Advanced Topics in Databases

Introduction

Gunter Saake, Sebastian Breß, David Broneske, Sebastian Dorok, Andreas Meister
Otto-von-Guericke University Magdeburg
Summer Term 2015
Motivation for the Lecture

• Familiarize students with current developments in database research

• Topics chosen:
  • First solutions currently make their way into database management systems and applications → practical relevance
  • Solutions not yet fully developed and open problems exist → research relevance

• Possible starting points for scientific work, e.g. master thesis, position in academia, Ph.D. thesis, etc.
Yesterday’s DBMS Landscape

Oracle
IBM DB2
Microsoft SQL Server
Teradata

Oracle
IBM
Microsoft SQL Server
Teradata

Application
"Banking, SAP, ..."
"Server"
"Disk"
Yesterday’s DBMS Hardware

Small main memory

Disk-based systems

Picture taken from [1]

Picture taken from [2]
Assumptions of yesterday’s DBMSs

- Capacity of main memory <1% of the stored data
- Fixed block size based on the transfer unit between disks and main memory
- Central scheduler to schedule transactions
- No redundant data storage in main memory
- Pipelining is always beneficial (no storage of intermediate results)
- Compiling of SQL for one processor architecture → Reuse of compiled plan
Today’s DBMS Topics

Genomic Data In Memory
Co Processing
Parallel Processing
Stream Processing
New Storage Devices
Cloud
XML
Social Networks
Graph Database
Security
Geographical Databases
Data Warehouses
Internet of Things
Scientific Data
Big Data
Data Provenance
Today’s DBMS Hardware

Large main memory

Multi-core CPUs

Solid state disks

Co-processors
Future DBMSs

- Capacity of main memory <1% of the stored data
  - *DB in main memory*
- Fixed block size based on the transfer unit
  - *direct access of data on all devices*
- Central scheduler to schedule transactions
  - *which processor should do the job?*
- No redundant data storage in main memory
  - *redundant data at co-processors*
- Pipelining is always beneficial
  - *co-processors like GPU support massive parallelism*
- Reuse of compiled plan
  - *load-balancing between co-processors requires different plans*
The Goals of a "Databaser"

• Performance

Picture taken from [6]
The Goals of a "Databaser"

- Performance
- Performance

Picture taken from [6]
The Goals of a "Databaser"

- Performance
- Performance
- Performance

Picture taken from [6]
The Goals of a "Databaser"

- Performance
- Performance
- Performance

How can we achieve more performance?

Picture taken from [6]
Trends for DBMS’s

- Use main memory as primary storage → Speed up data access

- Exploit all hardware capabilities such as co-processors → Speed up database operators
Are DBMSs written for yesterdays hardware efficient on todays hardware as well?
Are DBMSs written for yesterdays hardware efficient on todays hardware as well?

”30 years of Moore’s law has antiquated the disk-oriented relational architecture for OLTP applications” [Stonebraker et al., 2007]
Data Access – Yesterday’s Bottleneck
Data Access – Today’s Bottleneck

Bottleneck
The World of Co-Processors

PCI Express Bus

Picture taken from [7]
What do we have to change in DBMSs’ architecture to exploit new hardware capabilities and to meet tomorrow’s challenges?
Overview of Topics

1. Main-Memory DBMSs (Dorok, 3 lectures)

2. Hardware-sensitive DB Algorithms (Broneske, 2 lectures)

3. Advanced Query Optimization (Meister, 2 lectures)

4. GPU-accelerated Data Management (Breß, 3 lectures)
Main-Memory DBMSs

Address main-memory access bottleneck using cache-conscious

• data layouts (e.g., column-stores [Abadi et al., 2008]) and
• data processing [Manegold et al., 2000]

Query processing in row-stores

SELECT SUM (Sales)
FROM Shop
WHERE Location = 'MD'

σLocation = 'MD'

πSales

SUM

7
1
Sales

Adapted from [Köppen et al., 2012]
Hardware-sensitive DB Algorithms

**CPU "smaller than"-selection**

```c
tag pos = 0;
for (int i=0; i < array_size; ++i){
    if (array[i] < comp_val)
        result[pos++] = i;
}
```

**GPU "smaller than"-selection**

```c
tag tid = threadIdx.x + blockIdx.x * blockDim.x;
while (tid < array_size){
    bitmask[tid] = (array[tid] < comparison_value);
    tid += blockDim.x * gridDim.x;
}
```

Optimized Code differs between processing devices w.r.t.

- Access pattern
- Code optimizations (e.g., branch-free code)
- Parallelization capabilities
Advanced Query Optimization

\[
\text{r(ORDER)} \bowtie \text{r(PRODUCT)} \bowtie \text{r(CUSTOMER)} \bowtie \text{r(SUPPLIER)}
\]

Equivalent plans for one query \(\Rightarrow\) Large search space

Search space increases in modern systems
\(\Rightarrow\) Advanced optimization algorithms needed:
- Parallelization
- Usage of co-processors
GPU-accelerated Data Management

- Specialized GPU operators
- Predicting the benefit of GPU acceleration
- Data placement strategies
- Increased complexity of query optimization
Organization

• **Lecture**
  • Every week Monday at 9:15 in room G10-111
  • Two to four weeks for each topic read by Gunter Saake, Sebastian Breß, Sebastian Dorok, David Broneske, Andreas Meister
  • Lecture slides will be made available on lecture homepage http://wwwiti.cs.uni-magdeburg.de/iti_db/lehre/advdb/

• **Exercise**
  • Theoretical exercises based on Exercise sheets available on the lecture homepage, held by Andreas Meister
  • Every week Tuesday at 9:15 in room G05-208
  • Starting on April 28

• **Exam**
  • Prerequisite: Small *homework* and *presentation* about one advanced database technique
  • Oral exam of 20-30 minutes after end of lecture period
References

Column-Stores vs. Row-Stores: How different are they really?
In *SIGMOD*, pages 967–980.

*Data Warehouse Technologien*.
mitp-Verlag.


*Datenbanken: Implementierungstechniken*.
mitp-Verlag, 3rd edition.

The end of an architectural era: (it’s time for a complete rewrite).
Web Resources


