15-823 Advanced Topics in Database Systems Performance

Main Memory Databases

Mengzhi Wang
April 5, 2001

What is MMDB?

- Disk resident databases (DRDB)
- Memory resident databases (MRDB)
- Main memory databases (MMDB)

- Diff: Where the primary copy resides

Memory vs. Disk

- Access time
  - Memory is faster than disk
- Access pattern
  - Memory is better for random access than disks
- Stableness
  - Memory is volatile
  - Disk is nonvolatile
- Security
  - Memory is more vulnerable to software errors

MMDB vs. DRDB

- Recovery
- Concurrency control
- Data organization
  - Relations, indices
- Query Processing
  - Cost model
  - Algorithms
- …

Outline

- Overview
- **MMDB techniques**
  - **Data organization**
  - **Query processing**
  - **Recovery**
- Case studies
- MMDB revisited

MMDB techniques

- Assumption
  - Memory offers uniform access time
  - Memory accesses are cheap

- Consequence
  - Pointers are cheap

Data Organization

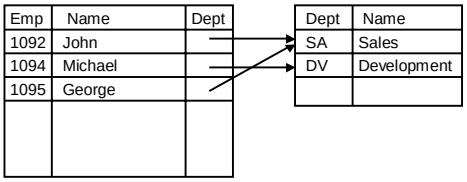
- Traditional DRDB
  - Relations: slotted pages
    - Two indirections for easy moving
    - Exploitation of the disk access pattern
  - Indices: B-tree
    - Shallow: pointer chasing is expensive for disks

Data Organization: Relations

- Relations
  - Still organized in pages?
  - How to deal with variable length fields?
  - How to deal with foreign keys?
- We may still want pages around
  - Units for recovery
  - Slotted pages? Maybe not

Data Organization: Relations

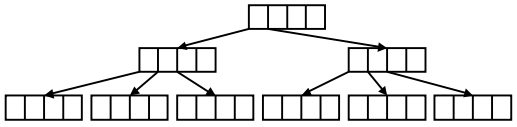
- Tuple Id: address
- Foreign key: Pointers



Data Organization: Relations

- Field values
  - Points to a domain
  - No need to worry about variable length fields
- Variable length fields
  - Pointers to heap space

Data Organization: Indices



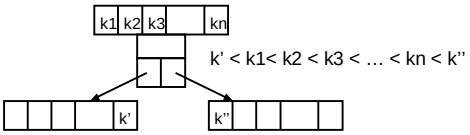
- B-tree
  - Shallow: good for disks
  - Space utilization:
    - Always > 50%
    - Keys ~ 50% among them

Data Organization: Indices

- Key values in indices
  - Store key values or pointers to tuples?
- Index structure
  - B-tree or something else?

Data Organization: T-tree

- Proposed by Lehman and Carey
- Modified binary AVL trees
  - Two pointers and more than one key values in each node.



Data Organization: T-tree

- Balanced by rotating nodes
- Number of rotation reduced by allowing a little variation in number of keys in each nodes
- Advantages
  - Space efficient
  - Logarithm performance

Outline

- Overview
- MMDB techniques
  - Data organization
  - **Query processing**
  - Recovery
- Case studies
- MMDB revisited

Query Processing

- Cost model
  - DRDB: I/O
  - MMDB: Computations

Query Processing

- Joins: pointer join

Emp	Name	Dept
1092	John	
1094	Michael	
1095	George	

Dept	Name
SA	Sales
DV	Development

Query Processing

- Temporary results
  - Use pointers to tuples instead of copying data

Emp	Name	Dept
1092	John	
1094	Michael	
1095	George	

temp

Dept	Name
SA	Sales
DV	Development

Outline

- Overview
- MMDB techniques
  - Data organization
  - Query processing
  - **Recovery**
- Case studies
- MMDB revisited

Recovery

- Recoverability
  - Frequency of failures
  - Data loss
- Performance
  - Transactions running faster
  - Committing still slow

Recovery

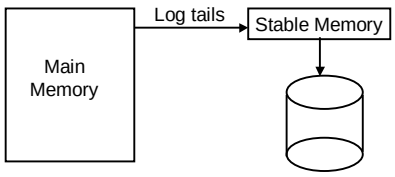
- Different Considerations
  - Commit process
  - Checkpointing
  - Reload

Commit Process

- Transactions: ACID
  - Durability: Log forced to disks at commit
- Problem: Log I/O becomes bottleneck
- How long do we need to keep the log?
  - Until the next checkpoint

Commit Process

- Solution 1:
  - Use stable memory for log tails



Commit Process

- Solution 2: Group commit
  - Accumulate log until page is full
  - Write a log page out only once
  - Also used in DRDB

Commit Process

- Solution 3: Precommit
  - Release the lock after logs in log buffer
  - Reduce blocking time of other transactions

Checkpointing

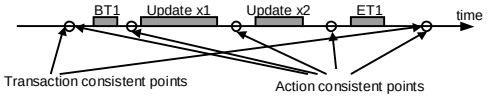
- Checkpoints in DRDB
  - Bring pages on disk up to date
  - Reduce the work of restart process
- Checkpoints in MMDB
  - Make a copy of the data on disks
  - Truncate the logs

Checkpointing

- Goal
  - Little interference with user transactions.
- Framework
  - Two copies of data on disk
  - Ping-pong algorithm
- Techniques
  - Non-fuzzy checkpointing
  - Fuzzy checkpointing

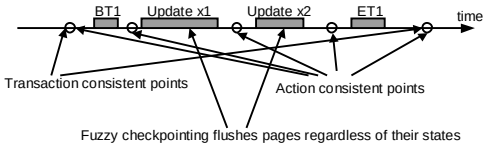
Checkpointing

- Non-fuzzy checkpointing
  - Action consistent or transaction consistent
  - Locks imposed by checkpointing process
  - Increasing lock contentions



Checkpointing

- Fuzzy checkpointing
  - Flush pages regardless of their states
  - Redo the operations during recovery



Reload

- Goal: fast
- Approach:
  - Simple: accept work after all the data is loaded
  - Fast: work after a small amount of data is reloaded

MMDB Summary

- Date organization
  - Use pointers as much as possible
- Recovery
  - Focus on performance
- Assumption:
  - Memory is cheap at random accesses

Outline

- Overview
- MMDB techniques
  - Data organization
  - Query processing
  - Recovery
- **Case studies**
- MMDB revisited

Case Study: Timesten

- Timesten (http://www.timesten.com)
  - Main memory database systems
  - Traditional system architecture
  - T-tree for indices
  - Ten times faster than DRDB

Case study: Dali

- http://www.bell-labs.com/project/dali/
- Main memory data storage system
- Goal: high performance
- Similar to ObjectStore
  - Data mapped to memory
  - Access through C++ API
  - Limited query capabilities

Case study: Dali

- Storage manager
  - Data organized as data files
  - Mapped to virtual memory address space
  - Pointers as offsets in files
  - Divided into segments
  - Different recovery mechanism for different data

Case study: Dali

- Transactions: Multi-level recovery
  - Physical undo and redo logs in memory for ongoing actions
  - Logical redo logs in system log at precommit
  - System log flushed to disk at commit
- Checkpointing:
  - Fuzzy

Case study: Dali

- Security
  - Process death
    - Release latches on data structures
    - Rollback transactions
  - Application errors
    - Memory protection
    - Codewords

Case study: Monet

- http://www.cwi.nl/
- Main memory databases
- Designed
  - For data warehouse apps
  - For cache performance

Case study: Monet

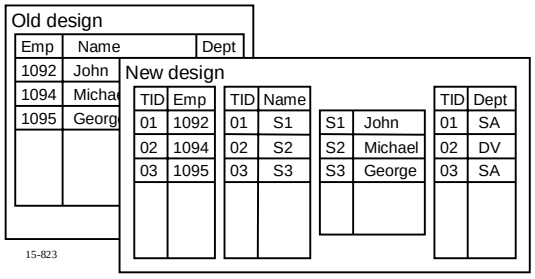
- Example

  Select name, salary
  From employee
  Where age < 35

- Observation:
  - Small strides offers better locality

Case study: Monet

- Vertical decomposing
- Special handling of strings



Outline

- Overview
- MMDB techniques
  - Recovery
  - Data organization
  - Query processing
- Case studies
- **MMDB revisited**

MMDB Revisited

- Assumption
  - Memory accesses are uniformly cheap
- Today's situation
  - Memory accesses are expensive
  - Memory accesses are not uniform
- Cache performance is important

Data Organization: Relations

- Pointers to domain?
- Foreign key as pointers?
- Variable length fields?

Data Organization: Relations

- Row major or column major?
- May be mixed
  - Group frequently accessed fields together

Data Organization: Indices

- Keys as pointers in indices?
- T-tree or B-tree or something else?
- T-trees are not good
  - Deep trees lead to heavy pointer chasing
- CSB-tree
- Prefetching B+-trees

Query Processing

- Cache-conscious processing
  - Similar techniques for memory-disk hierarchy
    - Partition
  - Blocking

Summary

- Previous MMDB work
  - Recovery
  - Index structures
  - Protection
- Today's focus
  - Cache awareness

References

- H. Garcia-Molina, K. Salem. Main memory database systems: an overview. TKDE 4(6), 1992.
- P. Bohannon et al. The architecture of the Dali main-memory storage manager. Journal of Multimedia Tools and Applications, 1997.
- Tobin J. Lehman, Michael J. Carey. A study of index structures for main memory database management systems. VLDB'86.
- K. Salem, H. Garcia-Molina. Checkpointing memory-resident databases. ICDE 89