University of Science and Technology of Hanoi

Distributed System

Final Report

Distributed Database

Author:
Nguyen Duc Dan
Nguyen Huu Chi Dat
Nguyen Tri Huan
Nguyen Xuan Duy Anh
Pham Minh Giang

Lecturer:
Dr. Tran Giang Son

April 15, 2020

Contents

Abstract

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network. Distributed databases can be classified into homogeneous and heterogeneous databases having further divisions.There are two main distributed architectures namely client – server, peer – to – peer and multi – DBMS. Our project will be based on the client - server concept.

Acknowledgement

This report is prepared by IMT team from ICT Department Generation 8 of USTH - University of Science and Technology of Hanoi.

We wish to express our gratitude to number of people without whose constant guidance and encouragement this project would not have been possible. We express our sincere thanks to our project guiders, Dr. Tran Giang Son for his enthusiastic guidance and for providing us help as and when required.

The purpose of this project report is to create a distributed database system.

Finally, We wish to thank all of the IMT members, who work so hard to complete this project.

1 Introduction

2 Objective

Building a distributed database systems:

- Used to create, read, update and delete distributed databases.
- Guarantees that the information altered at any site is all around updated.
- Designed for diverse database platforms.
- Keeps up the secrecy and information honesty of the databases.
- Contained large volumes of data and accessed by numerous users.

3 State of the art

3.1 MongoDB

An open source database management system (DBMS) that uses a document-oriented database model which supports various forms of data. It is one of numerous nonrelational database technologies which arose in the mid-2000s under the NoSQL banner for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in relational databases, the MongoDB architecture is made up of collections and documents.

3.2 Oracle Database

A multi-mode database management system produced and marketed by Oracle Corporation. It is a database commonly used for running online transaction processing (OLTP), data warehousing (DW) and mixed (OLTP & DW) database workloads. The latest generation, Oracle Database 19c, is available on-prem, on-cloud, or in a hybrid-Cloud environment.

3.3 Microsoft SQL Server

It is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

4 Method

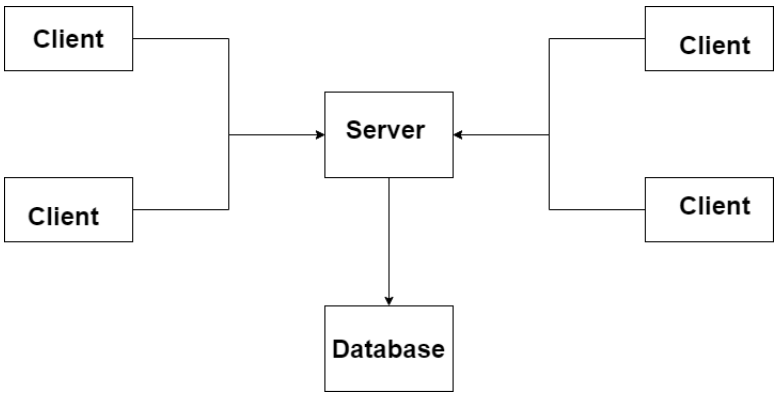
4.1 CRUD

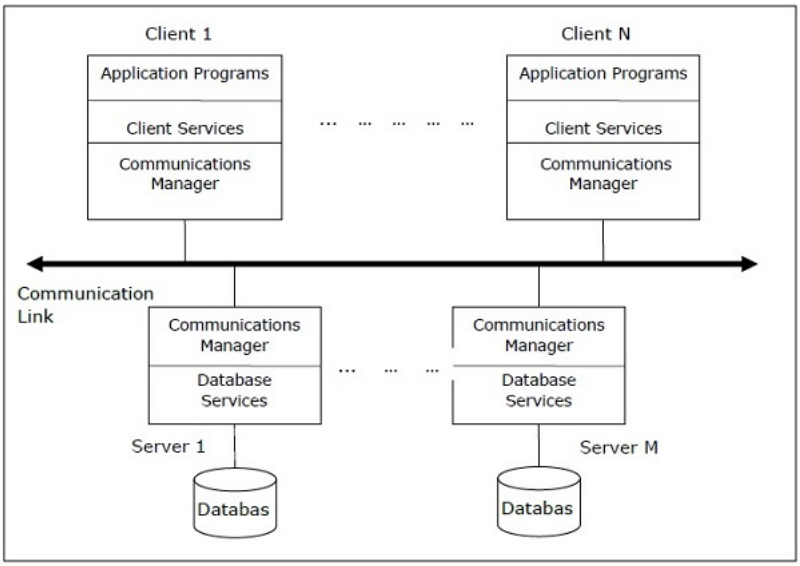
We want our models to provide four basic types of functionality. The model must be able to Create, Read, Update, and Delete resources.

- CREATE : This function calls when we want to add a new human and address to our database. After this function is called , there will be new humans and addresses are added to our database with a new ID.
- READ : Show the component in the database like name or address.
- UPDATE : If there are any changes from the client then new data will be updated to the database after this function is called new data will be provided.
- DELETE : Remove the data from the database.

4.2 Client-Server model

Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server.





4.3 Client-Server Component

4.3.1 Client

- The clients in this model are called clients. This is the place to send processing requests to the server.
- The client side organizes communication with the user, with the external environment at the workstation and with the server. After receiving the user's request, the client side sets up query strings sent to the server,

receives the results, and organizes them.

4.3.2 Server

- Our server is where the Database is stored and handles SQL transactions.
- The server processes and sends the results to the client. The client can continue to process these results for work.

The client initiates the connection and sends a request message, followed by processing the server's response message. Conversely, the server waits for a connection, processes the client's request message, and then sends a response message:

Step	Endpoint	Action/Message Content
1	Client	Sends a **Message** containing request content
2	Server	Receives and processes client request **Message**
3	Sever	Sends a **Message** containing response content
4	Client	Receives and processes server response **Message**

There are four basic types of action that our server can handle.How we implement our component:

Application	File	Code
Server	app-server.py	The server's main script
Server	libserver.py	The server's **Message** class
Client	app-server.py	The client's main script
Client	libserver.py	The client's **Message**

5 Evaluation

This is a basic example for distributed database so we don't have benchmark

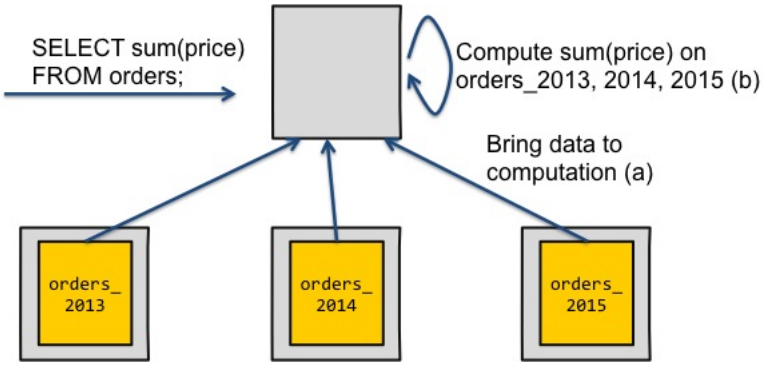
5.1 Response time

Varies depend on client nodes number:

- 0 to 20: 0 - 1s
- 20 to 50: 3s for any single nodes
- Above 50: system crashed

5.2 How can your system handle high load, stress?

We express any computation by pulling all data to a particular node, and then running the original query on this data. Of course, the drawback is that this method doesn't quite scale.For one, we are issuing a lot of network I/O during the data transfer, and network is our least scalable resource. For another, we aren't parallelizing any of the work, and instead running the entire computation on a single node.

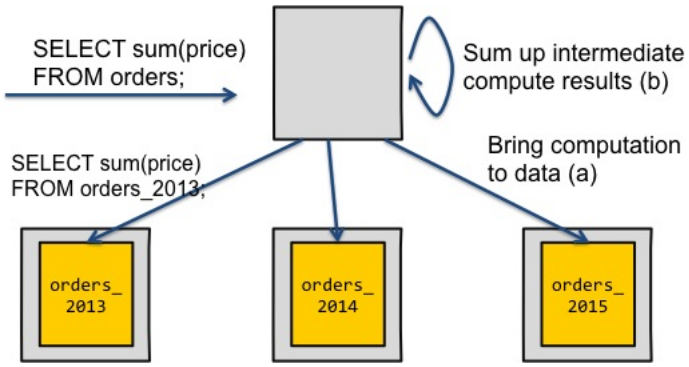


5.3 Scalability test

- We try to connect from 100 client nodes at a time, each node try to perform several SQL transactions. The system can not handle that many requests at a time and appear to crash.
- When we perform the test again with 50 client nodes at a time, our system react poorly and there is bottleneck effect appear.
- Our system perform best when handling 10 to 20 client nodes at a time.

5.4 Futher Improve

We could try to improve our model following this diagram. We're both parallelizing the computation here, and are also only transferring computation results over the network. We had to do a final sum on the coordinator node, but that was simple enough.



6 Conclusion

Our distributed database has four basic actions to handle and can simulate a basic distributed database. Our database on Client-Server which is very close to daily life. The Client-Server network model usually consists of one or more server computers that provide services and information to several workstation computers. On Internet clients, or computers with web browsers, access web sites that are hosted on servers. The performance for quite low but in the future, we can improve it.