Weaving Relations for Cache Performance

Anastassia Ailamaki
Carnegie Mellon

Computer Platforms in 1980

- DBMS Execution: 10 cycles/instruction
- DBMS Data and Instructions: 6 cycles
- Main Memory: 1 Megabyte (Buffer pool)
- I/O: < 1 MBps
- Hot data

The main performance bottleneck was I/O latency

Present Platforms

- DBMS Execution: Processor -> Cache -> Main Memory -> Database
- DBMS Data and Instructions: 25 MBps
- Main Memory: 1 Gigabyte
- Database: DB DB DB

Hot data migrates to larger and slower main memory
Future Platforms

DBMS Execution

Processor/CACHE

DBMS Data and Instructions

Main Memory

1 Terabyte

DB

75 MBps

Hot data migrates to larger and slower main memory

Processor/Memory Speed Gap

Why has the Bottleneck Shifted?

- Lots of I/O optimizations
  - hardware (DMA)
  - software (storage managers do sequential I/O)
- Disks are powerful
  - have cache memories and processors
  - disks are replacing tapes as backup devices
- Databases are largely memory-resident
  - 'hot' data are in big but slow main memory
- Compute- and memory-bound applications
  - e.g., spatial joins

New bottleneck: memory+computation resources
Why Study DB Workloads?

High average time per instruction for DB workloads

Outline

- The memory/processor speed gap
- Importance of Data Placement
  - What’s wrong with slotted pages?
  - Partition Attributes Across (PAX)
- Performance results
- Summary

An Execution Pipeline

Branch prediction, non-blocking cache, out-of-order
Memory hierarchy 101

- Memory hierarchy = main memory + caches
- What are caches?
- Why are they important?
- Caches in today's machines
  - PIII
  - A21164
  - Itanium
  - Power4
- Important cache design parameters?

Where Does Time Go?

Hardware Resources \{ Delays (Stalls) \}
Branch Mispredictions
Memory
Computation

Overlap opportunity:
- Load A
- D = B + C
- Load E

Execution Time = Computation + Stalls - Overlap

Execution Time Breakdown (%)

- PII Xeon running NT 4.0, 4 commercial DBMSs: A,B,C,D
- Stalls at least 50% of the time

Memory stalls are major bottleneck
Breakdown of 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

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

- Contribution: Partition Attributes Across
  - ... have the cake and eat it, too
  - Inter-record locality + low reconstruction cost

Outline

- The memory/processor speed gap
- Importance of Data Placement
- What’s wrong with slotted pages?
- Partition Attributes Across (PAX)
- Performance results
- Summary
Formal name: NSM (N-ary Storage Model)

Current Scheme: Slotted Pages

Predicate Evaluation using NSM

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...
Outline

- The memory/processor speed gap
- Importance of Data Placement
- What’s wrong with slotted pages?
- Partition Attributes Across (PAX)
- Performance results
- Summary

Partition Attributes Across (PAX)

Partition data within the page for spatial locality

Predicate Evaluation using PAX

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

Outline

- The memory/processor speed gap
- Importance of Data Placement
- What's wrong with slotted pages?
- Partition Attributes Across (PAX)
  - Performance results
- Summary
Sanity Check: Basic Evaluation

- Main-memory resident R, numeric fields
- Query:
  ```sql
  select avg(a) 
  from R 
  where a >= Lo and a <= 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

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

<table>
<thead>
<tr>
<th>Hardware Resource</th>
<th>NSM</th>
<th>PAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch Mispredict</td>
<td>1200</td>
<td>800</td>
</tr>
<tr>
<td>Memory</td>
<td>450</td>
<td>300</td>
</tr>
<tr>
<td>Computation</td>
<td>200</td>
<td>150</td>
</tr>
</tbody>
</table>

PAX improves overall execution time

Sensitivity Analysis (2)

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

<table>
<thead>
<tr>
<th>Projectivity</th>
<th>NSM</th>
<th>PAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.5</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>4.5</td>
</tr>
</tbody>
</table>

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
Conclusions

- 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