ΒΔ Κύριας Μνήμης

Main Memory (or, In-Memory) Databases: An Overview

Εργαστήριο Πληροφοριακών Συστημάτων, Παν/μιο Πειραιώς (http://infolab.cs.unipi.gr/)

έκδοση: Ιανουάριος 2010

Introduction

- In a MMDB, data (at least a major portion) resides permanently in main physical memory
  - MMDB can provide much better res. time, throughput.
  - real time applications
- differences from magnetic disks
  - access time / volatile / direct access
  - access cost

Question I about a MMDB

- entire database fits in main memory?
  - Yes: real time application or limited DB size
  - No: partition the data into one or more logical DB, and store the hottest one in MM.

Question II about a MMDB

- Difference between a MMDB and a DRDB with a large cache?
  - different cases: index structure , access through a buffer manager
  - As DRDB perform more and more in-memory optimization, they become closer to MMDB.

Question III about a MMDB

- main memory is nonvolatile and reliable with special H/W ?
  - there is no "yes" or "no" answer.
  - there are several factors to force the frequency of backups up.
  - performance of the backup mechanism is important.

Impact of Memory Resident Data

- Concurrency Control
  - lock contention may not as important as DRDB.
  - large lock granules (e.g. relations) are appropriate.
  - serial transaction processing may be desirable.
  - a small number of bits in data to represent their lock status.

Impact of Memory Resident Data(Cont.)

- Commit processing
  - the need for a stable log
  - it may undermine the performance advantages of a MMDB
  - a small amount of stable main memory can be used as the log
  - pre-committing
  - group committing

Impact of Memory Resident Data(Cont.)

- Access Methods
  - hashing: fast lookup/updates, not space-efficient, not support range queries well.
  - T-Tree (see http://en.wikipedia.org/wiki/T-tree)
  - index structures can store pointers rather than the data.
- Data Representation
  - relational tuples can be represented as a set of pointers to data values.

Impact of Memory Resident Data(Cont.)

- Query Processing
  - sequential access is not faster than random access in MM.
  - using pointers make some relational operations can be performed efficiently.
  - query processor must focus on processing costs.
  - it is difficult to measure processing costs.

Impact of Memory Resident Data(Cont.)

- Recovery
  - checkpointing and failure recovery are the only access to the disk DB copy.
  - checkpointing should interfere as little as possible with transaction processing
  - restoring after a failure

Impact of Memory Resident Data(Cont.)

- performance
  - depends primarily on processing time.
  - performance of backup / checkpointing algorithms is critical.
- API and Protection
  - applications may be given the actual memory position of the object.
  - transactions can access objects directly without private buffer.
    - can modify unauthorized parts
    - using a special compiler

Backup slides

- Prototypes

MM-DBMS

- university of Wisconsin
- two-phase locking(granuels : relation)
- commit : stable log tail by segment
- data rep.
  - self-contained segments
  - extensive pointer use
- access : hashing / T-tree / pointer
- recovery
  - segments recovered on demand
  - recovery processor

MARS

- Southern Methodist University
- database processor/recovery processor
- two-phase locking
- commit: stable shadow memory, log tail
- recovery : fuzzy checkpoints

HALO(HArdware LOgging)

- HALO is a proposed special-purpose device for logging.
- for every write request, HALO creates a log entry (loc., new value, old value)
- log entries are maintained in nonvolatile buffers.
- special commands are used to inform HALO of transaction id.

OBE(Office-By-Example)

- run on the IBM 370 architecture
- handling *ad hoc* queries rather than high update loads.
- data rep.: extensive use of pointer
- access : inverted index
- query
  - nested loop-join
  - focus on reducing processing costs

TPK

- runs on firefly multiprocessor
- transaction processing system, simple data model.
- implemented at Princeton Univ.
- consists of a set of concurrent threads : input/execution/output/checkpoint
- serial transaction execution
- commit : group commit, pre-commit
- recovery : two MMDB, fuzzy checkpoints

System M

- transaction processing testbed system
- developed at Princeton Univ.
- collection of servers on the Mach OS.
  - message/transaction/log/checkpoint
- two-phase locking, pre-commit, group commit
- minimize concurrency
- a variety of checkpointing and logging techniques are implemented

IMS/VS Fast Path

- commercial database product from IBM
- each DB is classified statically as either memory or disk.
- group commit
- lock requests is optimized(record-granule)
- VERIFY/CHANGE for hot spots