Supplementary slides for introduction

Rasmus Pagh
Information explosion

Figure 2

Information Versus Available Storage

Source: IDC, 2007
Information explosion, cont.

Figure 3

Organizational Information “Unit” Growth WW

Source: IDC, 2007
Database Tuning, Spring 2009

Intel Core 2 Duo Processor
3.0GHz, E8400 Wolfdale

Core 1
Throughput: ~1 instruction per cycle. One cycle takes ~0.33 nanoseconds. The exact number of cycles depends on the instruction.

L1 Data Cache
32 KB
Latency: 1ns (3 cycles)

L1 Instruction Cache
32 KB

Core 2
Same as Core 1.

L1 Data Cache

L1 Instruction Cache

L2 Cache
6MB
Latency: 4.7ns (14 cycles)

Front Side Bus
1333MHz DDR3
Bandwidth: 10GB/s

PCI Express x16. 8GB/s (each way)

Intel X48 Northbridge chip

RAM Modules - 8GB
Latency: ~83 ns (~250 cycles)

http://duartes.org/gustavo/blog
Sequential vs random access

- My laptop can, in 1 second:
  - Perform up to 20 billion CPU instructions
  - Read 0.8 billion 4-byte words, sequentially
  - Read 0.034 billion words, random access

```
Rasmus-Pagh$./linear Generating 8388608 pointer-value pairs
100 linear probing sequences begin
End: 838860700
1.000000 seconds

100 double hashing sequences begin
End: 1677721400
24.320000 seconds
```
Lecture overview

Rasmus Pagh
The lectures at a glance

• Data storage, tree indexes.
• Hash indexes, index tuning
• Impl. of relational operations, external sorting
• Query optimization, query tuning
• Concurrency control

• Decision support, OLAP
• Data mining
• Temporal databases
• Text indexing
• Spatial databases
• ITU research in databases

Exciting invited lectures are mixed in!
Tree indexes

- **B-trees**, a generalization of binary search trees, is the most important index type in DBMSs.
- You will get an understanding of what functionality B-trees offer, and how they are updated when the data changes.
- **Buffered B-trees**, a new B-tree variant that has exceptionally good update performance, is presented.
Hash indexes, index tuning

• External memory hash tables generalize hash tables as you know them.
• Faster than B-trees in some situations.
• Need to understand to choose!

• We will discuss general issues about how to choose the right indexes, and good physical organization of data in general (sparse vs dense, partitioning).
Relational algebra operations

• The building blocks in DBMS query evaluation are algorithms that implement relational algebra operations.

• May be based on:
  – sorting (quicksort is bad!),
  – hashing, or
  – using existing indexes

• The DBMS knows the characteristics of each approach, and attempts to use the best one in a given setting.
Query optimization, query tuning

- Query optimization is the process where the DBMS tries to find the “best possible” way of evaluating a given query.
- Standard approach builds on finding a “good” relational algebra expression and then choosing how and in what order the operations are to be executed.
- Query tuning is a “manual” effort to make query execution faster.
Concurrency control

• For databases with many users, the concurrency control mechanisms of a DBMS can cause performance problems.
• DBMSs are distinguished by their design of concurrency control system
  – Pessimistic (locking based) vs optimistic
  – Granularity
• To handle concurrency control problems, an understanding of the system in use is often required.
Second part

• First part is “classical DBMS” topics.
• Rest of course:
  – Extensions of capability in various settings…
  – Main tool: Efficient indexing
Decision support (OLAP)

• OLAP systems are specialized databases for decision support applications.

• Idea: Read-only (or write-rarely), optimized for fast answers to queries.

• Special indexing techniques for read-only data are used (bitmap indexing).

• Precomputation of aggregates important for performance.
Data mining

• Normal database operations follow a query-answer pattern:
  – To analyze data you may perform a sequence of queries.

• Data mining is another paradigm:
  – “Let the software find the interesting properties of the data”

• We will look at corellation mining, and see how it can be implemented in a scalable way.
Temporal databases

- It is increasingly feasible to never delete data (i.e., keep old versions)
- ⇒ Demand for capability to query old data.
- Need indexing capability also for old data!
- You will see surprisingly efficient ways of doing this.
Spatial databases

• Many large databases contain geographical data.
• In general, many data sets can be viewed as points in a multi-dimensional space. **Example:** (salary, age) pairs.
• Need for efficient indexes that allow the DBMS to find part of the space. **Example:** “Find all tuples with age below 30 and salary above 500,000”.
Text indexing

• Many database applications contain lots of text
• But in the relational model it is complicated to represent and query the structure of text.
• In practice, a text datatype that may contain long strings that have to be handled in queries.
• We look at two topics:
  – B-trees optimized for strings
  – Full-text indexing
ITU research in databases

- An overview of some results by ITU researchers on (or related to) performance aspects of databases.
- Mainly theoretical work - chance to be the first in the world to implement and test!
- Especially meant to serve as inspiration for formulating possible thesis projects.
Invited lectures

• Philippe Bonnet, DIKU (author of “Database Tuning”). Topic: TBA.
• Someone from LECTOR: Data partitioning in Oracle.
• Rasmus Resen Amossen, ITU (to be confirmed): The H-store database system.
• Jesper Larsson, Apptus Technologies (to be confirmed): Relational text indexing.
• ...
• As part of the project, each group will have to write a summary of one of the invited lectures.
The project

- Database development project (in groups of 4).
- Use of the database will be simulated by a java program supplied to you.
- Your task:
  - Make a good database design.
  - Implement various query and update ops.
  - Tune for performance.
  - Have fun!
- More information next week. Project kick-off on Tuesday February 10.