CSE544
Database Architecture

Tuesday, February 1st, 2011

Slides courtesy of Magda Balazinska
Where We Are

• **What we have already seen**
  – Overview of the relational model
    • Motivation and where model came from
    • Physical and logical independence
  – **How to query a database**
    • Relational calculus
    • SQL
  – Transactions

• **Where we go from here**
  – How can we efficiently implement this model?
References


Outline

DBMS architecture

• Main components of a modern DBMS
• Process models
• Storage models
• Query processor (we will cover this part in lecture 6)
DBMS Architecture

Process Manager
- Admission Control
- Connection Mgr

Query Processor
- Parser
- Query Rewrite
- Optimizer
- Executor

Storage Manager
- Access Methods
- Lock Manager

Shared Utilities
- Memory Mgr
- Disk Space Mgr
- Replication Services
- Admin Utilities

[Anatomy of a Db System. J. Hellerstein & M. Stonebraker.]

Dan Suciu -- 544, Winter 2011 Red Book. 4ed.]
Process Model

Why not simply queue all user requests? (and serve them one at the time)

Alternatives
1. Process per connection
2. Server process (thread per connection)
   - OS threads or DBMS threads
3. Server process with I/O process

Advantages and problems of each model?
Process Per Connection

• **Overview**
  – DB server forks one process for each client connection

• **Advantages**
  – Easy to implement (OS time-sharing, OS isolation, debuggers, etc.)
  – Provides more physical memory than a single process can use

• **Drawbacks**
  – Need OS-supported “shared memory” (for lock table, for buffer pool)
    • Since all processes access the same data on disk, need concurrency control
  – Not scalable: memory overhead and expensive context switches
Server Process

• **Overview**
  – DB assigns one thread per connection (from a thread pool)

• **Advantages**
  – Shared structures can simply reside on the heap
  – Threads are lighter weight than processes (memory, context switching)

• **Drawbacks**
  – Concurrent programming is hard to get right (race conditions, deadlocks)
  – Portability issues can arise when using OS threads
  – **Big problem**: entire process blocks on synchronous I/O calls
    • Solution 1: OS provides asynchronous I/O (true in modern OS)
    • Solution 2: Use separate process(es) for I/O tasks
DBMS Threads vs OS Threads

• Why do DBMSs implement their own threads?
  – Legacy: originally, there were no OS threads
  – Portability: OS thread packages are not completely portable
  – Performance: fast task switching

• Drawbacks
  – Replicating a good deal of OS logic
  – Need to manage thread state, scheduling, and task switching

• How to map DBMS threads onto OS threads or processes?
  – Rule of thumb: one OS-provided dispatchable unit per physical device
  – See page 9 and 10 of Hellerstein and Stonebraker’s paper
Historical Perspective (1981)

- See paper: “OS Support for Database Management”

- No OS threads
- No shared memory between processes
  - Makes one process per user hard to program
- Some OSs did not support many to one communication
  - Thus forcing the one process per user model
- No asynchronous I/O
  - But inter-process communication expensive
  - Makes the use of I/O processes expensive

- Common original design: DBMS threads
Commercial Systems

• **Oracle**
  – Unix default: process-per-user mode
  – Unix: DBMS threads multiplexed across OS processes
  – Windows: DBMS threads multiplexed across OS threads

• **DB2**
  – Unix: process-per-user mode
  – Windows: OS thread-per-user

• **SQL Server**
  – Windows default: OS thread-per-user
  – Windows: DBMS threads multiplexed across OS threads
Admission Control

• Why does a DBMS need admission control?

• When does DBMS perform admission control?
Admission Control

• **Why does a DBMS need admission control?**
  – To avoid thrashing and provide “graceful degradation” under load

• **When does DBMS perform admission control?**
  – In the dispatcher process: want to drop clients as early as possible to avoid wasting resources on incomplete requests
  – Before query execution: delay queries to avoid thrashing
Outline

• History of database management systems

• DBMS architecture
  – Main components of a modern DBMS
  – Process models
  – Storage models
  – Query processor
Storage Model

• **Problem**: DBMS needs spatial and temporal control over storage
  – Spatial control for performance
  – Temporal control for correctness and performance

Alternatives
• Use “raw” disk device interface directly
• Use OS files
Spatial Control
Using “Raw” Disk Device Interface

• **Overview**
  – DBMS issues low-level storage requests directly to disk device

• **Advantages**
  – DBMS can ensure that important queries access data sequentially
  – Can provide highest performance

• **Disadvantages**
  – Requires devoting entire disks to the DBMS
  – Reduces portability as low-level disk interfaces are OS specific
  – Many devices are in fact “virtual disk devices”
Spatial Control
Using OS Files

• **Overview**
  – DBMS creates one or more very large OS files

• **Advantages**
  – Allocating large file on empty disk can yield good physical locality

• **Disadvantages**
  – OS can limit file size to a single disk
  – OS can limit the number of open file descriptors
  – But these drawbacks have mostly been overcome by modern OSs
Historical Perspective (1981)

• **Recognizes mismatch problem between OS files and DBMS needs**
  – If DBMS uses OS files and OS files grow with time, blocks get scattered
  – OS uses tree structure for files but DBMS needs its own tree structure

• **Other proposals at the time**
  – Extent-based file systems
  – Record management inside OS
Commercial Systems

• Most commercial systems offer both alternatives
  – Raw device interface for peak performance
  – OS files more commonly used

• In both cases, we end-up with a DBMS file abstraction implemented on top of OS files or raw device interface
Temporal Control
Buffer Manager

• Correctness problems
  – DBMS needs to control when data is written to disk in order to provide transactional semantics (we will study transactions later)
  – OS buffering can delay writes, causing problems when crashes occur

• Performance problems
  – OS optimizes buffer management for general workloads
  – DBMS understands its workload and can do better
  – Areas of possible optimizations
    • Page replacement policies
    • Read-ahead algorithms (physical vs logical)
    • Deciding when to flush tail of write-ahead log to disk
Historical Perspective (1981)

- Problems with OS buffer pool management long recognized
  
  - Accessing OS buffer pool involves an expensive system call
  - Faster to access a DBMS buffer pool in user space
  
  - LRU replacement does not match DBMS workload
  - DBMS can do better
  
  - OS can do only sequential prefetching, DBMS knows which page it needs next and that page may not be sequential
  
  - DBMS needs ability to control when data is written to disk
Commercial Systems

- DBMSs implement their own buffer pool managers

- Modern filesystems provide good support for DBMSs
  - Using large files provides good spatial control
  - Using interfaces like the mmap suite
    - Provides good temporal control
    - Helps avoid double-buffering at DBMS and OS levels
Outline

• History of database management systems

• **DBMS architecture**
  – Main components of a modern DBMS
  – Process models
  – Storage models
  – Query processor (will go over the query processor in lecture 7)
Summary

- Past few weeks: transactions
- Today: overview of architecture of a DBMS
- Next few weeks
  - Storage and indexing
  - Query execution
  - Query optimization
  - Distribution
  - Parallel processing