Distributed Data Management and Distributed Databases

Christoph Lofi
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de
1.0 Organizational Issues

- Lecture
    - 9:45 – 12:15 hr. (3 academic hours lecture with a short intermediate break)
  - Exercises, detours, and home work discussion integrated into lecture

- 5 Credits

- Exams
  - Oral Exams
1.0 Introduction

1.1 Distributed Databases
1.2 Peer-To-Peer Systems
1.3 Cloud Storage
1.0 Recommended Literature

- Distributed literature

- P2P literature

- But: Most later parts will rely on research papers
• **Relational Databases** developed successfully since the 1960s
• **Classic Example:** **Banking Systems**
  – Huge amounts of data on customers, accounts, loans, balances,…
• In the beginning, a **central DBMS** was responsible for all the tasks
  
  – **Typically, all meta-data and processing capacity** was concentrated on one room-sized ultra-expensive machine
  
  – Each branch had its own database

  • Cross-branch transactions were handled manually
1.0 Distributed Data

• Need for **data integration**
  – e.g. access account data of another branch, perform cross-branch transactions

• Need for **resource sharing**
  – Use existing hardware in a more efficient manner

• Need for **fail-safe data storage**
  – No single-point-of-failure
  – Disaster resistance

• **Organizational structures** should be reflected in IT infrastructure
• **Integration** of several DBMS or running one DBMS at multiple hardware machines leads to **distributed databases**
  
  – Distributed databases provide all features of **relational databases**
  
  – First major topic of this lecture!
A distributed database is a DB (**DDB**) where data is stored on several **nodes** in a **network**

- A **distributed DBMS** (**DDBMS**) is thus a database management system managing several data nodes

**DDBMS** are **“real databases”** with the full feature set of relational databases

- Later in this lecture, we will focus on distributed data storage systems which **sacrifice features** for the sake of **performance** and **flexibility**
1.1 Towards Distributed DB

• Relational Databases established a set of valuable features
  – Controlled redundancy
  – Data normalization
  – Data consistency & integrity constraints
  – Powerful query languages
  – Effective and secure data sharing
  – Backup and recovery
### 1.1 Characteristics of Databases

- Databases are **well-structured** (e.g. ER-Model)
  - **Catalog** (data dictionary) contains all **meta-data**
  - Defines the **structure** of the data in the database

- Example: ER-Model
  - Simple banking system

![Entity-Relationship Diagram](image)

- **customer** has:
  - ID
  - `firstname`
  - `lastname`
  - `address`

- **account** has:
  - `AccNo`
  - `balance`
  - `type`
1.1 Characteristics of Databases

• Databases aim at **efficient** manipulation of data
  – Physical tuning allows for good data allocation
  – Indexes speed up search and access
  – Query plans are optimized for improved performance

• Example: Simple Index

<table>
<thead>
<tr>
<th>Index File</th>
<th>Data File</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccNo</td>
<td>type</td>
</tr>
<tr>
<td>1278945</td>
<td>saving</td>
</tr>
<tr>
<td>2437954</td>
<td>saving</td>
</tr>
<tr>
<td>4543032</td>
<td>checking</td>
</tr>
<tr>
<td>5539783</td>
<td>saving</td>
</tr>
<tr>
<td>7809849</td>
<td>checking</td>
</tr>
<tr>
<td>8942214</td>
<td>checking</td>
</tr>
<tr>
<td>9134354</td>
<td>saving</td>
</tr>
<tr>
<td>9543252</td>
<td>saving</td>
</tr>
</tbody>
</table>
1.1 Characteristics of Databases

• **Isolation** between applications and data
  – Database employs **data abstraction** by providing **data models**
  – Applications work only on the **conceptual representation** of data
    • Data is strictly **typed** (Integer, Timestamp, VarChar,...)
    • Details on where data is actually stored and how it is accessed is **hidden** by the DBMS
    • Applications can access and manipulate data by invoking **abstract operations** (e.g. SQL Select statements)
  – DBMS-controlled parts of the file system are **strongly protected** against outside manipulation (tablespaces)
• **Example:** Schema is changed and table-space moved without an application noticing

```sql
SELECT AccNo FROM account WHERE balance > 0
```
1.1 Characteristics of Databases

**Example:** Schema is changed and table-space moved without an application noticing.
1.1 Characteristics of Databases

- Supports multiple **views** of the data
  - Views provide a different perspective of the DB
    - A user’s conceptual understanding or task-based excerpt of all data (e.g. aggregations)
    - Security considerations and access control (e.g. projections)
  - For the application, a view does not differ from a table
  - Views may contain **subsets** of a DB and/or contain **virtual data**
    - Virtual data is **derived** from the DB (mostly by simple SQL statements, e.g. joins over several tables)
    - Can either be computed at query time or **materialized** upfront
1.1 Characteristics of Databases

- Example Views: **Projection**
  - Saving account clerk vs. checking account clerk

<table>
<thead>
<tr>
<th>Original Table</th>
<th>Saving View</th>
<th>Checking View</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AccNo</strong></td>
<td><strong>type</strong></td>
<td><strong>balance</strong></td>
</tr>
<tr>
<td>1278945</td>
<td>saving</td>
<td>€ 312.10</td>
</tr>
<tr>
<td>2437954</td>
<td>saving</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td>4543032</td>
<td>checking</td>
<td>€ -43.03</td>
</tr>
<tr>
<td>5539783</td>
<td>saving</td>
<td>€ 12.54</td>
</tr>
<tr>
<td>7809849</td>
<td>checking</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td>8942214</td>
<td>checking</td>
<td>€ -345.17</td>
</tr>
<tr>
<td>9134354</td>
<td>saving</td>
<td>€ 2.22</td>
</tr>
<tr>
<td>9543252</td>
<td>saving</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>
1.1 Characteristics of Databases

• **Sharing** of data and support for **atomic multi-user** transactions
  
  – Multiple user and applications may access the DB at the same time
  
  – **Concurrency control** is necessary for maintaining consistency
  
  – Transactions need to be **atomic** and **isolated** from each other
Transactions

**Atomicity:** Either all operations of the transaction are properly reflected in the database or none are.

**Consistency:** Execution of a transaction in isolation preserves the validity of the database.

**Isolation:** Each transaction must be unaware of other concurrently executing transactions.

**Durability:** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.
1.1 Towards Distributed DBs

• DDBMSs maintain all these features in a distributed environment
  – The area of DDBMS is thus within the area of databases and computer networks

• Potential additional advantages
  – Increased robustness vs. hardware or site failures
  – Increased performance due to parallelization and load balancing
  – Scalable storage capacity
  – Easy access to several shared data sources without complex integration
• Distributed databases can be categorized with regard to the **autonomy** of each node and the overall **heterogeneity** of all nodes

  – **Autonomy**: nodes are just loosely coupled with others (or not at all)
  
  – **Heterogeneity**: different DBMS may be used
• **Homogeneous Distributed Databases**
  – **Regular Distributed Database:**
    • Each node is controlled by the same DDBMS
      – Globally shared schema
    • Nodes are linked via **slow** wide area network
      – Nodes are semi-autonomous
      – Data is usually manually distributed between the sites
    • Typical examples:
      – Databases distributed over several branches / sites (e.g. banking, insurance companies, etc.)
### 1.1 Foundations of DDBs

**Parallel Distributed Database:**
- Each node is controlled by the same DDBMS
  - Globally shared schema
- Nodes are linked via **high-speed** local area network
  - DDBMS distributes data automatically among nodes
  - Nodes show **no autonomy**
- Typical examples:
  - High performance data center
1.1 Foundations of DDBs

- **Heterogeneous Distributed Database**
  - Multi Database:
    - Each node is *autonomous* and may run any DBMS software
    - Central node passes queries to the nodes
      - No integration between the nodes
      - No shared schema / views
1.1 Foundations of DDBs

– **Mediator Database**
  
  • Each node is **autonomous** and may run any DBMS software
  
  • Central mediator node passes queries to the nodes
    – Mediator provides some **shared views** of a subset of all data
    – Mediator maps all queries to the individual schemas of the respective nodes

  • Typical examples:
    – Distributed bio/chemical databases
1.1 Foundations of DDBs

- **Federated Database**
  - Each node is autonomous and may run any DBMS software
  - Central federation node passes queries to the nodes
    - Federator provides a *globally shared schema*
    - Federator maps all queries to the individual schemas of the nodes and transforms the results to the shared schema
  - Typical examples:
    - Big cooperate database systems
• **Example:** distributed data systems are important in astronomy
  
  — No site can hold all information

  • Telescope image archives are already in the multi-TB range
  • Promise to quickly grow larger with the increasing size of digital detectors and the advent of new all-sky surveys
1.1 DDBMS in Science

• Much of the astronomical information is **dynamic**
  – Static catalogs and indexes quickly become obsolete

• **Astronomers use multiple types** of data
  – images, spectra, time series, catalogs, journal articles,...
  – All should be easily located and easily accessed with query terms and syntax natural to the discipline

• **Astronomers need to know the provenance** of the data they are using and **all details** about it
  – No one data center is able to have expertise in the wide range of astronomical instrumentation and data sets
1.1 DDBMS in Science

• Sample distributed datasets at NASA

  Solar System Exploration – Lunar and planetary science data and mission information

  Heliophysics – Space and solar physics data and mission information

  Universe Exploration – Astrophysics data and mission information

http://nssdc.gsfc.nasa.gov/
• **Naval command systems**
  – **Collate information such as:**
    • Sensor data (RADAR)
    • Geographic data (Maps)
    • Technical information (Ship types)
    • Air, land, surface and underwater data
    • ...
  – **Highly interactive**
    • Operator may annotate and extend any given data
      – Many operators at a time
      – Each operator should see all annotations in real time
• **Hard requirements for the system:**
  
  – Consistent, up-to-date view on the situation
  – Distributed environment
  – Many write operations (sensor data)
  – High fault-safety
  – Real-time requirements
• **BAE Systems**
  
  – British defense, security and aerospace company
  
  – Creates electronic systems and software for e.g. “Eurofighter Typhoon” or “Queen Elizabeth class aircraft carriers”
  
  – This includes development of **naval command systems**
• **Distributed Databases** for sharing data among systems
  
  – Fulfill the ACID characteristics
    • Ensure **consistency**
    • Allow for **parallel access**
    • Ensure **durability** and **fault-recovery**
  
  – Work in a distributed environment
1.1 Transparency

- DDBMS can be further classified with respect to the degree of distribution transparency
  - **Schema Transparency**
    - During schema design, do I have to bother with distribution?
  - **Query Transparency**
    - When querying, do I need to know where the data is?
  - **Update Transparency**
    - When updating, do I need to specify which data goes where?
1.1 Transparency

• **Schema Transparency**
  – DB admin has the impression of using a single-node database
  – DDBMS decides automatically how to distribute data
    • “automatic sharding”

• **Advantages**
  – No considerations about distribution necessary

• **Challenges**
  – Semantic knowledge on the data is often not used
    • e.g. each branch of a business usually only accesses its own employees
  – Usually, high speed network is required
1.1 Transparency

• **Query Transparency**
  – The user or application needs no knowledge on where the data is stored when querying

• **Advantages**
  – Easy querying as when using a single node DB

• **Challenges**
  – **Distributed query optimizer** necessary
  – Statistical data on potential storage locations required
1.1 Transparency

• **Update Transparency**
  – The user or application doesn’t need any knowledge on where the data is stored when updating
  – Global **consistency** is ensured

• **Advantages**
  – Easy updating as when using a single node DB

• **Challenges**
  – **Distributed transaction manager** necessary
1.1 Important Issues

• In order to implement an DDBMS, several issues are important

• Data Distribution
  – Partitioning & Sharding
  – Topic of 2\textsuperscript{nd} lecture

• Fail Safety, Load Balancing & Replication
  – Topic of 3\textsuperscript{rd} lecture

• Distributed Transactions
  – Topic of 4\textsuperscript{th} lecture
Distributed DBs offer solid and valuable features. However, these features come at a cost:

- **Limited scaling**
  - Distributed DBs rarely scale over ~50 nodes.

- **Limited flexibility and high administration costs**
  - Many design decisions have are manually performed.
  - Adding and removing nodes is tedious.
  - Schemas are usually fixed or hard to change.

- **High hardware costs**
  - High-performance DDBMS require expensive specialized and reliable hardware.
• Can data be distributed with lower costs, higher flexibility, higher capacity, and higher performance?
  – Yes… however, sacrifices have to be made and/or new system design paradigms are necessary
• **Additional nice-to-have features**
  – **Unlimited Linear Scaling**
    • **Performance** and **storage capacity** scales linearly with the number of machines without any limit
  – **Flexible Schemas**
    • Data schemas can be defined partially and may freely evolve or change
    • Different schemas for tuples of the same entity class
    • **Loosen up the relational model!**
  – **Elastic Computing**
    • Machines can be added or removed freely without any configuration and time overhead (“Machine Plug & Play”)
Newer approaches:

- **P2P systems**
  - Store all data *decentrally* on users’ computers
  - Most DB features have to be sacrificed
  - Very *low costs* for data provider
  - Very high *scalability* and *flexibility*

- **Cloud systems**
  - Store data *centrally* on a large number of *low-cost systems*
  - Try to keep as many DB features as possible / necessary
    - However, sacrifices have to be made
  - Aim for high degrees of *elasticity* and *flexibility*
• Hardware costs of a DDBMS
  – Usually run by big companies with dedicated data centers
  – DDBMS usually resides on extremely expensive blade servers
    • DELL PowerEdge M910 (Oct 2011)
      – 4x XEON E7-8837, 2.67 GHz, 8 Cores each
      – 384 GB RAM
      – 3.0 TB RAID HD
      – 38.000 €
    • Building a data center with such Blades is very expensive… (1 Rack, 32 Blades)
      – ~1.2 Million € for 512 cores, 12 TB RAM, 96 TB HD
      – Additional costs for support, housing, etc…
  – Analogy: data lives in high class condos
• Hardware costs of a Cloud / P2P system
  – Software usually resides on very cheap **low-end hardware**
  • **DELL Vostro D 460 (Oct 2011)**
    – Intel Core i7-2600 3,4 GHz, 8 Cores
    – 16 GB RAM
    – 2 TB HD
    – 1000 €

  • Performance comes **cheap** (1,200 machines)
    – ~ 1.2 Million € for 9600 cores, 19,2 TB RAM, 2,4 PB HD
    – Blade: ~1.2 Million € for 512 cores, 12 TB RAM, 96 TB HD

  – Analogy: data lives in the **slums**

Distributed Data Management – Christoph Lofi – IfIS – TU Braunschweig
1.2 Peer-To-Peer Systems

• Peer To Peer (P2P) Systems
  – P2P systems have been popularized in 1999 by Napster for sharing MP3’s
  – Base Problem: How can resources easily be shared within a highly volatile and decentralized network of independent peers (nodes)?
    • There is an (potentially) large number of peers
    • Peers may join or leave the network any time
    • Only rudimentary features necessary
1.2 Peer-To-Peer Systems

What is a P2P network?

- A virtual overlay network for sharing resources
  - Virtual and physical network are logically independent
  - Mostly IP based
- Decentralized and self-organizing
- Peers can transfer data directly without intermediate servers

Enabling technologies

- Performance increase of home user’s personal computers
- General availability of high-speed internet

Major Challenges

- Discovering resources
- Organizing the network
- Transfer data
1.2 Peer-To-Peer Systems

• Development of the terminal capabilities
  
  – 1992
    • Average hard disk size: \( \sim 0.3 \) Gbyte
    • Average clock frequency of personal computers: 100 MHz
  
  – 2002
    • Average hard disk size: 100 Gbyte
    • Personal computers have capabilities comparable to high-end servers in the late 1980s
  
  – 2007
    • Average clock frequency of personal computers: \( \sim 3 \)GHz
    • Average hard disk size: 320 Gbyte

Distributed Data Management – Christoph Lofi – IfS – TU Braunschweig
1.2 Peer-To-Peer Systems

• Development of internet connectivity
  – Early 1990s
    • Private users start to connect to the Internet via 56kbps modems
    • First broadband connections for residential users become available
    • Cable modem with up to 10Mbps
  – 1999
    • Introduction of DSL and ADSL connection
    • Data rates of up to 8.5Mbps via common telephone connections become available
    • The deregulation of the telephone market shows first effects with significantly reduced tariffs, due to increased competition on the last mile
  – 2000+
    • Bandwidth is plentiful and cheap!
1.2 Peer-To-Peer Systems

• What can be shared?
  – **Information**
    • File & document sharing
  – **Bandwidth**
    • Load balancing
    • Shared bandwidth
  – **Storage space**
    • DAS, NAS, SAN
    • Storage networks
  – **Computing Power**
    • High Performance Computing
1.2 P2P Applications

• **File sharing**
  – Classical application of P2P systems
  – Users offer files (music, videos, etc.) for free download
  – The application provides a unified view
  – Napster, Gnutella & Co

• **First large scale occurrence of digital copyright infringement**
  – Strong reactions from industry, e.g. Recording Industry Association of America (RIAA)
1.2 P2P Applications

• **Distribution of Software/Updates**
  – Basic idea of distributing software updates or patches in a P2P fashion
  – Used for a wide variety of various software distributions

• **Prominent examples**
  – Patches for the game „World of Warcraft“ by Blizzard Entertainment
  – Several Linux distributions
  – VMware images

• **Today, mostly BitTorrent is used**
  – Block-based File Swarming
1.2 P2P Applications

• **Document Management**
  – Usually centrally organized
    • But large portion of the documents created in a company are distributed among desktop PCs
    • Central storage is cumbersome for most users

• **Solution**
  – P2P networks which create a connected repository of local data on the individual peers
    • Optionally, use centralized server for support
  – Indexing and categorization of data by each peer on the basis of individually selected criteria
  – Self organized aggregation of information from areas of knowledge
1.2 P2P Applications

- **Routing & Bridging**
  - Use peers to route traffic to avoid central bottlenecks
  - Possibly, unreachable nodes can connected by bridge nodes
    - e.g. to cross firewalls
  - Sample application: Skype

- **Peer-to-Peer Unicast:**
  - Initial requests for files have to be served by a central server
  - Further requests can be automatically forwarded to peers within the network, who have already received and replicated these files
1.2 P2P Development

• The “hot” years for P2P had been 1999-2008
• In 2006, nearly 70% of all network traffic was attributed to P2P traffic
  – Nowadays, P2P traffic declines in favor of video streaming and social networks...

Survey by Cisco’2010
1.2 P2P Impacts

- Which protocols are used?
  - Traffic measured between 2002 and 2004 in Abilene backbone
1.2 P2P Development

• What was transferred?
1.3 Cloud Storage

• **Cloud storage** gained momentum with the advent of **web age applications**
  – Most notable pioneers are Google, Amazon, and Yahoo

• **The main problem:**
  – “Traditional” storage solutions could not keep up with the high demand wrt. to **throughput**, **latency**, and **storage space**
    • Not enough performance & storage space
    • Too expensive
    • Too inflexible
    • Many features not necessary
1.3 Cloud Storage

• Solution:
  – **Combine** ideas from **P2P** with techniques from **DDBMS** to create highly scalable data centers

• Optimized for **cheap large-scale data centers**
  – Focus on **low-end hardware**
  – High hardware **elasticity**
  – **Near-linear** performance and storage **scaling**
  – **Flexible schemas**
  – **High fault tolerance**
  – **Unified service interfaces**
Web Age applications may grow extremely fast

- Users, page views, and data
- Example: Facebook now has more than 800 M active users

  - Severe challenges to data backend
  - Optimization for read access
    - #read \gg #writes
1.3 Web Age Apps

- Facebook
  - More than **400 million active users**
  - **50%** of our active users log on to Facebook in any given day
  - More than **35 million** users update their status each day
  - More than **60 million status updates** posted each day
  - More than **3 billion photos** uploaded to the site each month
  - More than **5 billion pieces** of content (web links, news stories, blog posts, notes, photo albums, etc.) shared each week
  - More than **3.5 million events** created each month
  - More than **3 million active Pages** on Facebook
  - More than **1.5 million local businesses** have active Pages on Facebook
  - More than **20 million people** become fans of Pages each day
  - Pages have created more than **5.3 billion fans**
1.3 Web Age Apps

– Example: Farmville has more than 150M users
  • March 2010
  • Ratio reads : writes = 3:2

Daily active users!
1.3 Google Servers

• … or how to build one of the most powerful data centers out of crappy hardware
  – Google has jealously guarded the design of its data centers for a long time
    • In 2007 & 2009 some details have been revealed

• The Google Servers
  – Google only uses custom build servers
  – Google is the world 4th largest server producer
    • They don’t even sell servers…
    • In 2007, it was estimated that Google operates over 1,000,000 servers over 34 major and many more minor data centers
1.3 Google Servers

– Data centers are connected to each other and major internet hubs via massive fiber lines (2010)
  • ~7% of all internet traffic is generated by Google
  • ~60% of that traffic connects directly to consumer networks without connecting to global backbone
    – If Google was an ISP, it would be the 3rd largest global carrier
1.3 Google Servers

• Some Google Datacenter facts & rumors
  – In 2007, four new data centers were constructed for 600 million dollars
  – Annual operation costs in 2007 are reported to be 2.4 billion dollars
  – An average data center uses 50 megawatts of electricity
    • The largest center in Oregon has an estimated use of over 110 megawatts
    • The whole region of Braunschweig is estimated to use up roughly 225 megawatts
Each server rack holds 40 to 80 commodity-class x86 PC servers with custom Linux (2010)
- Servers run outdated hardware
- Each system has its own 12V battery to counter unstable power supplies
- No cases used, racks are setup in standard shipping containers and are just wired together

More info: http://www.youtube.com/watch?v=Ho1GEyftpmQ
1.3 Google Servers

• Google servers are very **unstable**
  – … but also very cheap
  – High “bang-for-buck” ratio

• Typical **first year for a new cluster** (several racks):
  – ~0.5 **overheating** (power down most machines in <5 mins, ~1-2 days to recover)
  – ~1 **PDU** (power distribution unit) **failure** (~500-1000 machines suddenly disappear, ~6 hours to come back)
  – ~1 **rack-move** (plenty of warning, ~500-1000 machines powered down, ~6 hours)
  – ~1 **network rewiring** (rolling ~5% of machines down over 2-day span)
1.3 Google Servers

- ~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
- ~5 racks go wonky (40-80 machines see 50% packet loss)
- ~8 network maintenances (might cause ~30-minute random connectivity losses)
- ~12 router reloads (takes out DNS and external VIPs for a couple minutes)
- ~3 router failures (traffic immediately pulled for an hour)
- ~dozens of minor 30-second DNS blips
- ~1000 individual machine failures
- ~thousands of hard drive failures
- Countless slow disks, bad memory, misconfigured machines, flaky machines, etc.
• Challenges to the data center software
  – Deal with all these hardware failures while avoiding any data loss and ~100% global uptime
  – Decrease maintenance costs to minimum
  – Allow flexible extension of data centers
  – Solution:
    • Use cloud technologies
    • GFS (Google File System) and Google Big Table Data System
      – Now, replaced by Spanner
    • To be discussed in a couple of weeks
We will discuss three major distributed data systems

- **Distributed Databases**
  - “Classic” approach to data distribution
  - Provides full DB feature sets
    - Transactions, query languages, schemas, etc.
  - Limited flexibility, elasticity, and scalability

- **P2P Systems**
  - Overlay network
  - Data is stored decentrally on users’ PCs
  - Very limited feature sets in exchange for very high scalability and flexibility
Summary

– **Cloud Storage & NoSQL Systems**
  - Fusion of DDBMS and P2P technologies
  - Tailored for data center usage
  - **Main goals:** **Cheap large-scale data centers**
    - Focus on low-end hardware
    - High hardware elasticity
    - Near-linear performance and storage scaling
    - Flexible schemas
    - High fault tolerance
Summary

DB features (data model complexity, query power, storage guarantees, transactional support)

Network & hardware features (elasticity, extensibility, flexibility, volatility)

- Distributed DB
- P2P
- Cloud
Outlook

• Next Lecture
  – Data Partitioning
  – Sharding
    • “Share-Nothing-Architectures”