Verteilte Informationssysteme

Mitschrift von Aaron Winziers
Wintersemester 2019/20

1 Organisation

Prüfungstermine:

- 19.02.2020
- 01.04.2020

2 Introduction

2.1 Lifecycle of an information system

- Small company(startup) - Bookkeeping etc still possible on paper
- Growth - Information and data becomes too big for paper - database is needed
- Continued growth - Need for more databases or even specialized systems, more locations need more databases

2.2 What to do?

- Data warehouse
- Distributed architecture
- Replication - Which locations need which data?
- Cloud computing

There are 2 important use cases that should be considered: transactional load(OLTP)(write heavy) and analytical load(OLAP)(read heavy)

2.3 SQL vs NoSQL

2.3.1 SQL:

- (+) declarative queries
- (+) consistency
- (+) guarantees
- (+) data independence

- (+) normalization
- (-) do not scale well with increasing load
- (-) features not always needed/are not used
- (-) rigid data structures

2.3.2 NoSQL:

- ( ) not relational
- (+) scale well, are distributes in nature (more nodes = more performance)
- ( ) often w/o query language but with simple API
- (-) weak consistency models - distributed copies may not be identical
- (+) offer high performance

Both systems are typically combined into hybrid systems

2.4 Reasons for distributing data

- Cost and scalability - mainframes are difficult to extend
- Replications leads to higher availability
- Integration of different software modiles - prevent collisions
- Integration of legacy systems - old system can continue to exists parallel to new system
- New kinds of applications - esp. e-commerce
- Market forces

2.5 Why distribution?

Distributed data better corresponds to modern enterprise structures

3 Distributed query processing

3.1 Important aspects

More replications faster queries but slower updates

Fragmentation storing local data locally

Parallelism multiple queries can be performed at once, or a single query can be split into parts and executed at the same time

Transparency fragmentation, replication etc should not be need to be taken into account by the user

3.2 Systems differ in terms of:

- Degree of coupling
- Interconnection structure
- Interdependence of components
- Synchronization of components

3.3 Forms of distributed systems

- Peer-to-Peer and file sharing
- Cloud computing
- Web services and the deep Web
- Semantic Web
- Big Data Analytics

3.4 Fallacies of distributed computing:

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogenous
- Location is irrelevant

3.5 Promises of distributed database systems

3.5.1 Transparent data management

- Systems hide implementation details

Data independence

- Immunity of applications to changes in the definition and organization of data, and vice versa
- Logical data independence (changes in the schema definition) Application should still be running if additional attributes are added to a relation
- Physical data independence (changes to physical data organization) Hiding the physical data organization (relations, indexes)

Network Transparency

Replication Transparency

Fragmentation Transparency

Who should provide transparency?
Application - are implemented in a distributed fashion - communicate with standard protocols
Operating system - provides network transparency on file system or protocol level
Database system - Transparent access to data at remote database instances - Requires splitting queries, transaction control, replication

3.5.2 Reliability

- Compensating node failures through data copies
- Distributed transactions guarantee that:
  - A sequence of operations is performed as an atomic action
  - A consistent database state transforms into another consistent database state, even with multiple transactions occuring concurrently
- Increased effort for updates, system may crash with server failure

3.5.3 Improved performance

- Fragmenting data in a way that enables data to be stored in close proximity to its points of use
- Distributed systems inherently have parallelism
  - Inter-query - execution of multiple queries
  - Intra-query - Parallel execution of sub-queries
- Read-only vs Update access
  - Query database (ad-hoc querying) and production database (for updates by application programs) - production database is copied into the query database in regular intervals
  - Read-only access during regular hours, updates are batched and performed during off-hours

3.5.4 Easier system expansion

- Ability to expand database size and/or decrease query time is a necessity
- Done by adding additional storage and processing power to the network
- A system of smaller computers is often cheaper than a larger single computer with equivalent power

3.6 Challenges

- Distributed database design - fragmentation, replication, distribution
- Distributed query processing - maximizing cost-effectiveness of executing queries
- Distributed concurrency - Synchronizing access for integrity
- Reliability - Ensure consistency, detect failures, recover from failures
- Heterogeneous DBS - Translation between DBS - data model and data language

3.7 Standard architectures

- Distributed information system - Applications communicate for data exchange
- DBMS - OS hides distribution
- Distributed DBS - Distribution handled by DBS
- Parallel DBS
  - Data processing by simultaneous computers(multi-processor, special hardware)
  - Increase performance by using multiple processing units

3.8 Relationale Algebra

- Union eliminates duplicates - SQL doesn't

4 Fragmentation and allocation in distributed database management systems

5 Replication and synchronization

6 Grid and cloud computing

7 Distributed transactions

8 Information integration

9 Distributed information retrieval

10 Parallel database systems