Distributed Database Systems

Part of the materials are from M. Ozsu and P. Valduriez. Principles of Distributed Database Systems, Prentice Hall, (2ed)

What is a Distributed Database System (DDBS)?

A distributed database is a collection of multiple, logically interrelated databases distributed over different computers of a computer network.

- Each site has autonomous processing capability and can perform local applications.
- Each site also participates in the execution of at least one global application which requires accessing data at several sites.
What we miss here is the existence of local applications, in the sense that the integration of the system has reached the point where no one of the computers (i.e., IFPs & ACPs) is capable of executing an application by itself.

3 Basic Hardware Architectures for Parallel DBMS
Hybrid Architecture for Parallel DBMS

Examples

• Shared Everything (SE)
  - HP T500
  - SGI Challenge
  - Pentium-based SMP

• Shared Disk (SD)
  - Intel Paragon
  - nCUBE/2
  - Tandem’s ServerNet-based machine

• Shared Nothing (SN)
  - Teradata’s DBC
  - Tandem NonStopSQL
  - IBM 6000 SP
Network Transparency

- The user should be protected from the operational details of the network.
- It is desirable to hide even the existence of the network, if possible.
  - Location transparency: The command used is independent of the system on which the data is stored and the system on which the command is executed.
  - Naming transparency: a unique name is provided for each object in the database. The name does not have object location associated with it.

Replication & Fragmentation Transparency

- The user is unaware of the replication of fragments.
- Queries are specified on the relations (rather than the fragments).
Disadvantages of DDBSs

Cost: replication of effort (manpower).

Security: More difficult to control

Complexity:
- The possible duplication is mainly due to reliability and efficiency considerations. Data redundancy, however, complicates update operations.
- If some sites fail while an update is being executed, the system must make sure that the effects will be reflected on the data residing at the failing sites as soon as the system can recover from the failure.
- The synchronization of transactions on multiple sites is considerably harder than for a centralized system.

Why Distributed Databases?

1. Local Autonomy: permits setting and enforcing local policies regarding the use of local data (suitable for organization that are inherently decentralized).

2. Improved Performance: The regularly used data is proximate to the users and given the parallelism inherent in distributed systems.

3. Improved Reliability/Availability:
   - Data replication can be used to obtain higher reliability and availability.
   - The autonomous processing capability of the different sites ensures a graceful degradation property.
• Incremental Growth: supports a smooth incremental growth with a minimum degree of impact on the already existing sites.

• Shareability: allows preexisting sites to share data.

• Reduced Communication Overhead: The fact that many applications are local clearly reduces the communication overhead with respect to centralized databases.

Distributed DBMS Architecture
**ANSI/SPARC Architecture**

- **External Schema**: deals with the physical definition and organization of data.
- **Conceptual Schema**: abstract definition of the database. It is the "real world" view of the enterprise being modeled in the database.
- **Internal Schema**: individual user's view of the database.

**A Taxonomy of Distributed Data Systems**

A distributed database can be defined as a logically integrated collection of shared data which is physically distributed across the nodes of a computer network.

- **Distributed data systems**
  - **Distributed DBMS** (A0, D2, H0)
  - **Federated DBMS** (A1, Dx, Hy)
  - **Multi-DBMS** (A2, Dx, Hy)
- **Loosely coupled** (interoperable DB systems using export schema)
- **Tightly coupled** (w global schema)

For distributed DBMS, global schema refers to the union of all the local databases.
For multi-DBMS, global schema refers to a subset of the union of all the local databases.
• Autonomy (distribution of control)
  - A0, tight integration
    - Single image of the entire database
    - There exists a coordinator who controls the processing of data
    - Each site do not operate independently although it is capable
  - A1, semi-autonomous
    - DBMSs operate independently but decide to collaborate to share local data
    - DBMS needs to be modified to be able to exchange shared local information
  - 2, isolation
    - DBMSs operate independently
    - No modification to DBMSs
    - Another layer of software is placed on top of these DBMSs to allow sharing of data

• Distribution (physical distribution of data)
  - D0 → no distribution
  - D1 → client/server (server manages the data, but client focuses on user interface and some computation)
  - D2 → Full distribution
    - Each node is equal → peer
• Heterogeneity
  - Focus more on heterogeneity in terms of data models, query languages, transaction management protocols
  - H0→homogeneity
  - H1→heterogeneity

A Taxonomy of Distributed Data Systems

For distributed DBMS, global schema refers to the union of all the local databases
For multi-DBMS, global schema refers to a subset of the union of all the local databases
Distributed DBMS (A0, D2, H0)

This type of DDBMS resembles a centralized DB, but instead of storing all the data at one site, the data is distributed across a number of sites in a network.

Fragmentation Schema & Allocation Schema

Fragmentation Schema: describes how the global relations are divided among the local DBs.

Allocation Schema: specifies at which sites each fragment is stored.

Example: Fragmentation of global relation R.

To materialize R, the following operations are required:
\[ R = (A \bowtie B) U (C \bowtie D) U E \]
Query Decomposition:
- Normalization
- Semantically analyze the normalized query to eliminate incorrect queries.
- Simplify the correct query by removing redundant predicates.
- Restructure the algebraic query into a "better" algebraic specification.

This step is the same as standalone DBMS.

Schema Architecture of a Tightly-Coupled System

An individual node's participation in the MDB is defined by means of a participation schema.
Auxiliary Schema (1)

Auxiliary schema describes the rules which govern the mappings between the local and global levels.

- Rules for unit conversion: may be required when one site expresses distance in kilometers and another in miles, ...
- Rules for handling null values: may be necessary where one site stores additional information which is not stored at another site.
  - Example: One site stores the name, home address and telephone number of its employees, whereas another just stores names and addresses.

Auxiliary Schema (2)

- Rules for naming conflicts: naming conflicts occur when:
  - semantically identical data items are named differently
    - DNAME → Department name (at Site 1)
    - DEPTNAME → Department name (at Site 2)
  - semantically different data items are named identically.
    - NAME → Department name (at Site 1)
    - NAME → Manager name (at Site 2)
- Rules for handling data representation conflicts: Such conflicts occur when semantically identical data items are represented differently in different data source.
  - Example: Data represented as a character string in one database may be represented as a real number in the other database.
Auxiliary Schema (3)

- Rules for handling data scaling conflicts: Such conflicts occur when semantically identical data items stored in different databases using different units of measure.
  - Example: “Large”, “New”, “Good”, etc.

These problems are called domain mismatch problems

Loosely-Coupled Systems
(Interoperable Database Systems)

Each site share all local information.
Loosely-Coupled Systems

Global user view 1
Global user view 2
Global user view m

Export schema 1
Export schema 2
Export schema 3
Export Schema n

Local user view 1
Local user view 2

Local Conceptual schema 1
Local Conceptual Schema 2
Local Conceptual Schema n

Local internal schema 1
Local internal Schema 2
Local internal Schema n

Local DB 1
Local DB 2
Local DB n

Global user views are constructed using powerful query language such as MySQL.

Export schema describes data a database site is willing to share.

Tightly-coupled vs loosely-coupled systems.

• Tightly coupled system has a global schema.
  • Translation is between local and global schema.
  • More work on the database administrator.
  • Less work on the user: query is written based on a global schema.

• Loosely-coupled system does not have a global schema.
  • Translation is between external schemas and local conceptual schemas.
  • More work on users.
Integration of Heterogeneous Data Models

- Adopt a single model (called canonical model) at the global level and map all the local models onto this model
  - Advantage: requires only $2n$ translators
  - Disadvantage: translations must go through the global model.

- Provide bidirectional translators between all pairs of models
  - Advantage: no need to learn another data model and language
  - Disadvantage: requires $n(n-1)$ translators, where $n$ is the number of different models.

(The 1st approach is more widely used)