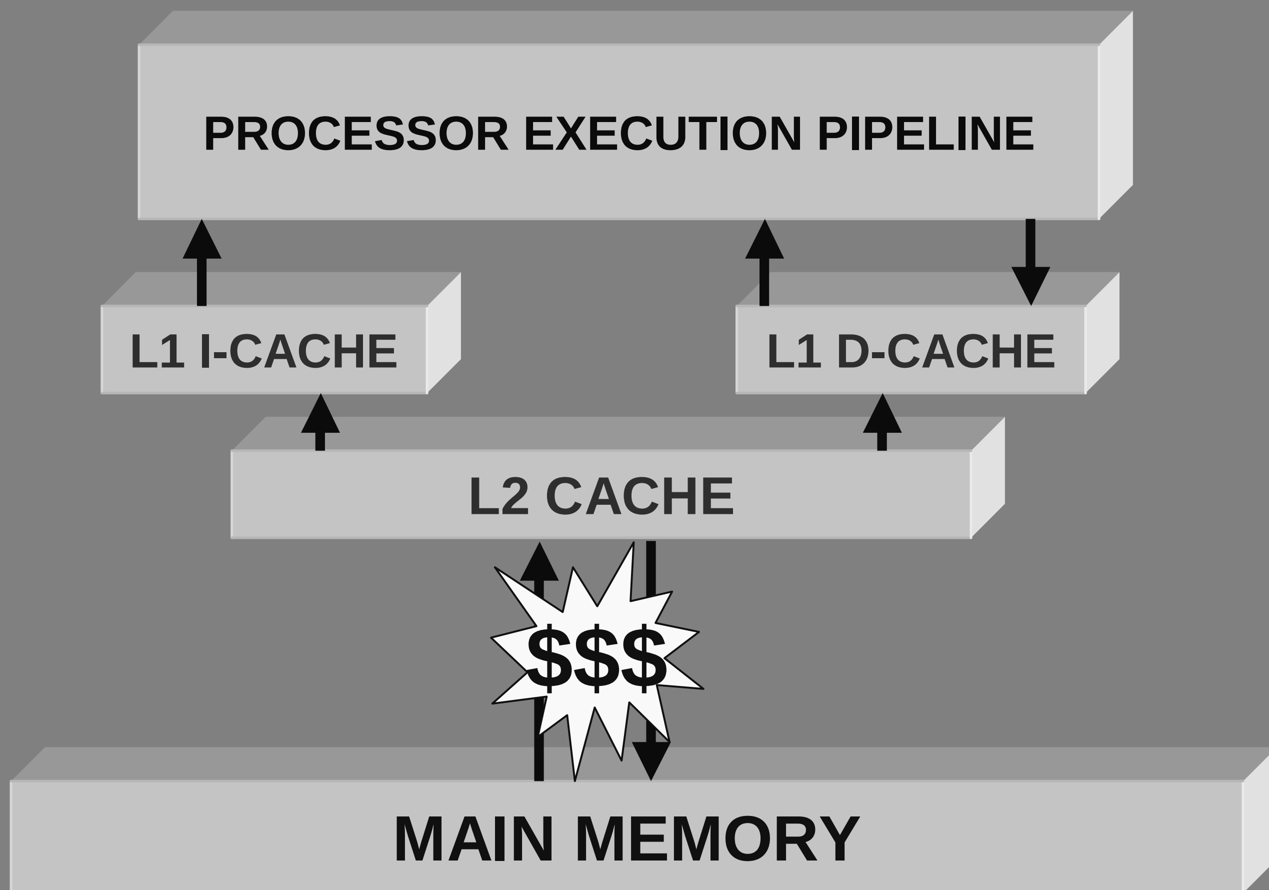
Bridging the Processor/Memory Performance Gap in Database Applications

Anastassia Ailamaki

Carnegie Mellon

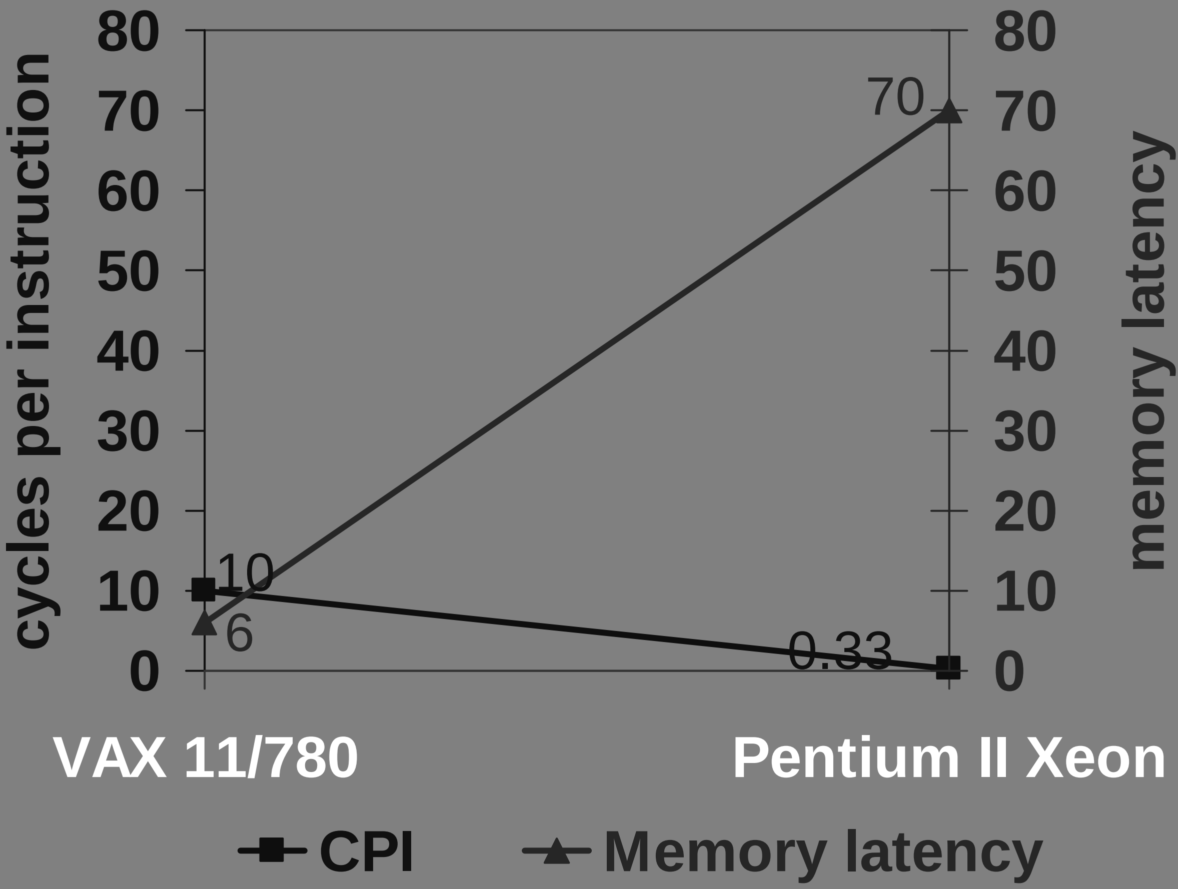
http://www.cs.cmu.edu/~natassa

Memory Hierarchies



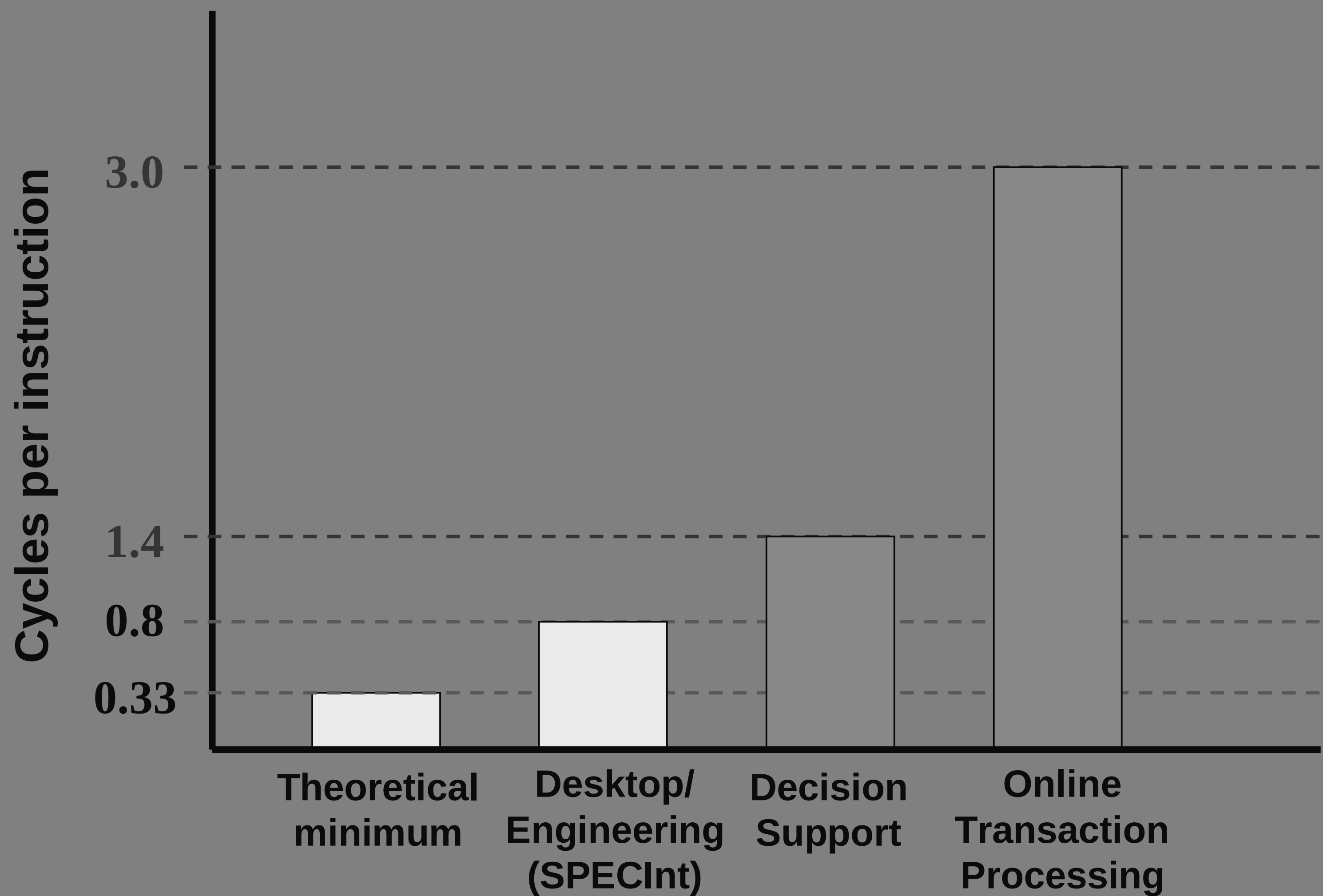
Cache misses are extremely expensive

Processor/Memory Speed Gap



1 memory access $\cong$ 1000 instructions

Who Cares?



We (database people) do.

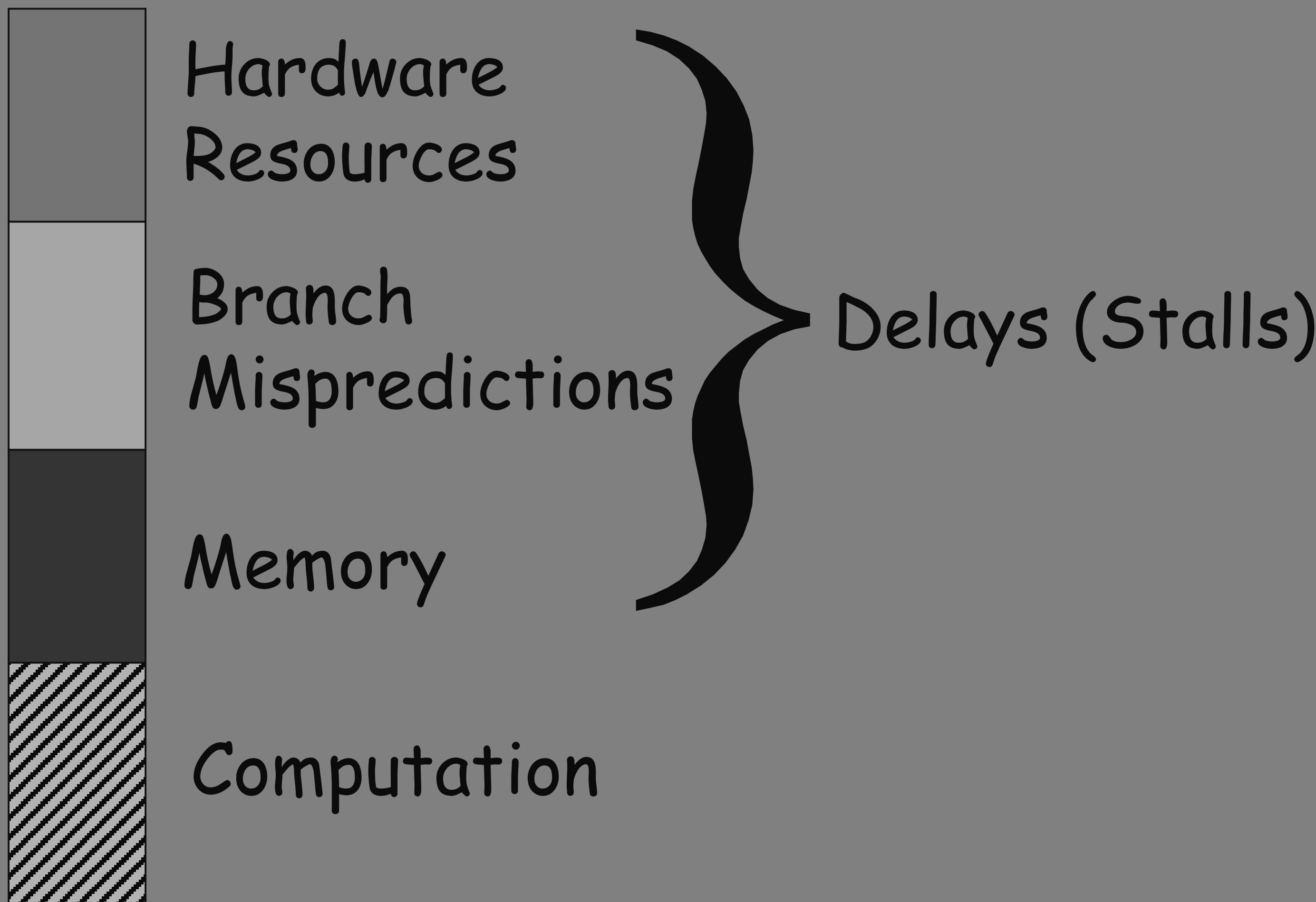
Why DBs? Why Now?

- Memory-intensive, tight instruction streams
- Bottleneck transfer from I/O to memory
  - Larger/slower main memories
  - Smart storage managers/disks hide I/O
- Changing hardware, aging software
- Too many knobs, TPC too complex

Outline

- DBs and the memory/processor speed gap
- **Execution Time Analysis**
  - Query execution time breakdown
  - Bottleneck assessment
- A Bridge over the Memory/Processor Gap

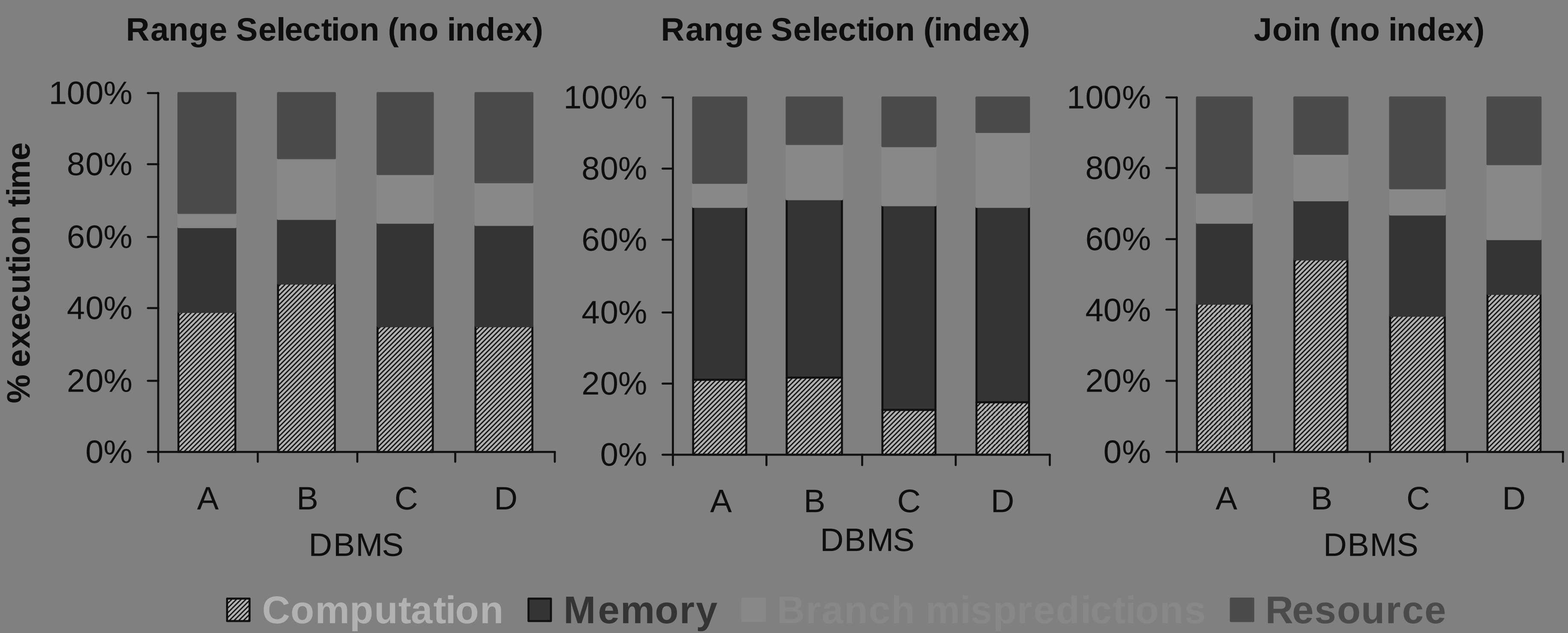
Where Does Time Go?



Execution Time = Computation + Stalls

Breaking Up Execution Time

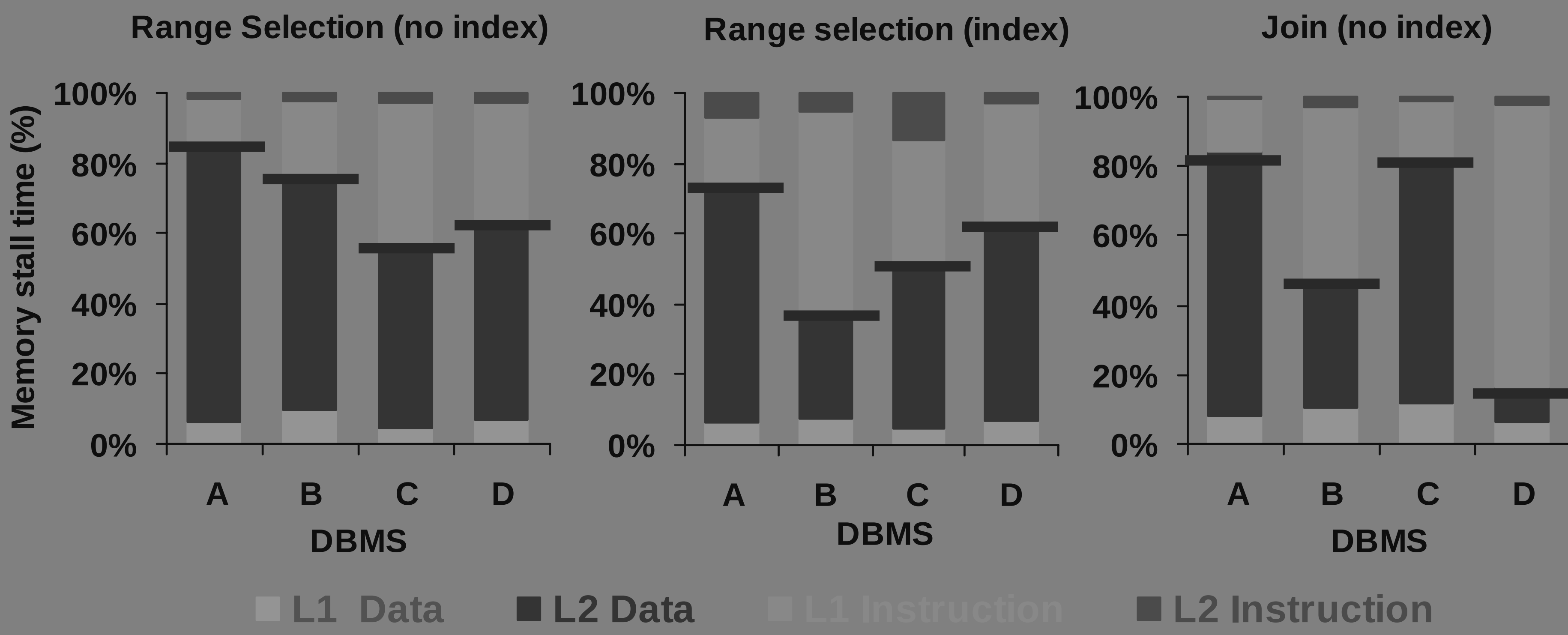
- PII Xeon running NT 4.0, 4 commercial DBMSs: A,B,C,D
- Memory-related delays: 40%-80% of execution time



Memory stalls are major bottleneck

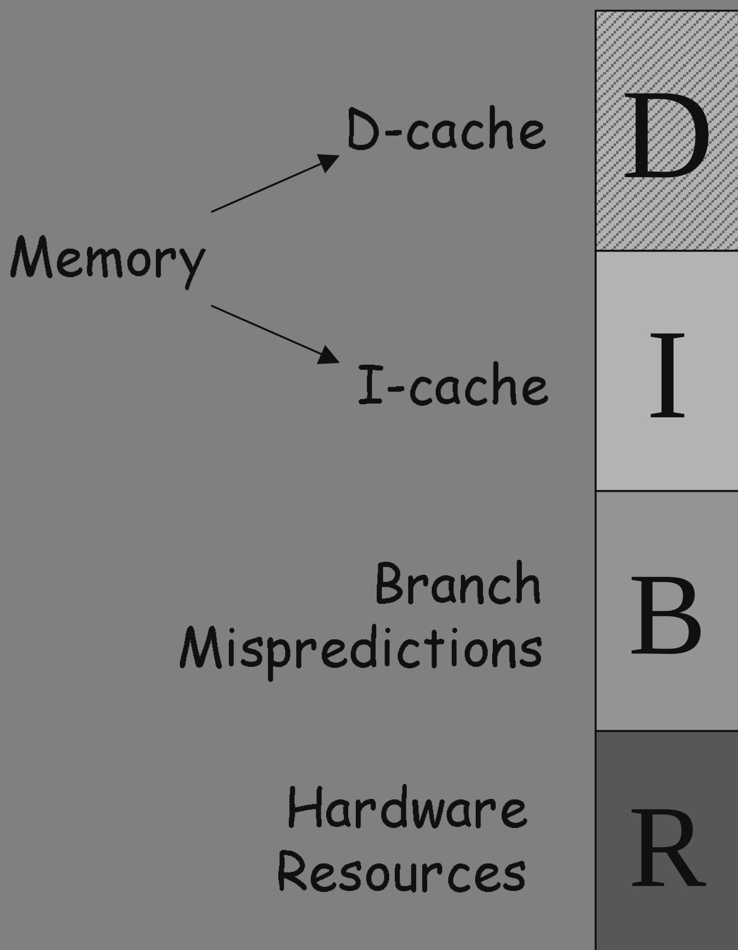
Breaking Up Memory Delays

- PII Xeon running NT 4.0, 4 commercial DBMSs: A,B,C,D
- Memory-related delays: 40%-80% of execution time



Data accesses: 19%-86% of memory stalls

Addressing Bottlenecks



DBMS: <u>improve locality</u>

DBMS + Compiler

Compiler + Hardware

Hardware

Data cache: A clear responsibility of the DBMS

Bridging the Gap

- The "CRDB" performance illusion:
  "My database is cache-resident"
- Make cache misses "disappear"
  - Prevent cache misses
  - Hide penalty from compulsory latencies
- Techniques
  1. Static data placement (my talk today)
  2. Dynamic Data Placement
  3. Aggressive prefetching to hide latencies

Outline

- DBs and the memory/processor speed gap
- Execution time analysis
- **Static Data Placement**
  - What's wrong with slotted pages?
  - Partition Attributes Across (PAX)

Static Data Placement on Disk Pages

- Commercial DBMSs use *Slotted pages*
  - ✓ Store table records sequentially
  - ☺ Intra-record locality (attributes of record r together)
  - ☹ Doesn't work well on today's memory hierarchies

- Alternative: *Vertical partitioning* [Copeland'85]
  - ✓ Store n-attribute table as n single-attribute tables
  - ☺ Inter-record locality, saves unnecessary I/O
  - ☹ Destroys intra-record locality => expensive to reconstruct record

- **New: Partition Attributes Across**
  - ☺ **… have the cake and eat it, too**

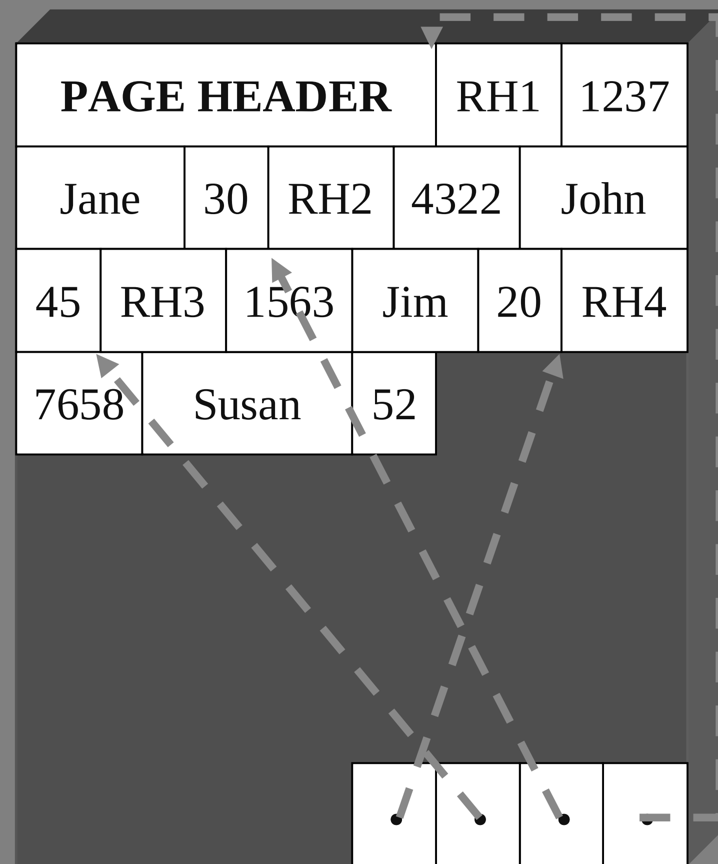
Inter-record locality + low reconstruction cost

Current Scheme: Slotted Pages

Formal name: NSM (N-ary Storage Model)

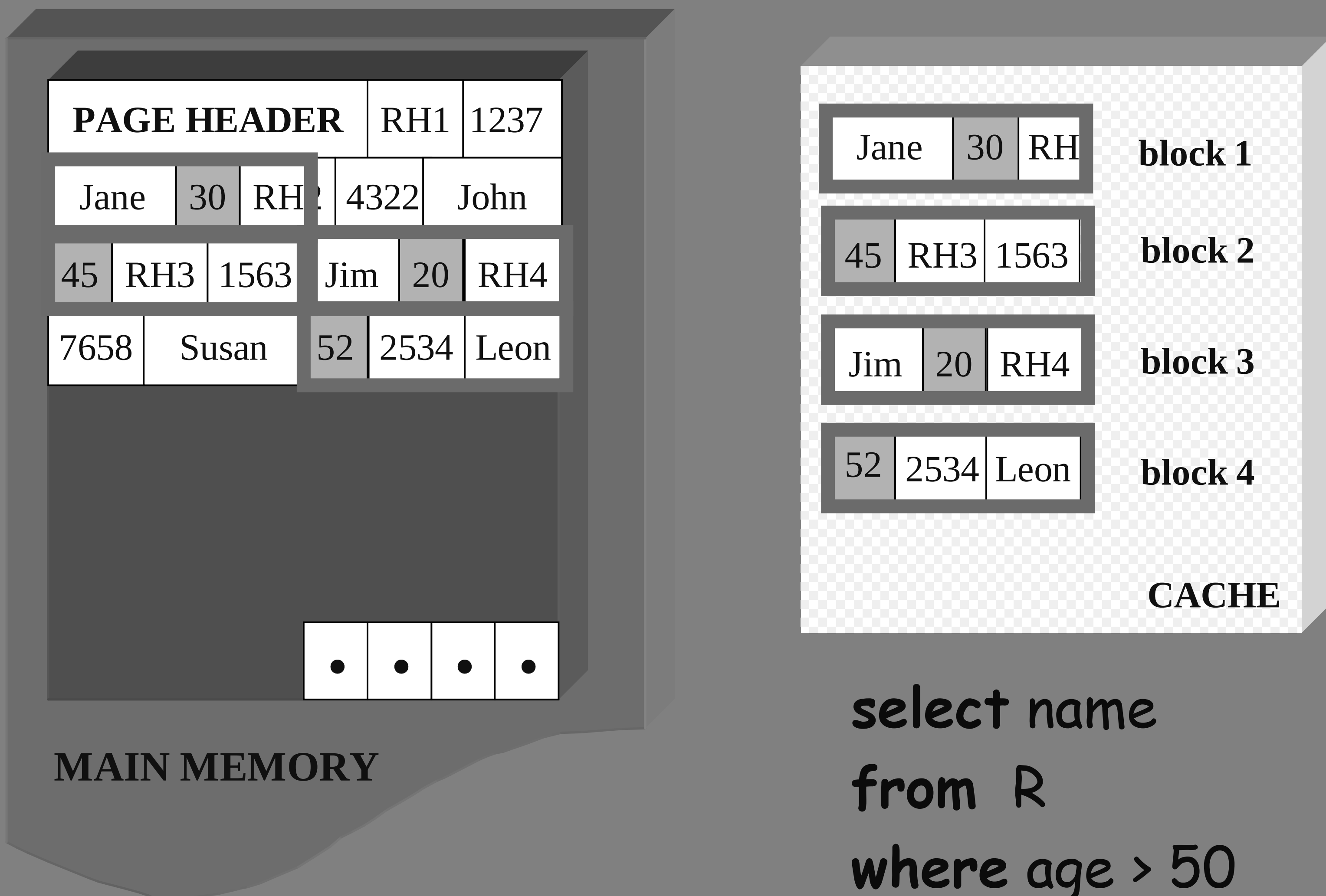
R

RID	SSN	Name	Age
1	1237	Jane	30
2	4322	John	45
3	1563	Jim	20
4	7658	Susan	52
5	2534	Leon	43
6	8791	Dan	37



NSM stores records sequentially w/ offsets

Predicate Evaluation using NSM



select name
from R
where age > 50

NSM pushes non-referenced data to the cache

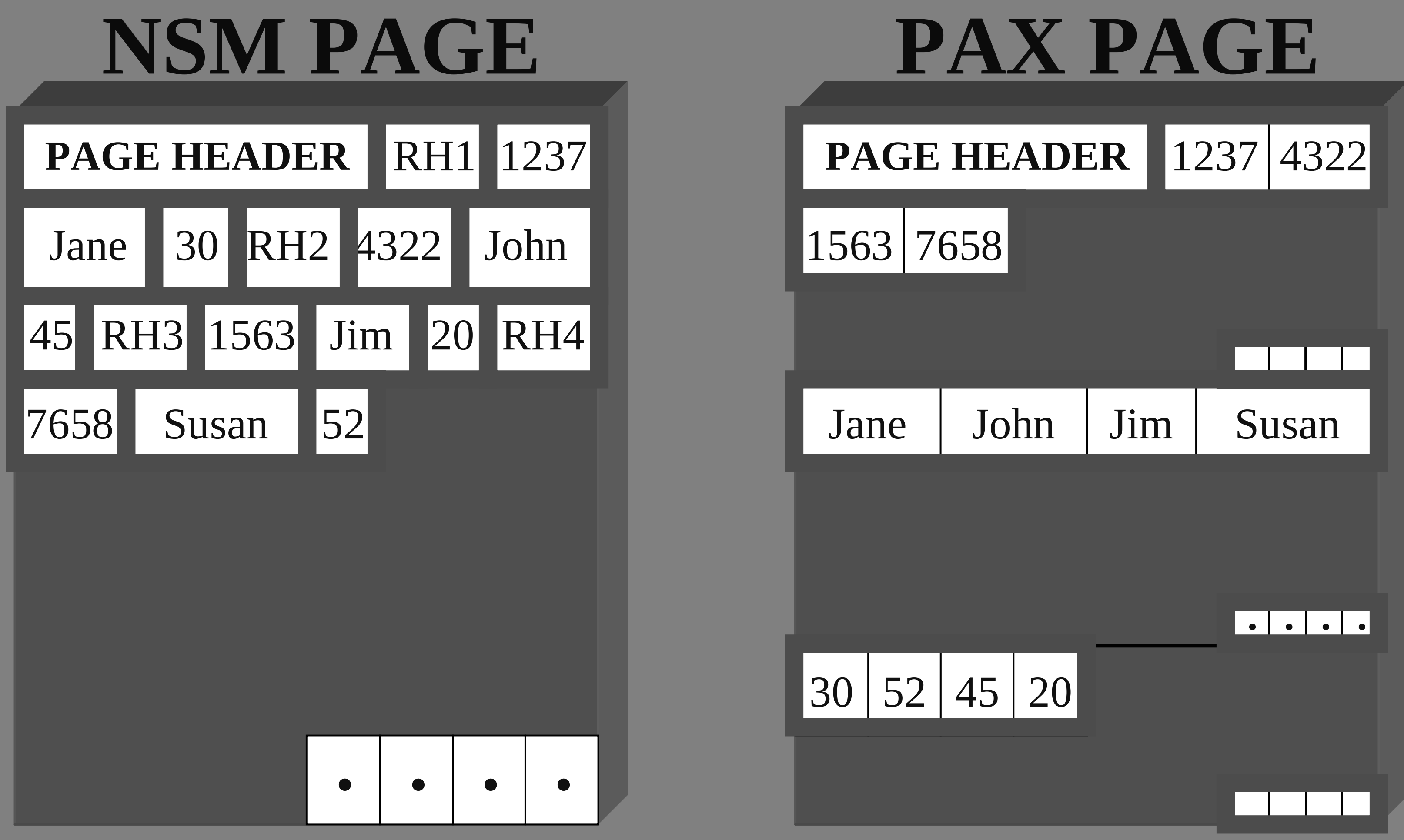
Need New Data Page Layout

- Eliminates unnecessary memory accesses
- Improves inter-record locality
- Keeps a record's fields together
- Does not affect I/O performance

and, most importantly, is…

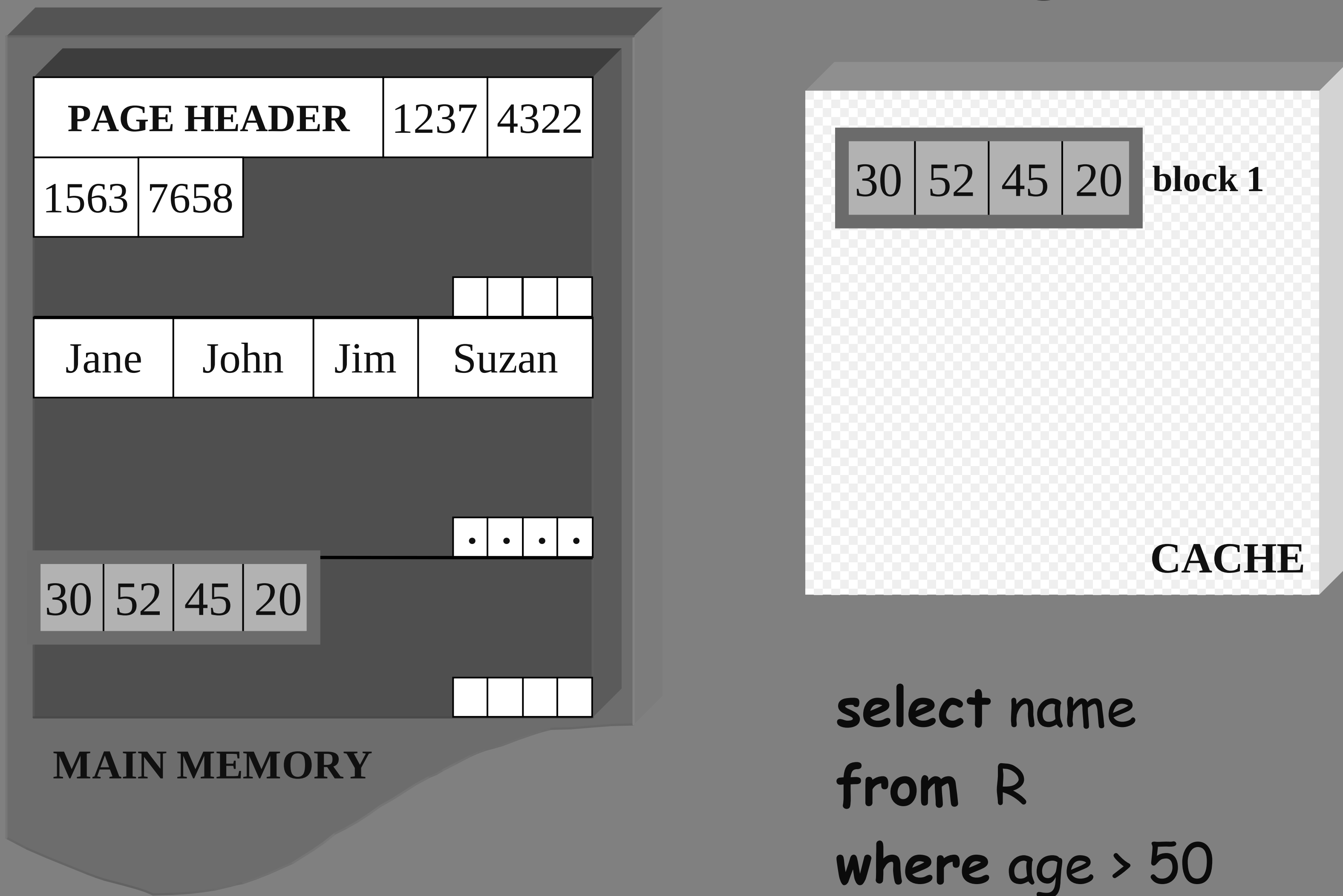
low-implementation-cost, high-impact

Partition Attributes Across (PAX)



Partition data *within* the page for spatial locality

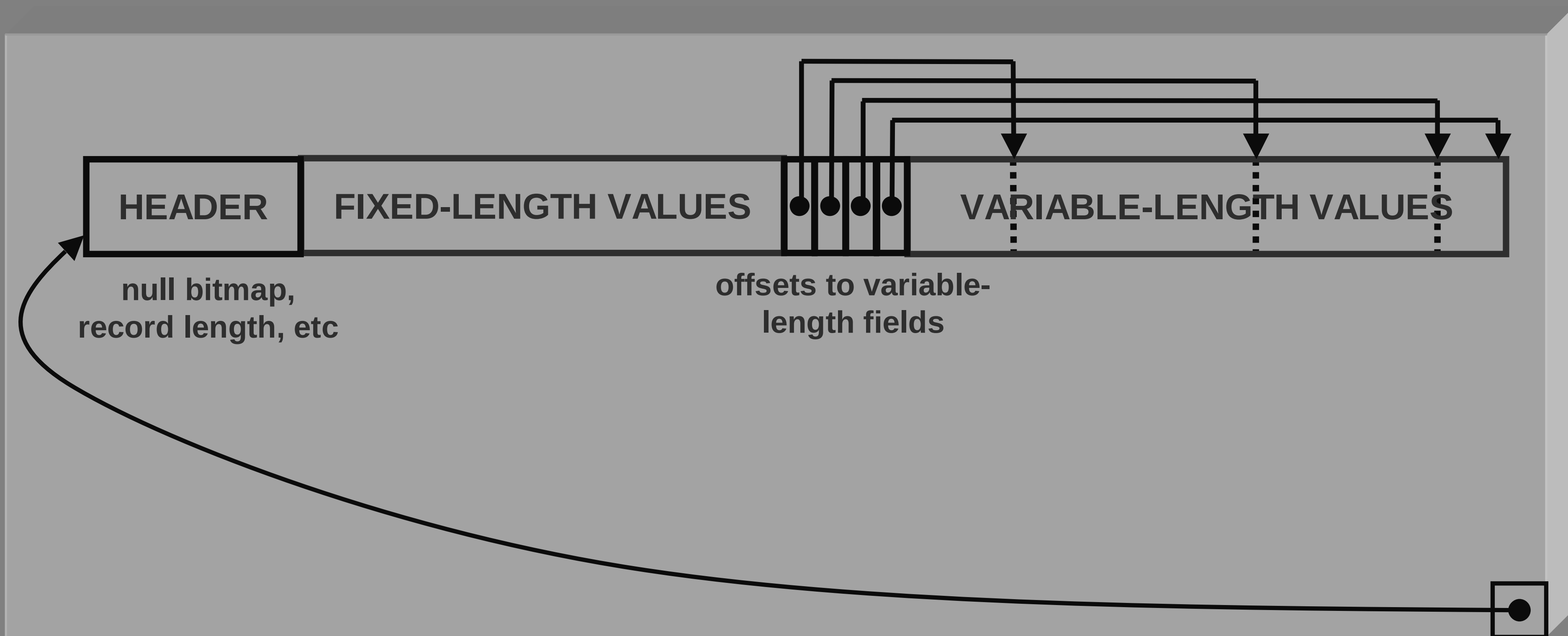
Predicate Evaluation using PAX




```
select name
from R
where age > 50
```

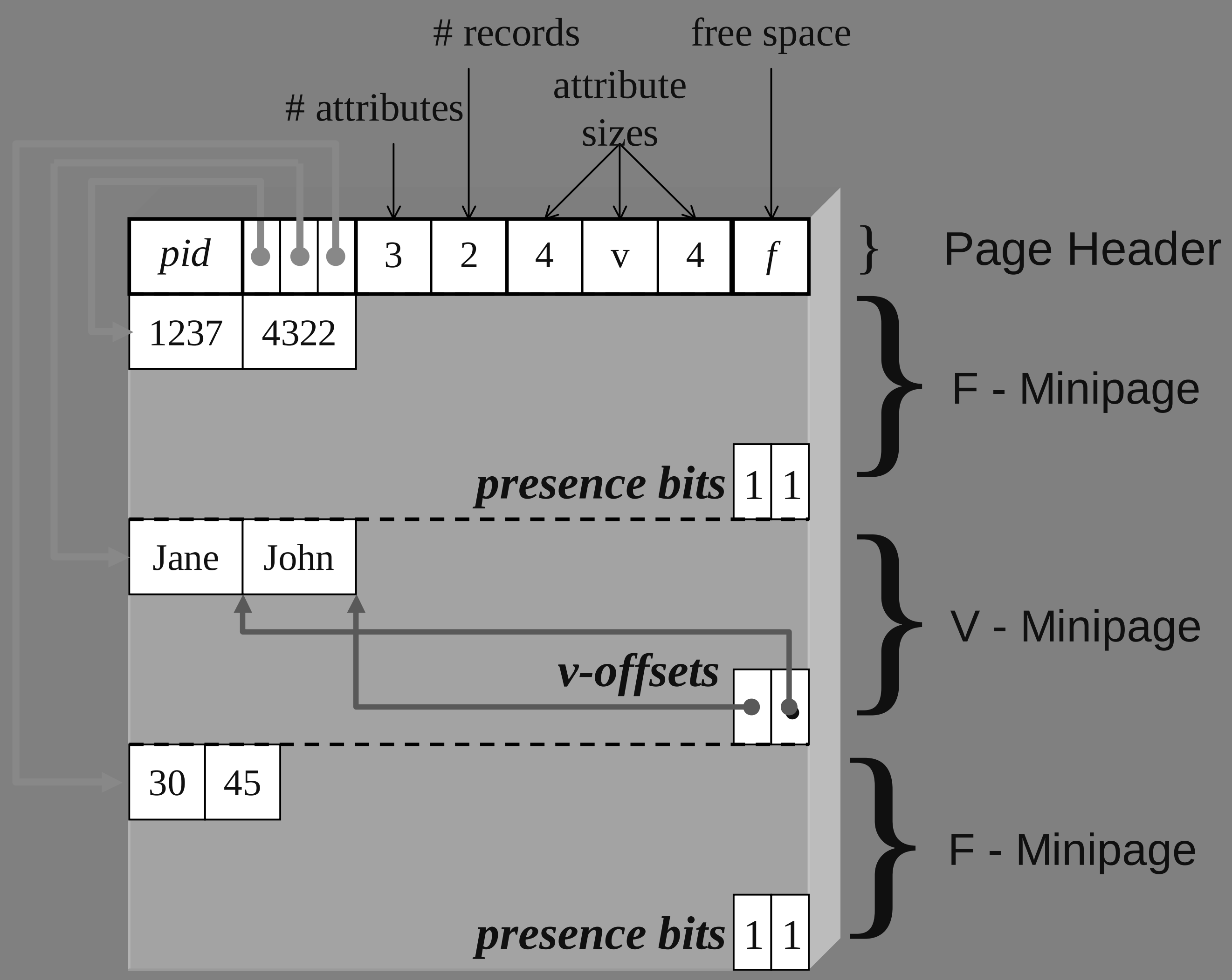
Fewer cache misses, low reconstruction cost

A Real NSM Record



NSM: All fields of record stored together + slots

PAX: Detailed Design



PAX: Group fields + amortizes record headers

Sanity Check: Basic Evaluation

- Main-memory resident R, numeric fields
- Query:

  select avg (a_j)

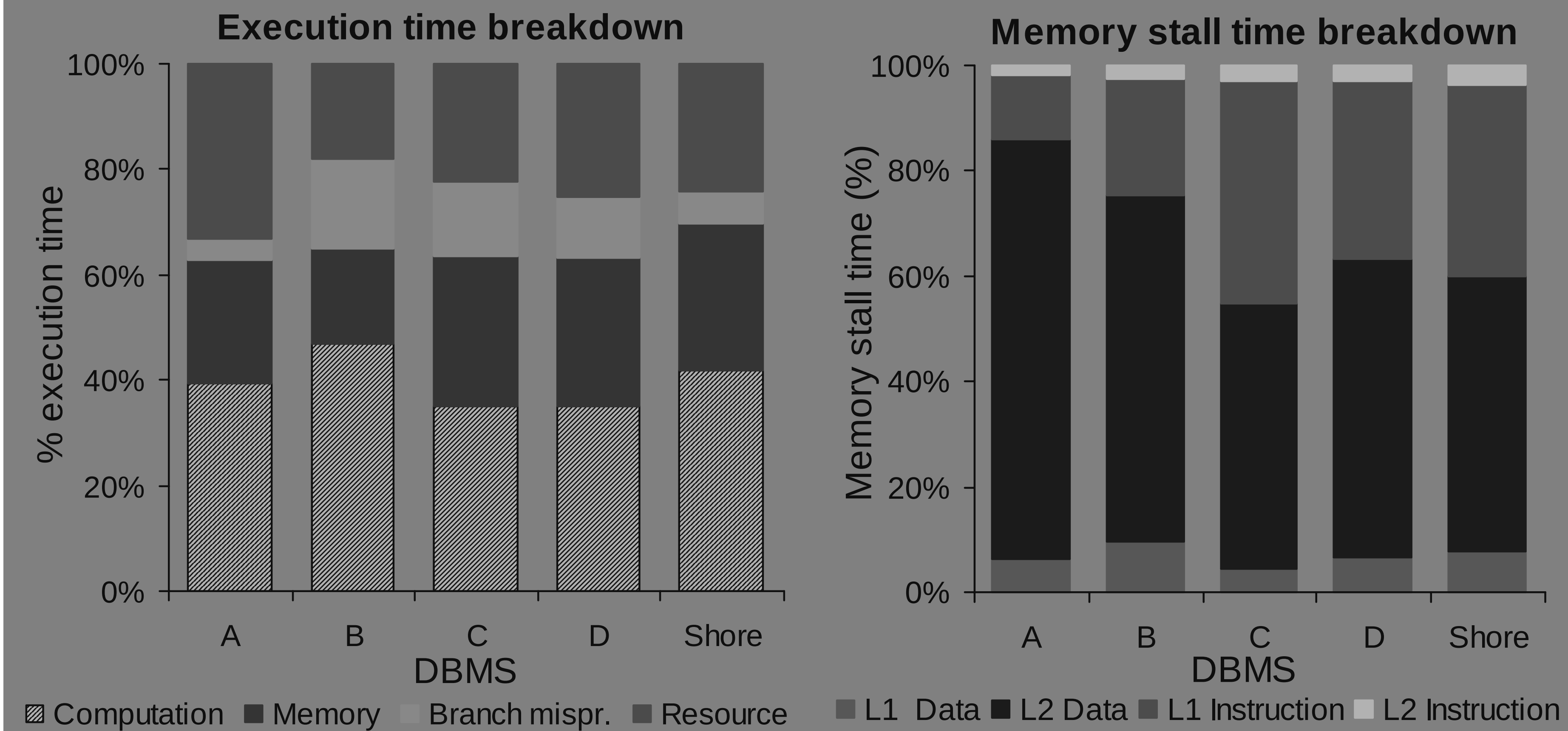
  from R

  where a_j >= Lo **and** a_j <= Hi
- PII Xeon running Windows NT 4
- 16KB L1-I, 16KB L1-D, 512 KB L2, 512 MB RAM
- Used processor counters
- Implemented schemes on Shore Storage Manager
  - Similar behavior to commercial Database Systems

Carnegie Mellon

Why Use Shore?

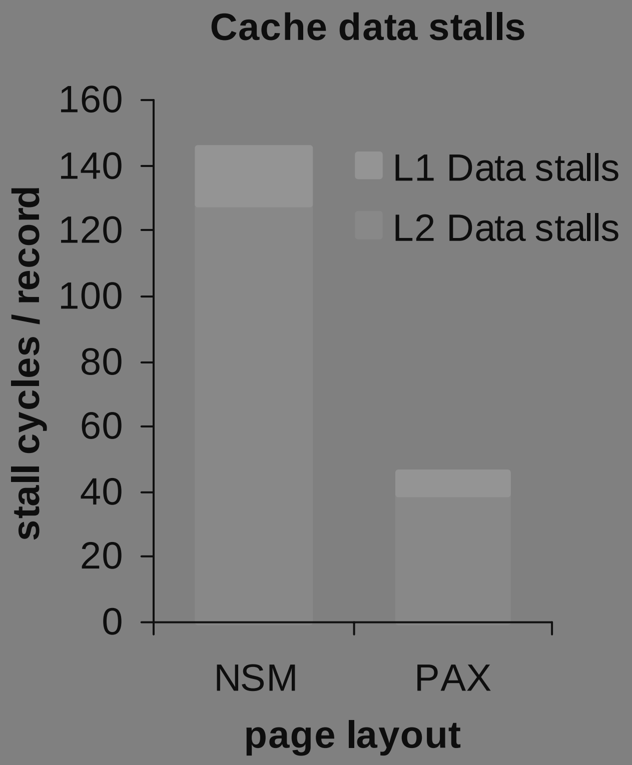
- Compare Shore query behavior with commercial DBMS
- Execution time & memory delays (range selection)

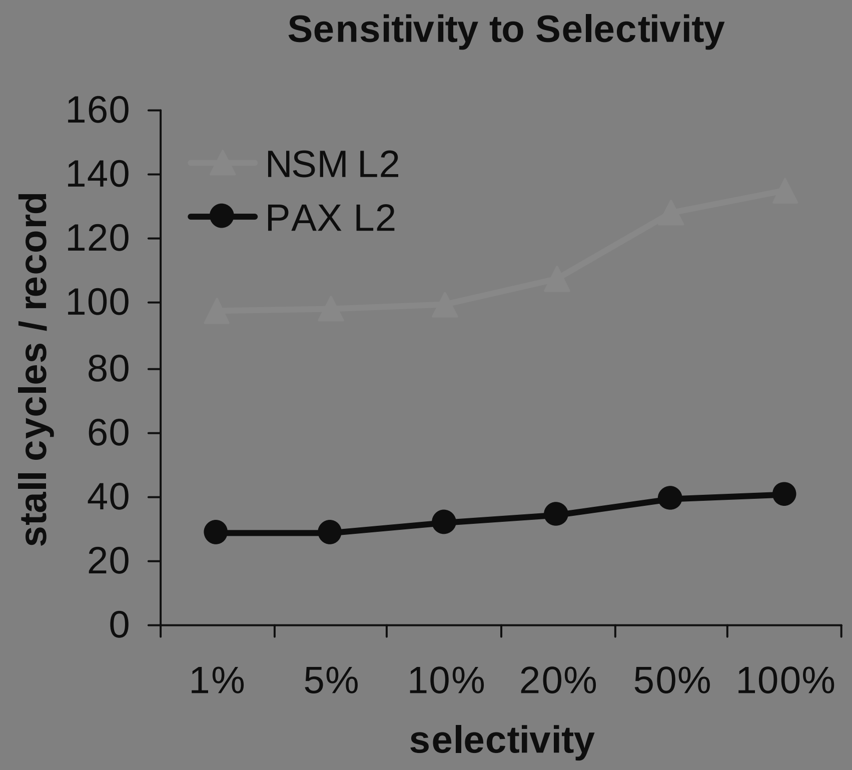


We can use Shore to evaluate workload behavior

Effect on Accessing Cache Data

- PAX saves 70% of data penalty (L1+L2)
- Selectivity doesn't matter for PAX data stalls

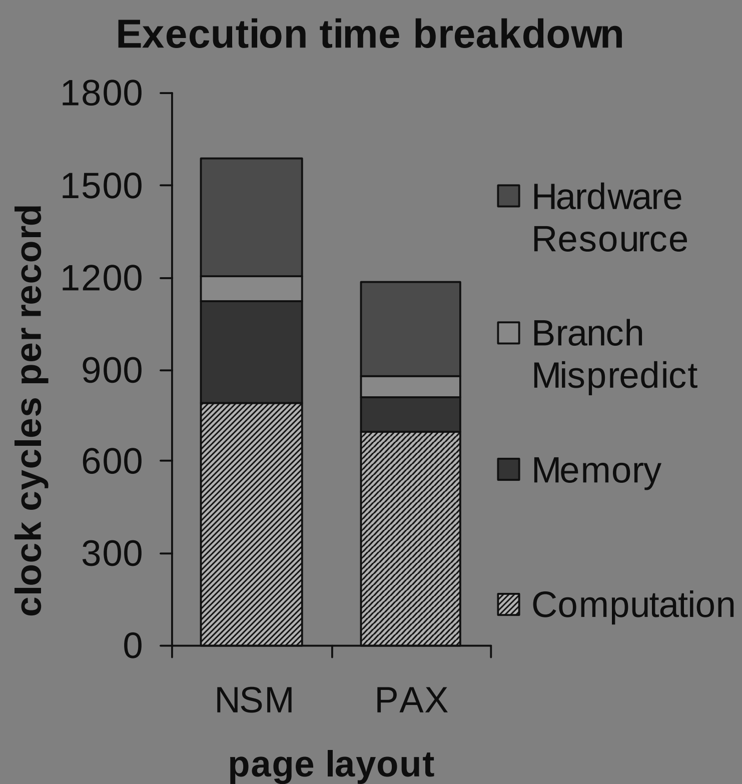


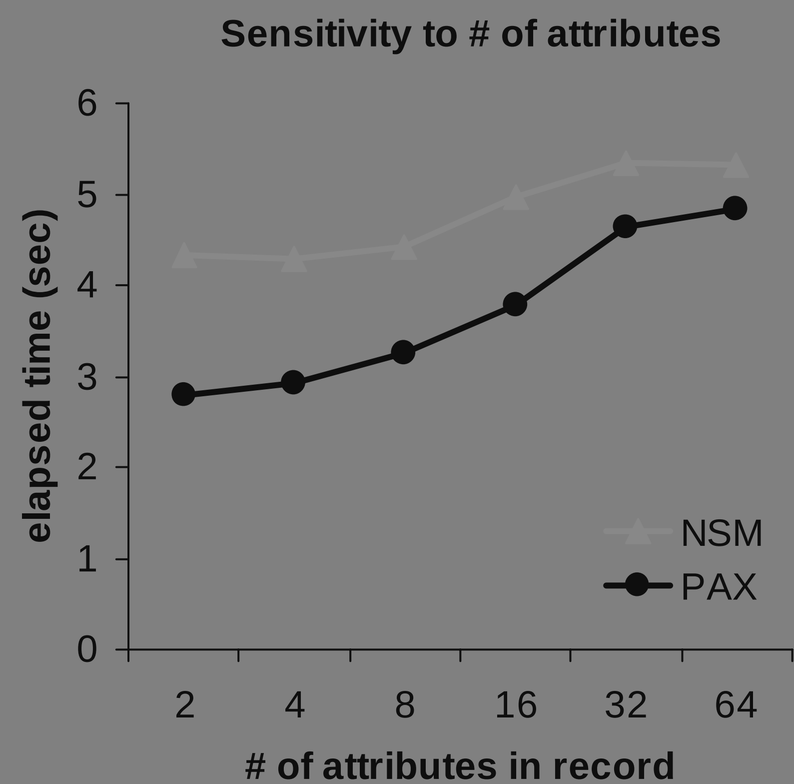


PAX drastically reduces data stalls

Time and Sensitivity Analysis

- PAX: 75% less memory penalty than NSM (10% of time)
- Execution times converge as number of attrs increases

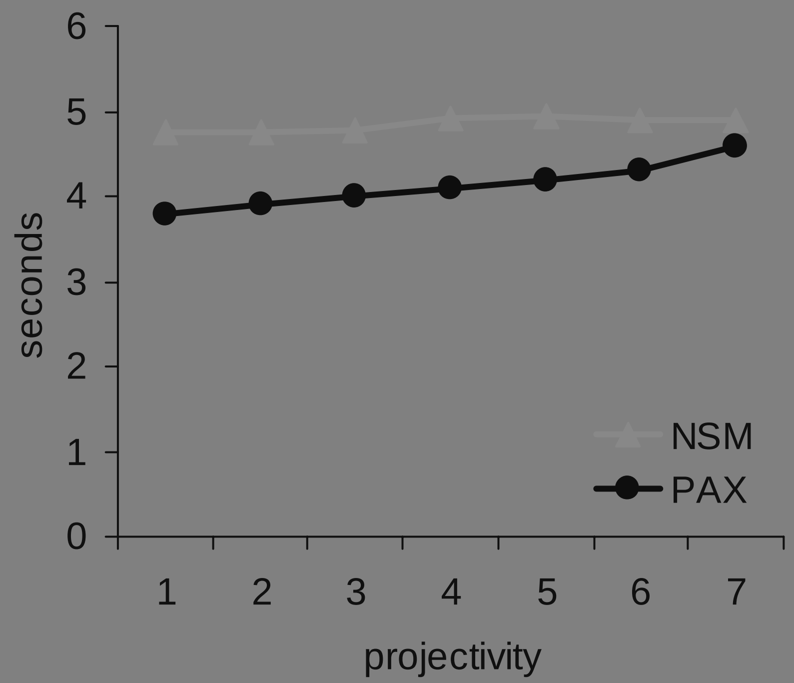


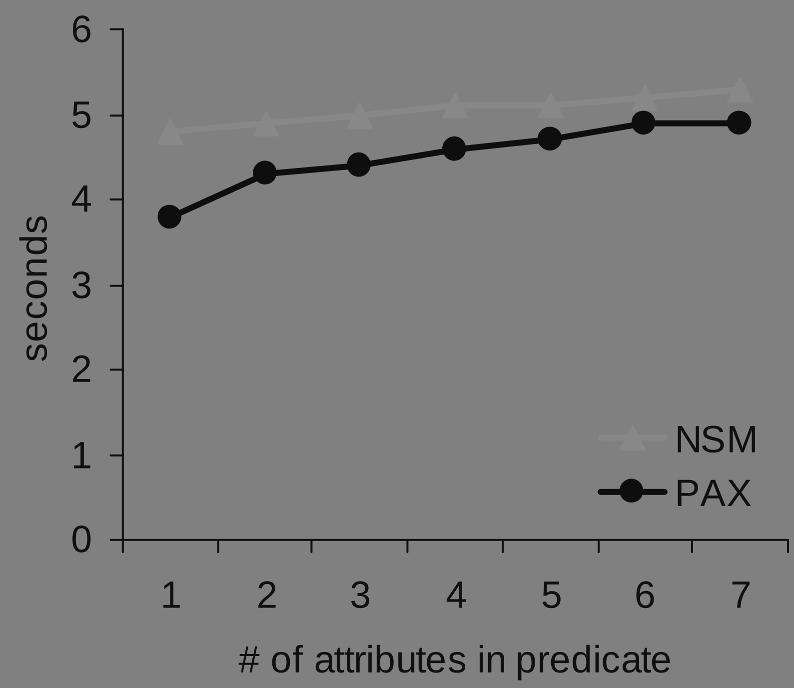


PAX improves overall execution time

Sensitivity Analysis (2)

- Elapsed time sensitivity to projectivity / # predicates
- Range selection queries, 1% selectivity





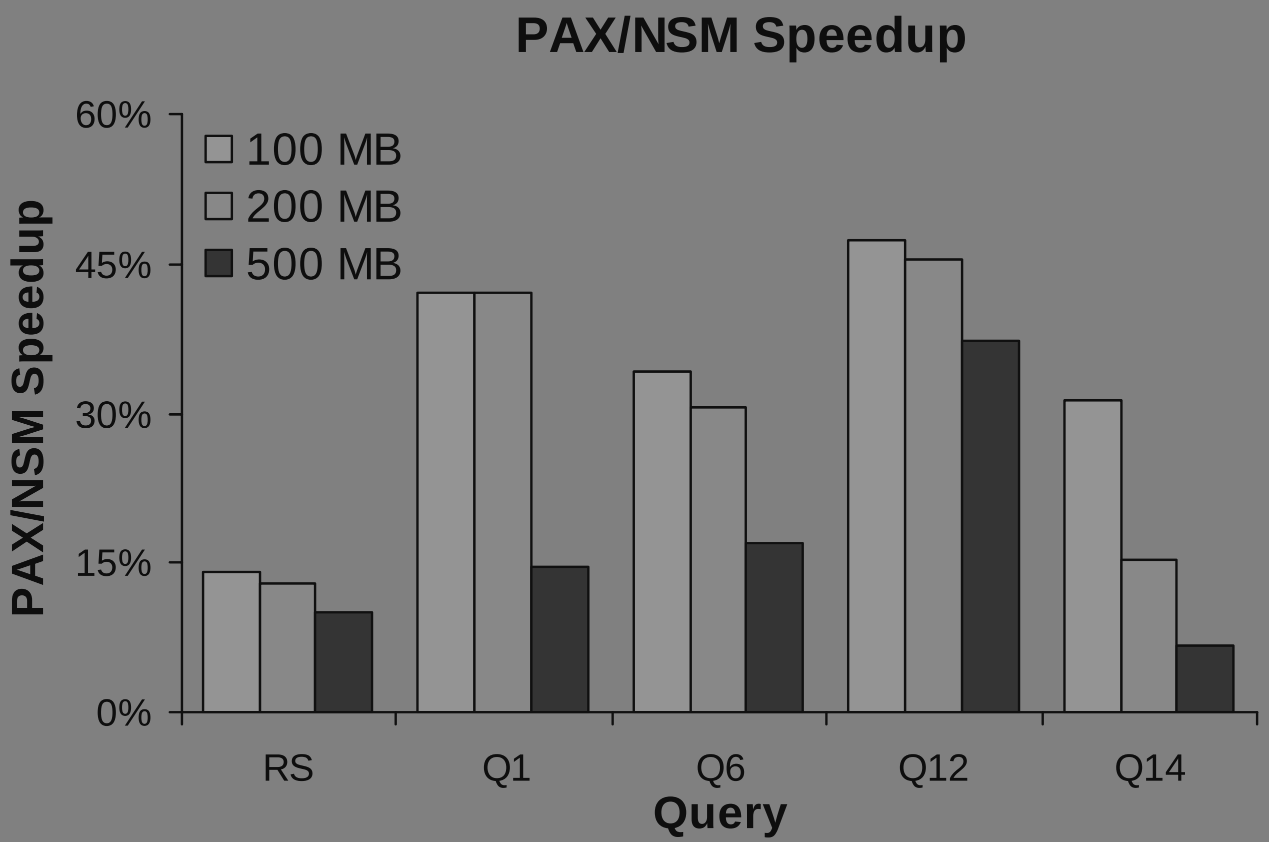
PAX,NSM times converge as query covers entire tuple

Evaluation Using a DSS Benchmark

- 100M, 200M, and 500M TPC-H DBs
- Queries:
  1. Range Selections w/ variable parameters (RS)
  2. TPC-H Q1 and Q6
     - sequential scans
     - lots of aggregates (*sum*, *avg*, *count*)
     - grouping/ordering of results
  3. TPC-H Q12 and Q14
     - (Adaptive Hybrid) Hash Join
     - complex 'where' clause, conditional aggregates
- 128MB buffer pool

TPC-H Queries: Speedup

- Avg(range selections) + 4 TPC-H queries
- Shore on PII/NT

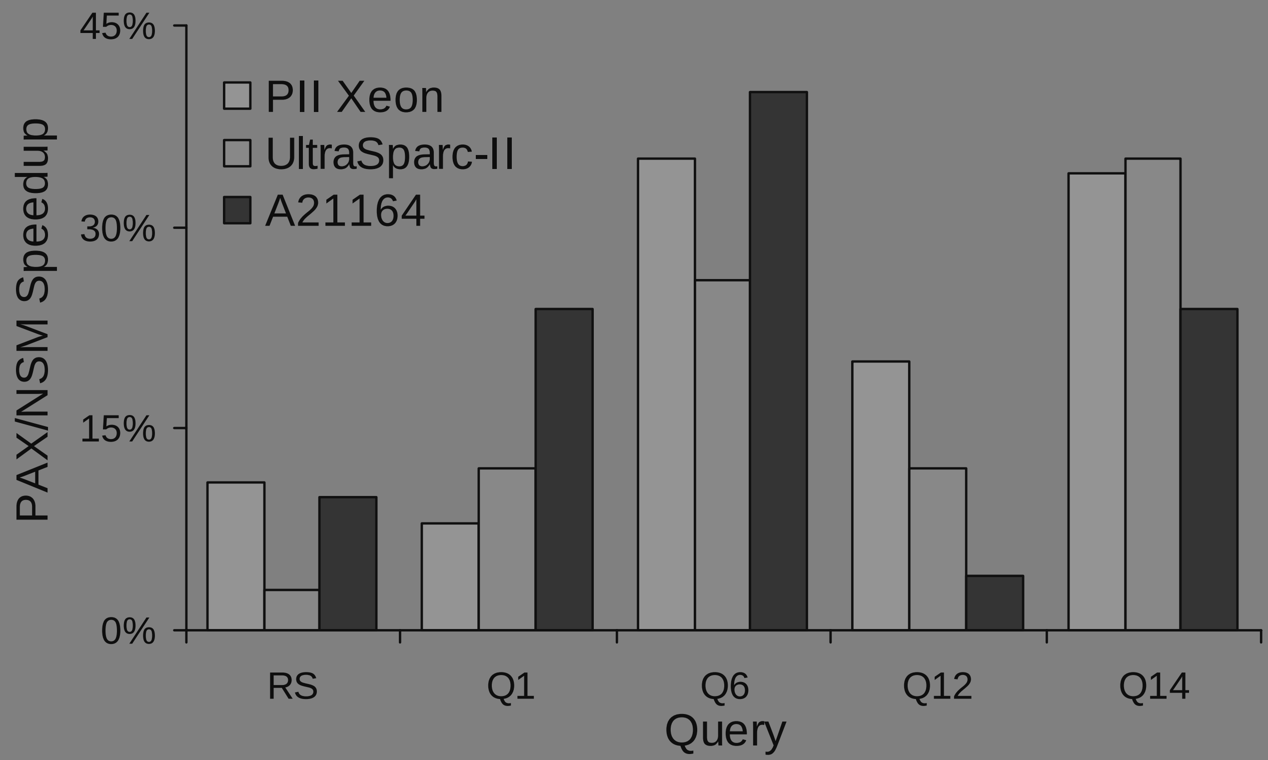


PAX: 50% elapsed time improvement in TPC-H

PAX vs. NSM across platforms

- Avg(range selections) + 4 TPC-H queries
- Shore on PII/Linux, UltraSparc-II/Solaris, A21164/Tru64

PAX/NSM Speedup on Unix (100MB database)

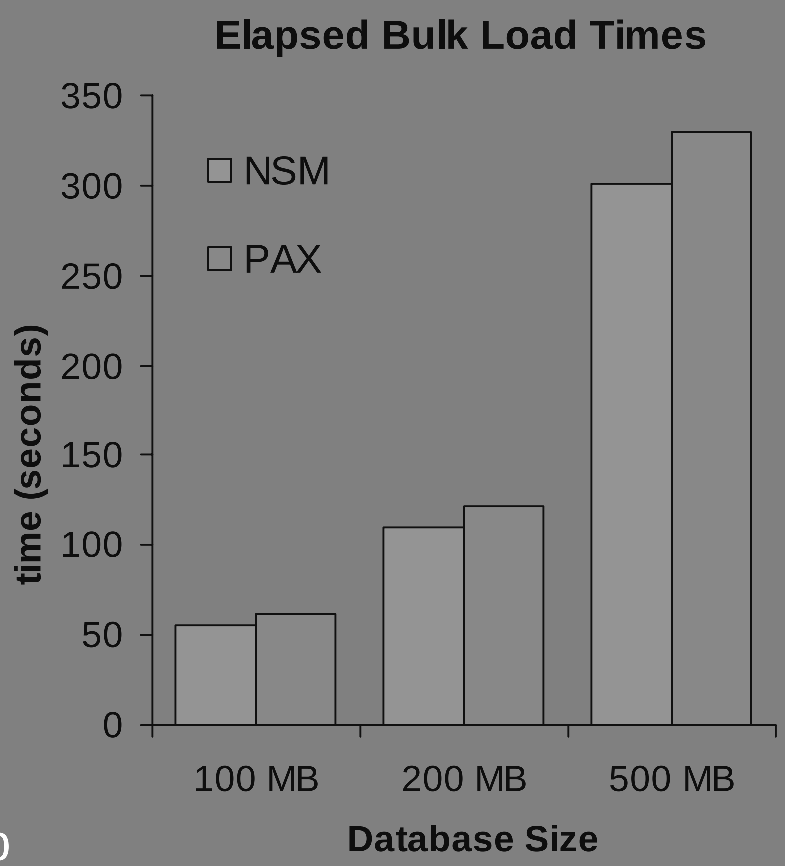


PAX improves performance across platforms

Insertions

- Estimate average field sizes
- Start inserting records
- If a record doesn't fit,
  - Reorganize page
  - (move minipage boundaries)
- Adjust average field sizes

- 50% of reorganizations to accommodate a single record
- Threshold 10%: penalty =0.8%



Max bulk load penalty: 2-10% for a TPC-H DB

Updates

- Policy: Update in-place
- Variable-length: Shift when needed
- PAX only needs shift minipage data

- Update statement:

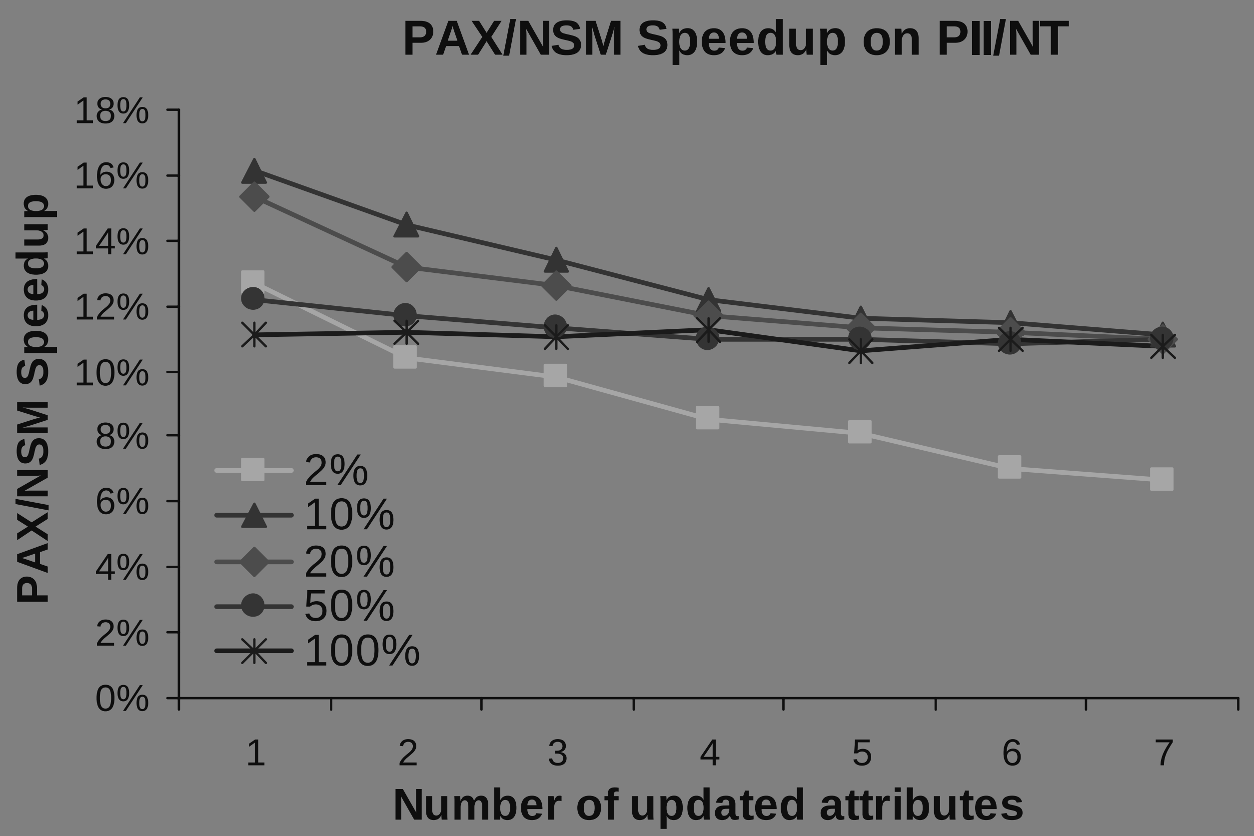
update R

set $a_p = a_p + b$

where $a_q > Lo$ and $a_q < Hi$

Updates: Speedup

- Lower selectivity => reads dominate speedup
- High selectivity => write-backs dominate speedup



PAX always speeds updates up as well (7-17%)

PAX Summary

- PAX: a *low-cost, high-impact* DP technique

- Performance
  - Eliminates unnecessary memory references
  - High utilization of cache space/bandwidth
  - Faster than NSM (does not affect I/O)

- Usability
  - Orthogonal to other storage decisions
  - “Easy” to implement in large existing DBMSs

Conclusions

- It's the memory…
- Need techniques to
  - Drastically improve performance on today's platforms
  - Prepare for future deeper memory hierarchies
- Data placement (static and dynamic)
- Fully exploit space/bandwidth in cache hierarchy
- Collaboration and feedback to the architects

Carnegie Mellon

References

- A. Ailamaki, D.J. DeWitt, M.D. Hill, and D.A. Wood. **DBMSs on a Modern Processor: Where Does Time Go?**, *VLDB* 1999.

- A. Ailamaki, D.J. DeWitt, M.D. Hill, and M. Skounakis. **Weaving Relations for Cache Performance**, *VLDB* 2001.

Carnegie Mellon