Distributed Database Systems

By
Syed Bakhtawar Shah Abid
Lecturer in Computer Science
Application architectures

- The architecture of a system defines its structure.
- This means that the components of the system are identified, the function of each component is specified, and the interrelationships and interactions among these components are defined.
Application architectures

- The specification of the architecture of a system requires
  - Identification of the various modules,
  - Their interfaces and interrelationships, in terms of the data and control flow through the system.
There are three “reference” architectures for a distributed DBMS:
- Client - Server systems,
- Peer-to-Peer distributed DBMS and
- Multi-Database Systems (MDBS).
Client-server architecture

server:
- always-on
- fixed/known IP address
- serves many clients at the same time

clients:
- communicate with server only
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

E.g., of client-server archit.:
- Google, Amazon, YouTube etc
Client-server architecture

- This provides *two-level architecture* which make it easier to manage the complexity of modern DBMSs and the complexity of distribution.
- In relational systems, the server does most of the data management work.
- All of
  - Query processing
  - Query optimization,
  - Transaction management and
  - Storage management is done at the server.
Client-server architecture

- The client has
  - The application installed
  - The user interface
  - DBMS client module is responsible for managing the data that is cached to the client and
  - DBMS client module is responsible for managing the transaction locks.
Client/Server Reference Architecture
Client/Server Reference Architecture

- The **user interface handler** is responsible for interpreting user commands as they come in, and formatting the result data as it is sent to the user.

- The **semantic data controller** uses the integrity constraints and authorizations that are defined as part of the global conceptual schema to check if the user query can be processed.
Client/Server Reference Architecture

- The **global query optimizer** and decomposer determines an execution strategy to minimize a cost function, and translates the global queries into local ones using the global and local conceptual schemas.
- The global query optimizer is responsible, among other things, for generating
- the best strategy to execute distributed join operations.
The distributed execution monitor coordinates the distributed execution of the user request.

The execution monitor is also called the distributed transaction manager.

In executing queries in a distributed fashion, the execution monitors at various sites may, and usually do, communicate with one another.
The **local query optimizer**, which actually acts as the access path selector, is responsible for choosing the best access path to access any data item.

The **local recovery manager** is responsible for making sure that the local database remains consistent even when failures occur.
The run-time support processor physically accesses the database according to the physical commands in the schedule generated by the query optimizer.

The run-time support processor is the interface to the operating system and contains the database buffer (or cache) manager, which is responsible for maintaining the main memory buffers and managing the data accesses.
Peer to Peer architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- example: BitTorrent

Pros and cons:
- scalable
- difficult to manage
- not secure
The physical data organization on each machine may be different.

*Local internal scheme (LIS)* - is an individual internal schema definition at each site.

*Global conceptual schema (GCS)* - describes the enterprise view of the data.

*Local conceptual schema (LCS)* - describes the logical organization of data at each site.

*External schemas (ESs)* - support user applications and user access to the database.
DISTRIBUTED DBMS ARCHITECTURE
PEER-TO-PEER DISTRIBUTED SYSTEMS

ES_1 → LCS_1 → LIS_1
ES_2 → LCS_2 → LIS_2
... → ... → ...
ES_n → LCS_n → LIS_n

Distributed Database Reference Architecture
The detailed components of a distributed DBMS.

Two major components:
- user processor
- data processor
User processor

- **User interface handler** - is responsible for interpreting user commands as they come in, and formatting the result data as it is sent to the user,

- **Semantic data controller** - uses the integrity constraints and authorizations that are defined as part of the global conceptual schema to check if the user query can be processed,

- **Global query optimizer and decomposer** - determines an execution strategy to minimize a cost function, and translates the global queries in local ones using the global and local conceptual schemas

- **Distributed execution monitor** - coordinates the distributed execution of the user request.
Data processor

- *Local query optimizer* - is responsible for choosing the best access path to access any data item,
- *Local recovery manager* - is responsible for making sure that the local database remains consistent even when failures occur,
- *Run-time support processor* - physically accesses the database according to the physical commands in the schedule generated by the query optimizer. This is the interface to the operating system and contains the *database buffer (or cache) manager*, which is responsible for maintaining the main memory buffers and managing the data accesses.
MDBS Architecture

- Multi-Database System (MDBS)
  - An integrated Database system composed of a collection of two or more autonomous databases / datasets
  - Component DBs can be
    - Traditional Databases (relational, network, hierarchical)
    - Documents
    - Spatial (Map, Images)
    - Statistical
    - Objects and/or
    - File systems
Components of an MDBS
## MDBS Architecture

- **Models using a Global Conceptual Schema (GCS)**
  - The GCS is defined by integrating either the external schemas of local autonomous databases or parts of their local conceptual schemas.
  - If the heterogeneity exists in the system, then two implementation alternatives exist:
    - Unilingual MDBS
    - Multilingual MDBS
MDBS Architecture

- **Unilingual**
  - Each DB only speaks one language, so users/applications must query DBs in their native data representation, i.e. each DB speaks one so users speak many
  - A unilingual multi-DBMS requires the users to utilize possibly different data models and languages when both a local database and the global database are accessed.
  - Any application that accesses data from multiple databases must do so by means of an external view that is defined on the global conceptual schema.
  - One application may have a local external schema (LES) defined on the local conceptual schema as well as a global external schema (GES) defined on the global conceptual schema.
MDBS Architecture

- Multilingual
  - All queries in the local DB language – runtime translation of queries into the data representation of the GCS.
  - An alternative is multilingual architecture, where the basic philosophy is to permit each user to access the global database by means of an external schema, defined using the language of the user’s local DBMS.
  - The multilingual approach obviously makes querying the databases easier from the user’s perspective. However, it is more complicated because we must deal with translation of queries at run time.
MDBS Architecture

MDBS Architecture with a GCS
MDBS Architecture

- MDBS without a GCS
  - Each external schema joins some set of LCSs
  - **Advantages** – no need to replicate and keep GCS consistent across all machines that act as clients, fault isolation
  - **Disadvantages** – the GCS provides a global-uniform namespace, which is essential if computers wish to share data.
MDBS Architecture

MDBS Architecture Without a GCS
MDBS Architecture

- The architecture identifies two layers:
  - The local system layer and
  - The multi-database layer on top of it.

- The local system layer consists of a number of DBMSs, which present to the multi-database layer the part of their local database they are willing to share with users of the other databases. This shared data is presented either as the actual local conceptual schema or as a local external schema definition.

- The multi-database layer consist of a number of external views, which are constructed where each view may be defined on one local conceptual schema or on multiple conceptual schemas. Thus the responsibility of providing access to multiple databases is delegated to the mapping between the external schemas and the local conceptual schemas.
THE END