Database Tuning - Introduction

Vu Tuyet Trinh
trinhvt@soict.hust.edu.vn

Department of Information Systems
SoICT-HUST

Copyrights:

Many slides belong to the tutorial:
Database Tuning
Principles, Experiments and Troubleshooting Techniques
Dennis Shasha (shasha@cs.nyu.edu)
Philippe Bonnet (bonnet@diku.dk)

and some from the web …

What is Database Tuning?

Activity of making a database application run faster:

– Faster means higher throughput (or response time)

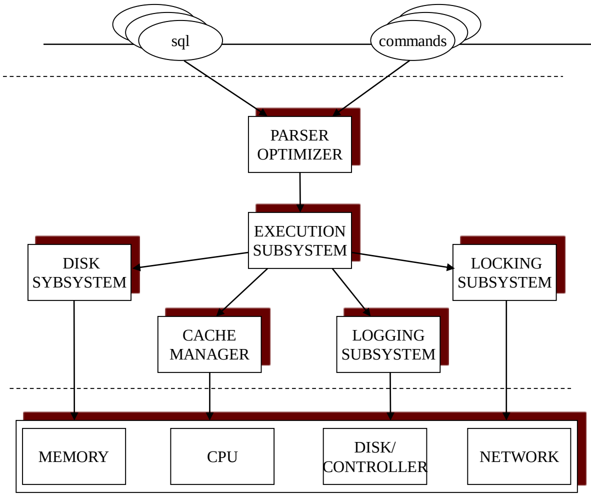
– Avoiding transactions that create bottlenecks or avoiding queries that run for hours unnecessarily is a must.

– A 5% improvement is significant.

Why Database Tuning?

- Troubleshooting:
  - Make managers and users happy given an application and a DBMS
- Capacity Sizing:
  - Buy the right DBMS given application requirements
- Application Programming:
  - Coding your application for performance

Why is Database Tuning hard?



sql

commands

PARSER
OPTIMIZER

EXECUTION
SUBSYSTEM

DISK
SYBSYSTEM

LOCKING
SUBSYSTEM

CACHE
MANAGER

LOGGING
SUBSYSTEM

MEMORY

CPU

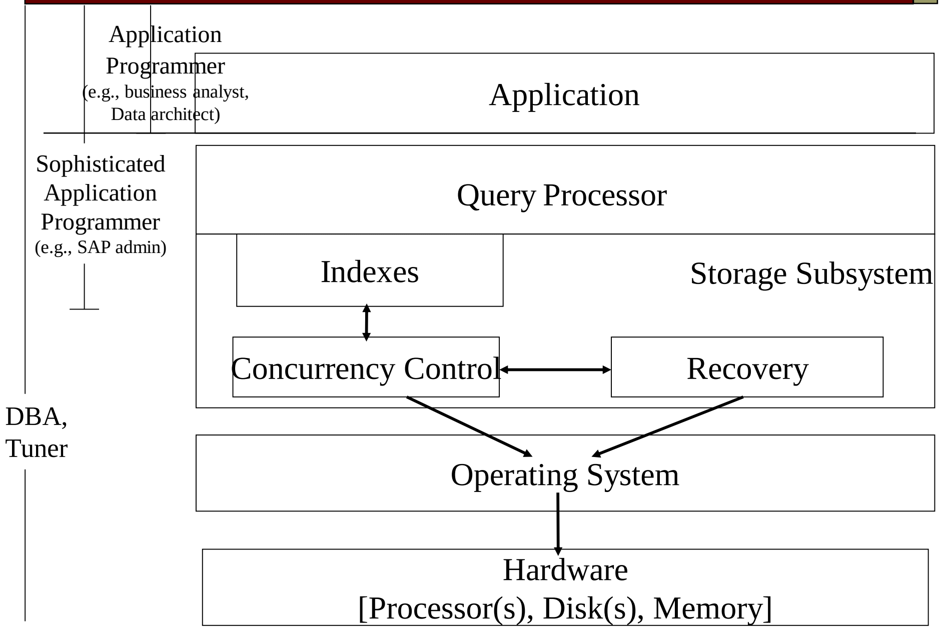
DISK/
CONTROLLER

NETWORK

The following query runs too slowly

```
select *
from R
where R.a > 5;
```

What do you do?



Application
Programmer
(e.g., business analyst,
Data architect)

Application

Sophisticated
Application
Programmer
(e.g., SAP admin)

Query Processor

Indexes

Storage Subsystem

Concurrency Control

Recovery

DBA,
Tuner

Operating System

Hardware
[Processor(s), Disk(s), Memory]

Outline

- Schema tuning
- Index tuning
- Query Processing
- Query tuning
- Transaction Management
- Transaction tuning
- Tuning Distributed Application

Tuning Principles

1. Think globally, fix locally
2. Partitioning breaks bottlenecks
   - temporal and spatial
3. Start-up costs are high; running costs are low
4. Render unto server what is due unto server
5. Be prepared for trade-offs

Think globally, fix locally

- Proper identification of problem; minimal intervention
- Understand the whole, including the application goals before taking a set of queries and find the indexes that speed them up.
- Example:
  - High I/O, paging and processor utilization may be due to frequent query scans instead of using an index or log sharing a disk with some frequently accessed data.

Partitioning breaks bottlenecks

- Technique for reducing the load on a certain component of the system either by dividing the load over more resources or by spreading the load over time
- Partitioning may not always solve bottleneck:
  - First, try to speed up the component
  - If it doesn’t work, partition
- Example:
  - Lock and resource contention among long and short transactions

Start-up costs are high; running costs are low

- Obtain the effect you want with the fewest possible start-ups
- Examples:
  - It is expensive to begin a read operation on a disk, but once it starts disk can deliver data at high speed.
    - So, frequently scanned tables should be laid out consecutively on disk.
  - Cost of parsing, semantic analysis, and selecting access paths for simple queries is significant
    - So, often executed queries should be compiled

Render unto server what is due unto server

- Important design question is the allocation of work between the DB system (server) and the application program (client)
- Depends on:
  - Relative computing resources of client and server
  - Where the relevant information is located
  - Whether the DB task interacts with the screen

Be prepared for trade-offs

- Increasing speed of application requires combination of memory, disk and computational resources
- Examples:
  - Adding an index => speeds up critical query, but increases disk storage, and space in RAM
  - Increasing RAM => Decreasing I/O, speed up query, but spending more money