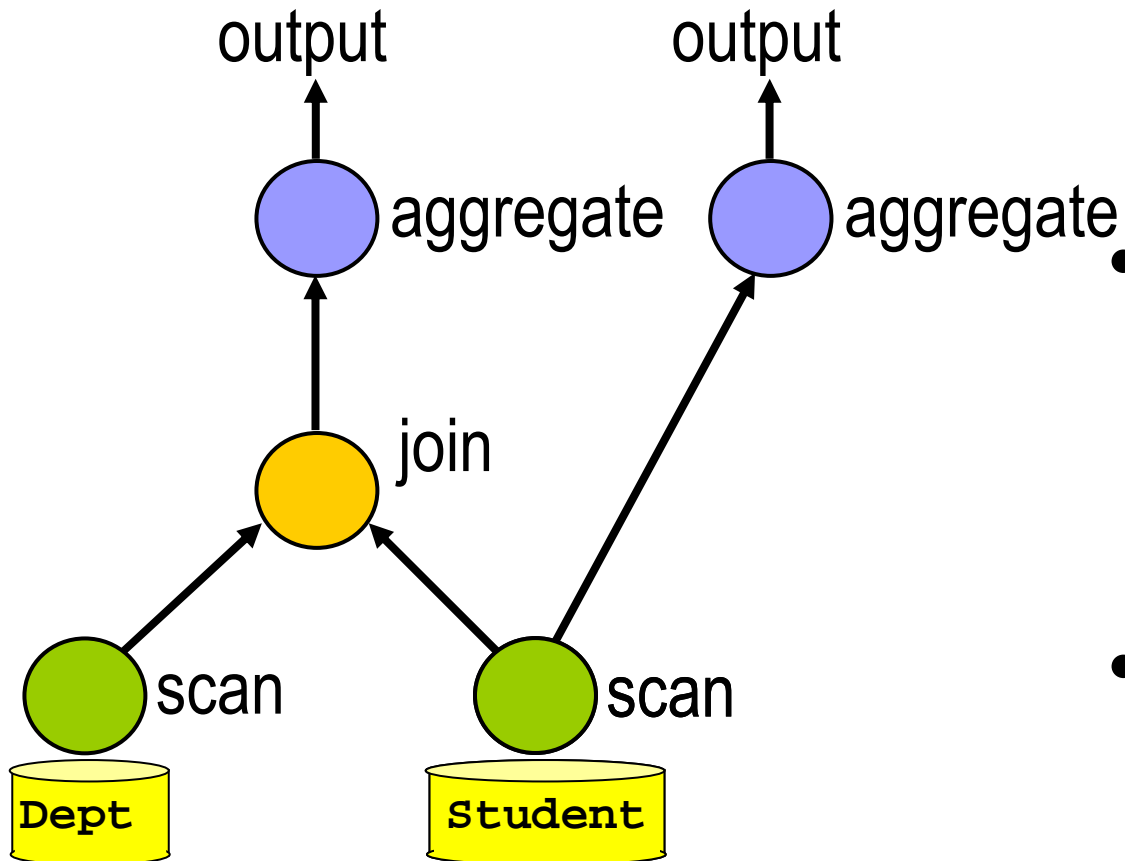
Challenges for DBMS



Work Sharing



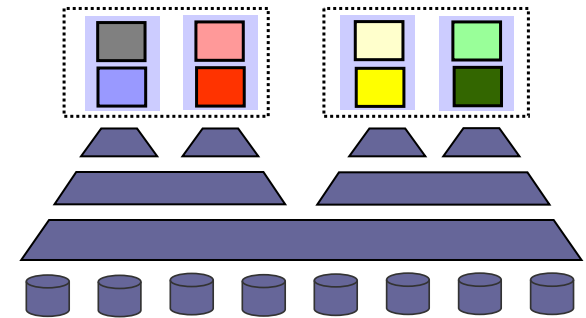
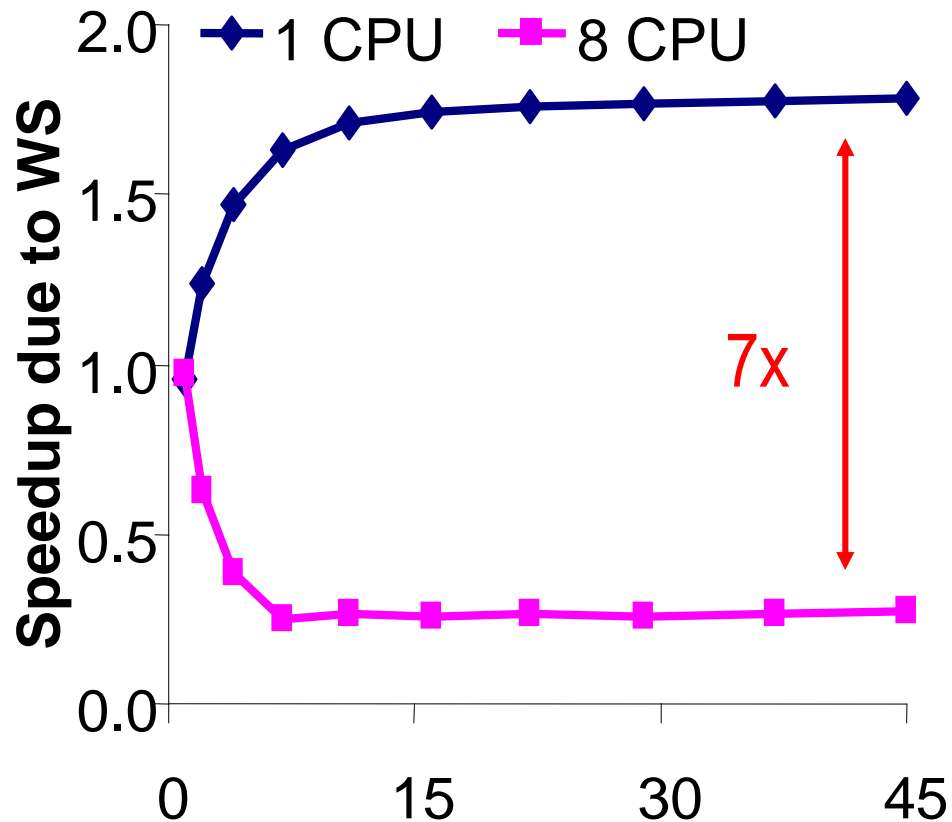
Work Sharing



- Many queries in system
 - Similar requests
 - Redundant work
- Work Sharing
 - Detect redundant work
 - Compute results once and share
- Big win for I/O, uniprocessors

➡ 2x speedup for TPC-H queries [hariz05]

Work Sharing on Modern Hardware



- ✓ Deep hierarchy
- ✗ High Parallelism

➡ Work sharing can actually hurt performance!

Contributions

- **Observation**
 - Work sharing can hurt performance on parallel hardware
- **Analysis**
 - Identify trade-off between total work, critical path
- **Application**
 - Adaptive work sharing policy significantly outperforms static all-or-nothing policies

Outline

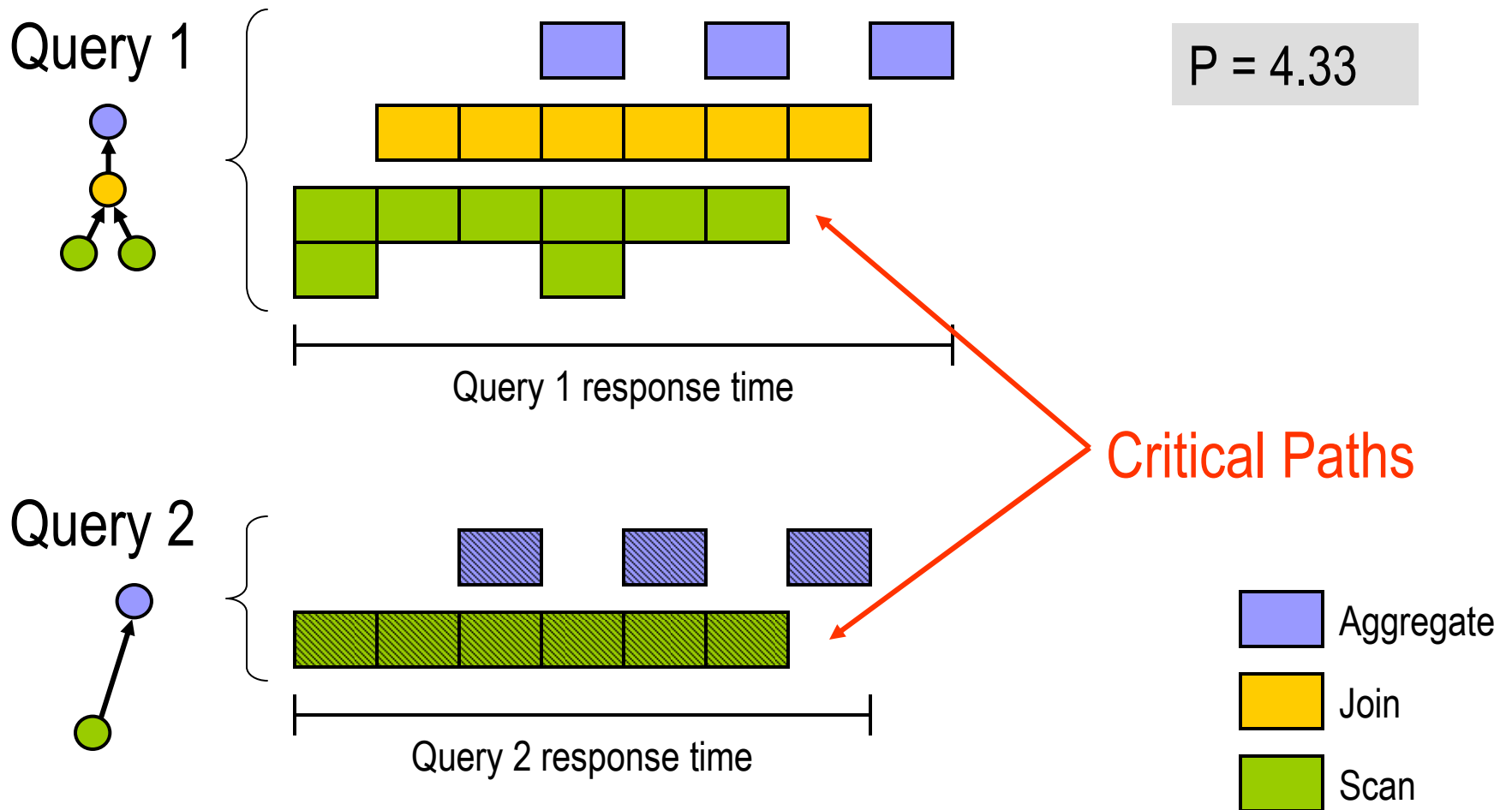
- Introduction
- **Understanding Work Sharing**
- Analysis and Experiments
- Work Sharing and Working Set Size

Challenges of Exploiting Work Sharing

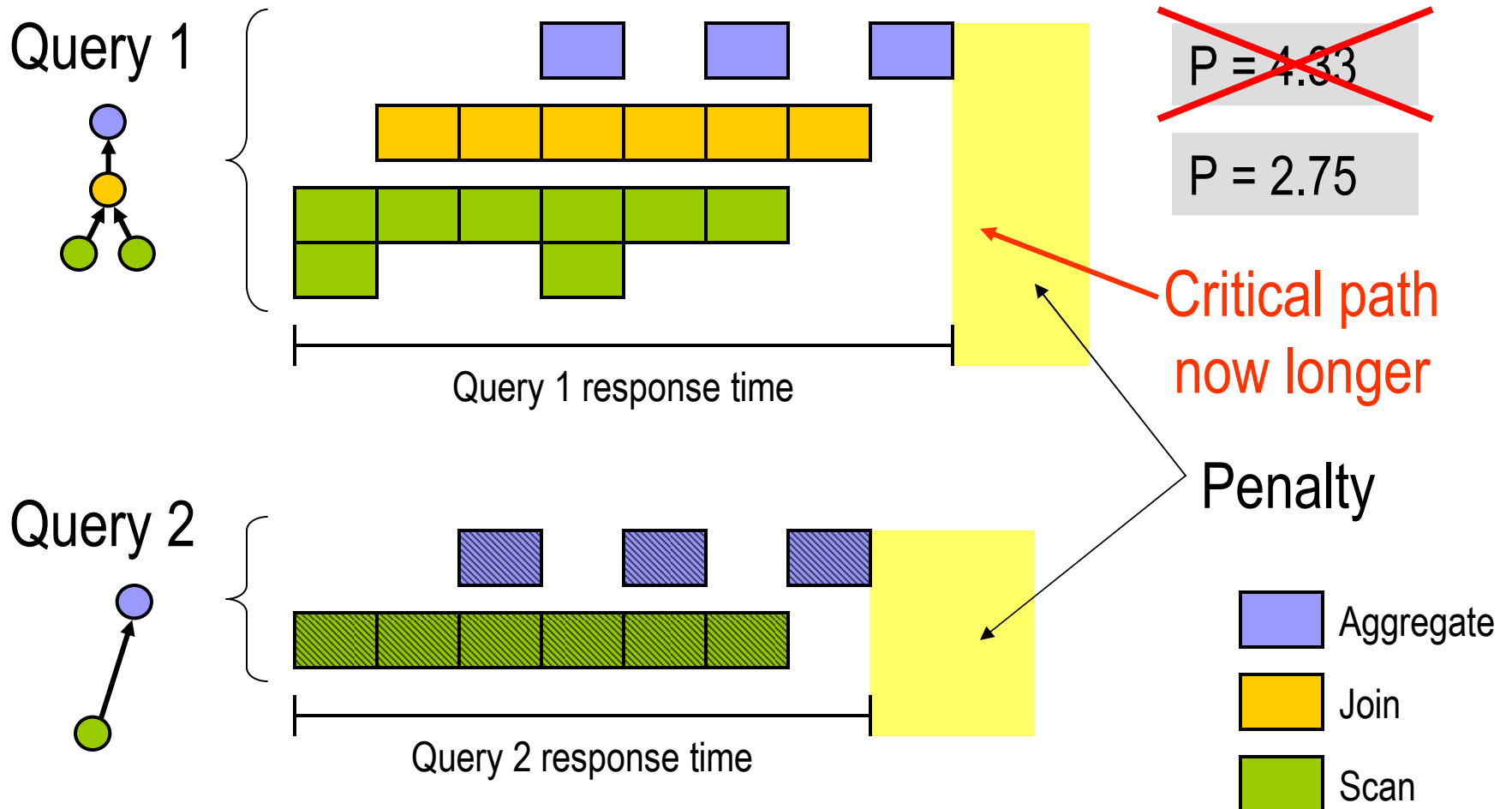
- Independent execution only?
 - Load reduction from work sharing can be useful
- Work sharing only?
 - Indiscriminate application can hurt performance
- To share or not to share?
 - System and workload dependent
 - Adapt decisions at runtime

➡ Must understand work sharing to exploit it fully

Work Sharing vs. Parallelism



Work Sharing vs. Parallelism



➡ Total work and critical path both important



Understanding Work Sharing

- Performance depends on two factors:

$$\textit{Throughput} = f\left(\frac{1}{\textit{TotalWork}}, \frac{1}{\textit{CriticalPath}}\right)$$

- Work sharing presents a trade-off
 - Reduces total work
 - Potentially lengthens critical path

➡ Balance both factors or performance suffers

Outline

- Introduction
- Understanding Work Sharing
- **Analysis and Experiments**
- Work sharing and working set size

Exploring WS vs. Parallelism

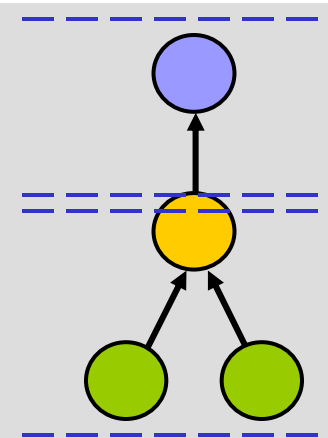
- Work sharing splits query into three parts
 - Independent work
 - Per-query, parallel
 - Total work
 - Serial work
 - Per-query, serial
 - Critical path
 - Shared work
 - Computed once
 - “Free” after first query

Example:

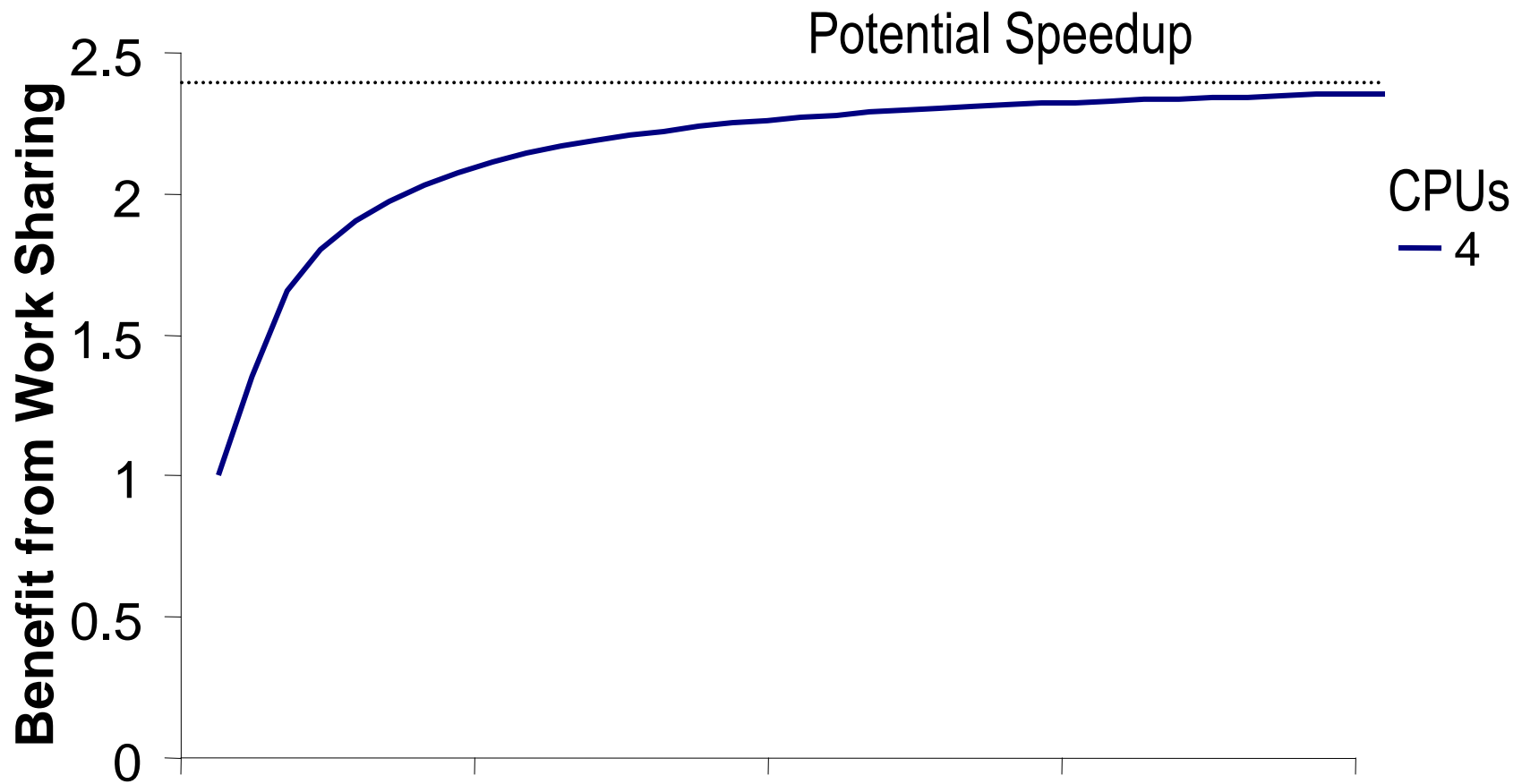
Independent - 37%

Serial - 4%

Shared - 59%

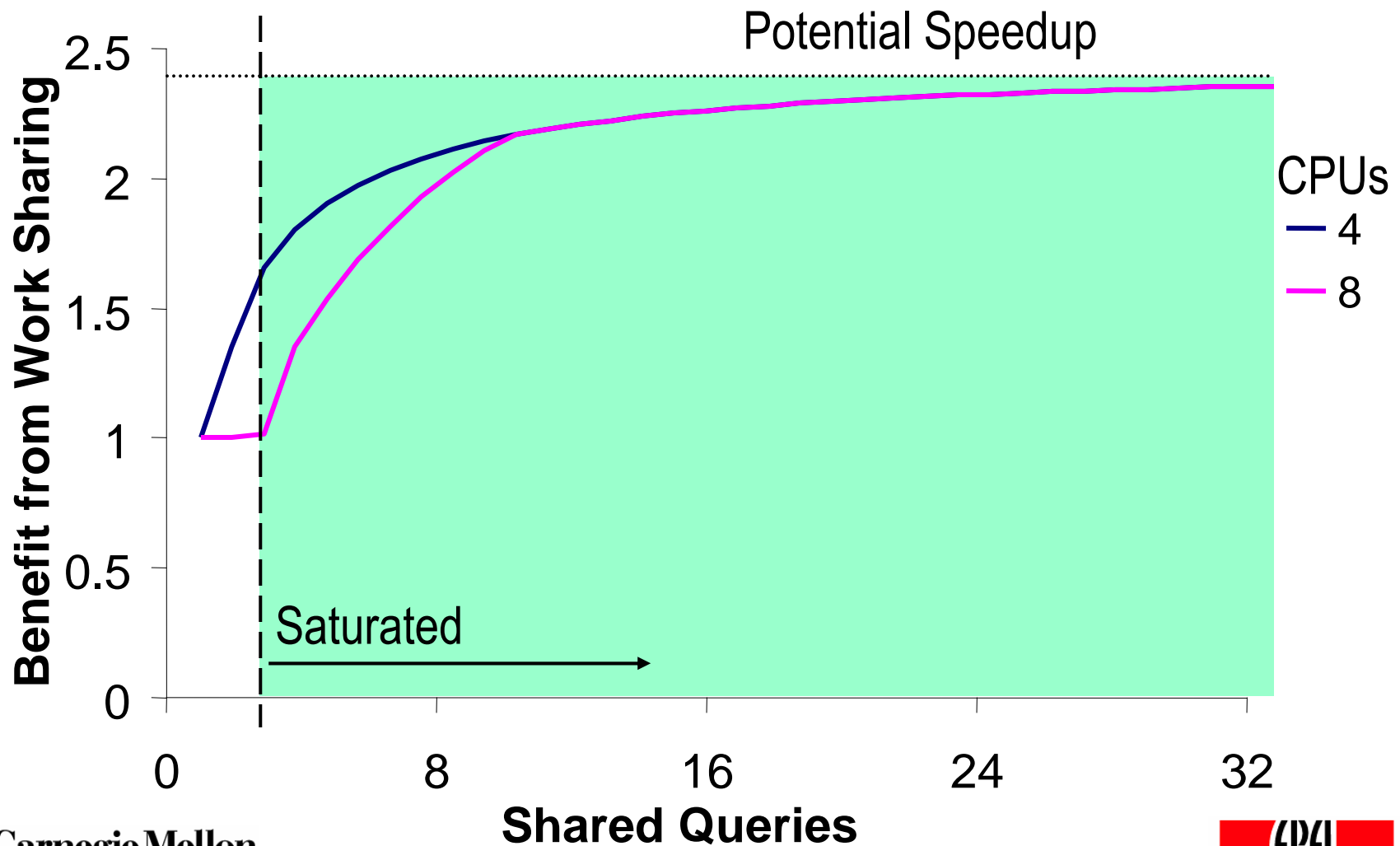


Exploring WS vs. Parallelism

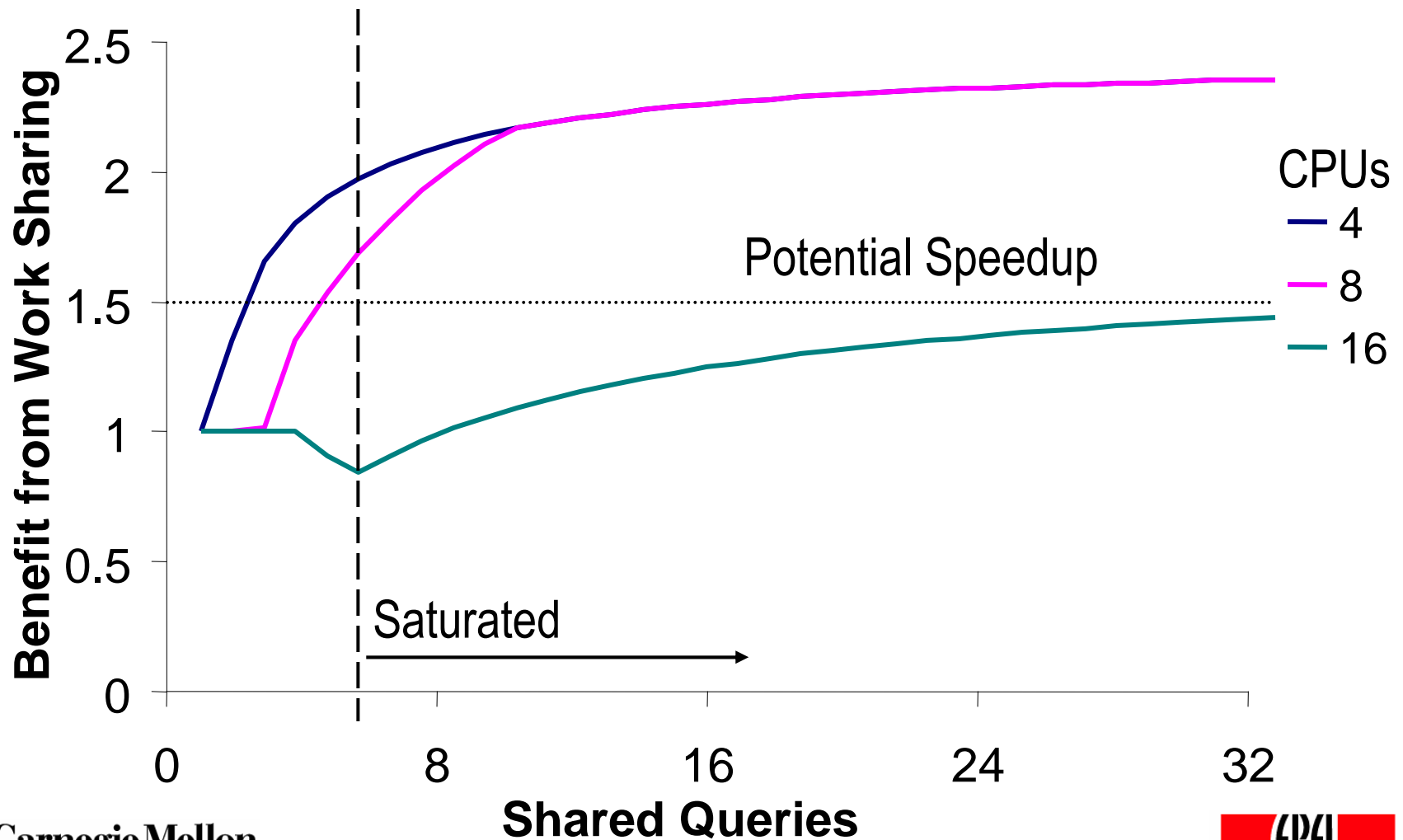


Behavior matches previously published results

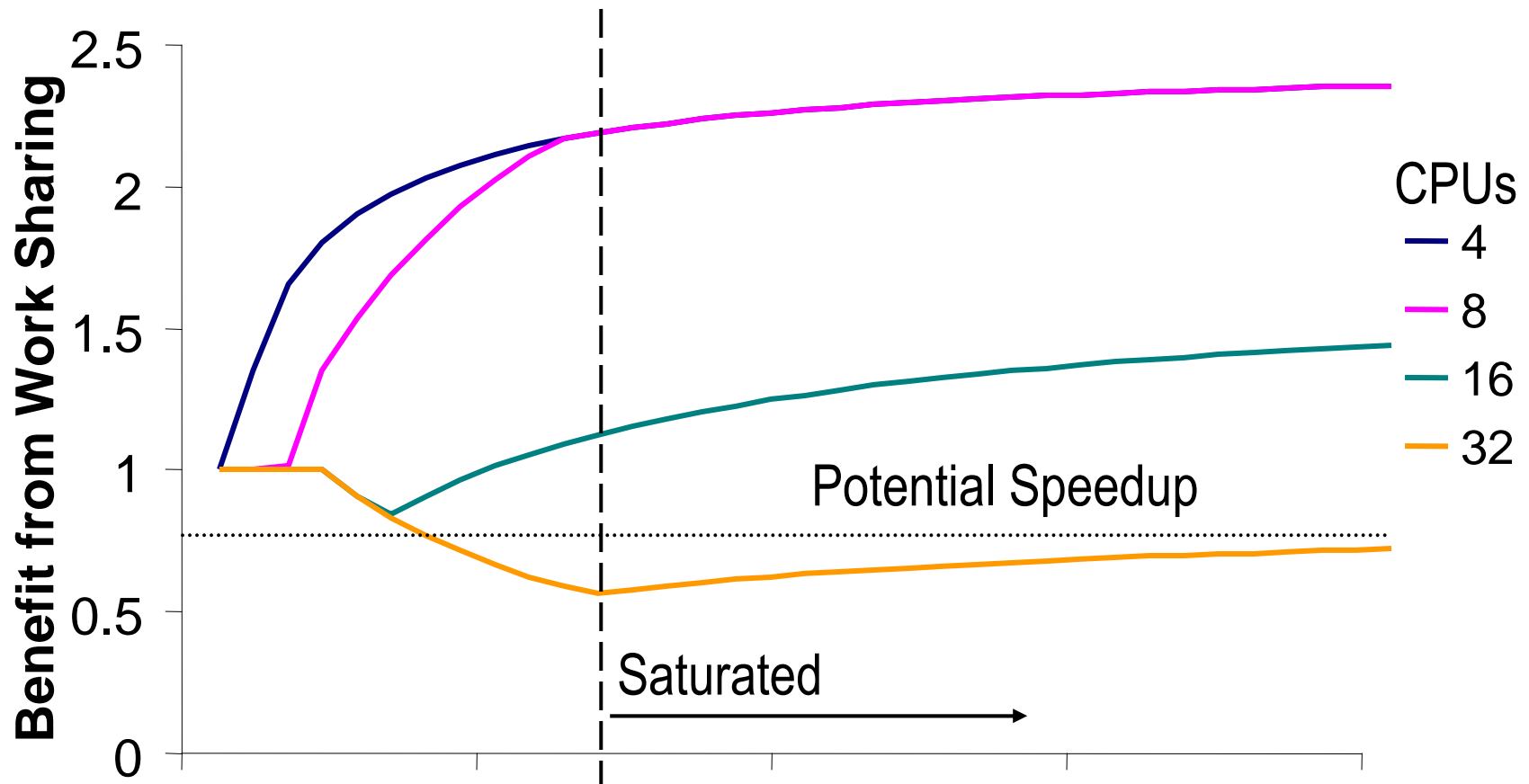
Exploring WS vs. Parallelism



Exploring WS vs. Parallelism



Exploring WS vs. Parallelism



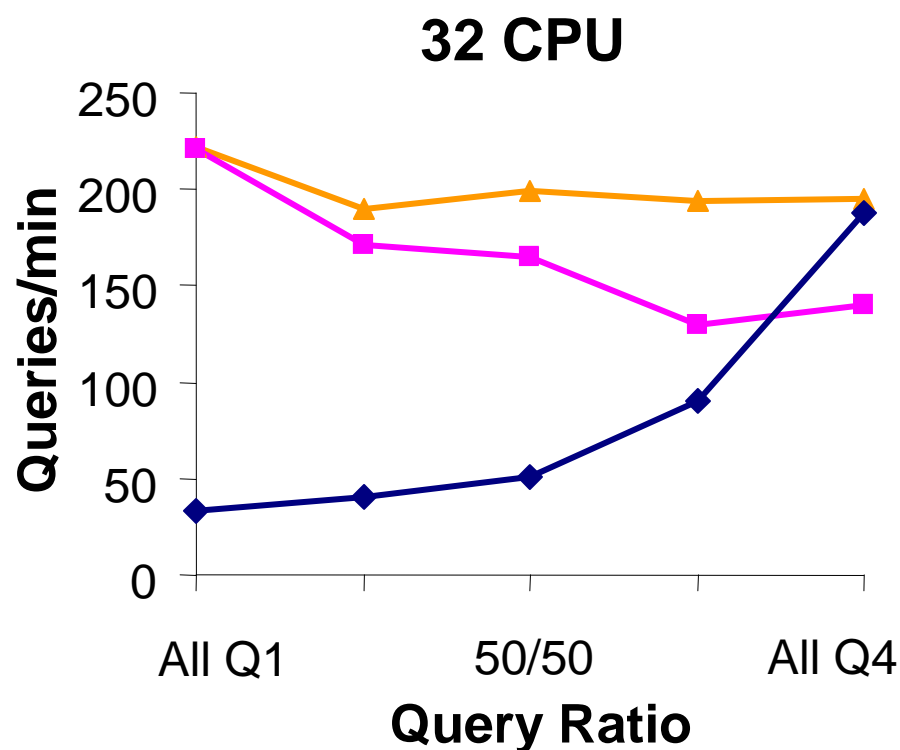
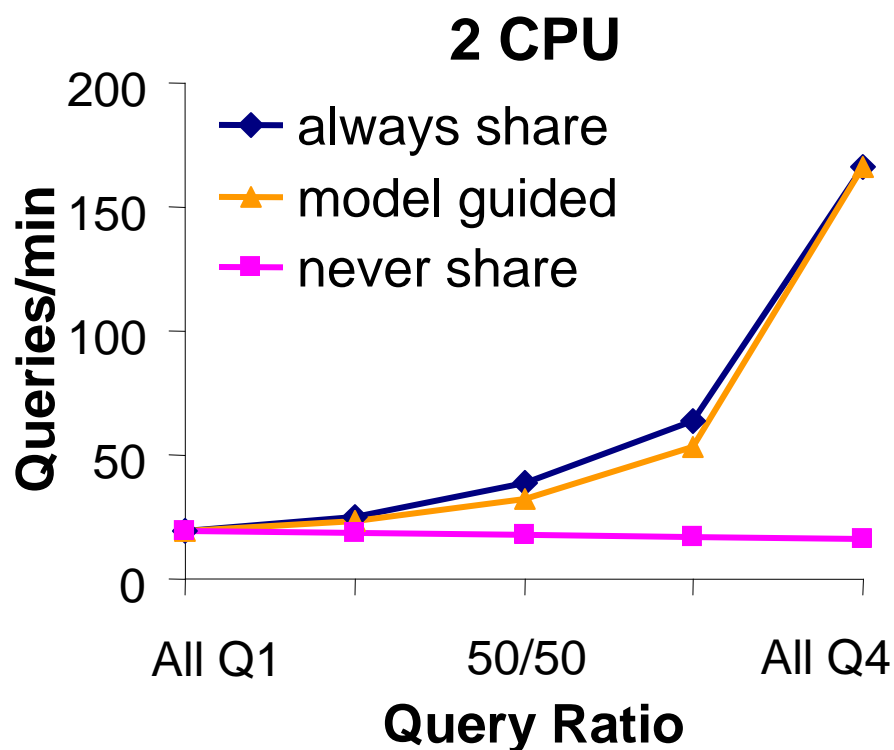
➡ More processors shift bottleneck to critical path

Model-guided Work Sharing

- Integrate predictive model into Cordoba
 - Extract model parameters with profiling tools
 - Predict benefit of work sharing for each new query
- Experiment
 - 20 clients submit back-to-back queries
 - Vary mix of TPCH Q1 and Q4

➡ Compare against always-, never-share policies

Comparison of Work Sharing Strategies



➡ Model-based policy balances critical path and load

Outline

- Introduction
- Understanding Work Sharing
- Analysis and Experiments
- **Work sharing and Working Set Size**

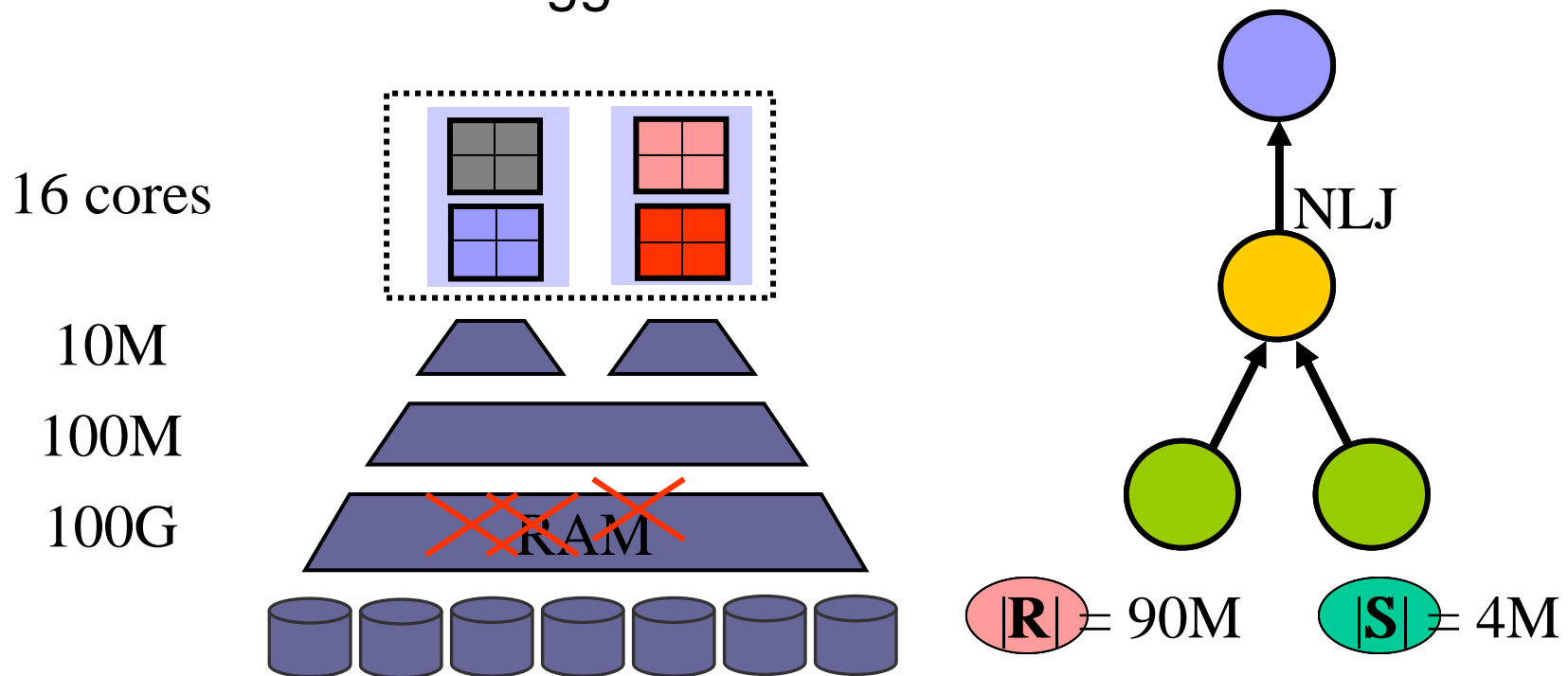
When is work sharing beneficial?

- Benefits of work sharing
 - Disk-bound: avoid extra I/O
 - CPU-bound: eliminate redundant computation
 - *Memory-bound: reduce working set size*
- So far we assume everything “fits” in memory
- Two problems:
 - Enormous datasets
 - Shared memory hierarchy

➡ Challenge: parallelism *and* small working set

Memory-bound Query Processing

- Intermediate results overwhelm mem/cache
 - Limited: capacity within levels, BW between them
 - Multi-core aggravates situation



➡ Work sharing will remain useful for multi-core

Conclusions

- Work sharing can hurt performance
 - Highly parallel, CPU-bound machines
 - Trade-off between total work and critical path
- Model-guided work sharing highly effective
 - Significantly outperforms static policies
- Work sharing can reduce working set sizes
 - Important for shared memory hierarchy

<http://www.cs.cmu.edu/~StagedDB/>