A Focus On Testing Issues In A Distributed Database System

Mungamuru Nirmala
Research Scholar
Department of Comp., Sci., & Engineering
Senate Hall, University of Allahabad,
Allahabad, U.P, INDIA.

Dr. R. Mahammad Shafi
Professor & Head
Department of Master of Computer Applications
Sree Vidyanikethan Engineering College
Tirupati, Chittoor (Dist), A.P, INDIA

Abstract

Our previous research paper about Testing based software development in association with Extreme Programming (XP) led us to a conclusion that XP supports many good engineering practices but there is still place for refinements. A Distributed Database (DDB) is formed by a collection of multiple databases logically inter-related in a computer network. Any testing process, when used in DDB correlates a series of stages for the construction of a DDB project from the scratch and is employed in homogeneous systems. A sincere attempt is made in this paper to uncover the difficulties that often challenge the programmers in building DDB systems. Those difficulties are identified as openness, concurrency, scalability, fault tolerance, latency, global clock, security, and heterogeneity. In this paper, each issue is presented and is accompanied by the solutions.

Keywords: Distributed Database System, Openness, Latency, Global clock, Security, Heterogeneity.

1. Introduction

A distributed system is a collection of independent computers that are used jointly to perform a single task or to provide a single service. A Distributed Database Management System (DDBMS) is used as a system that enables the management of the individual database management system (DBMS). It distributes the data through a transparent way to the user. Figure-1 shows a common example of a DDB [7]. The simplest and most well known example of a distributed system is the collection of Web servers or more precisely, servers implementing the HTTP protocol—that jointly provide the distributed database of hypertext and multimedia documents commonly known as the World-Wide Web.

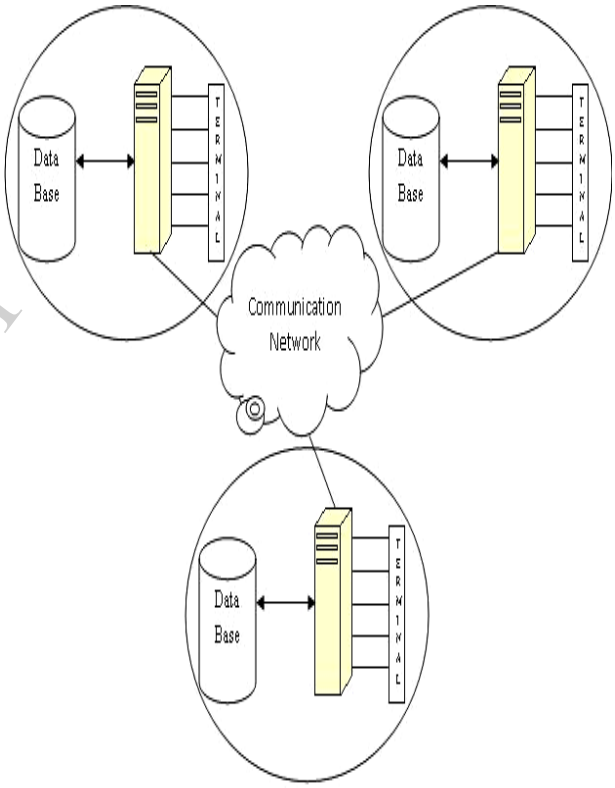

Fig-1: A Distributed Database on a Geographic dispersed Network

The transparences provided by a DDBMS can be understood as the high level semantic separation of the details inherent to the physical implementation of a DDB. The focus is to provide data independency in a distributed environment. This way, the user sees only one logically integrated image of the DDB as if it was not distributed. Figure-2 shows the logical view of the DDB that a user has, which was presented in figure-1.

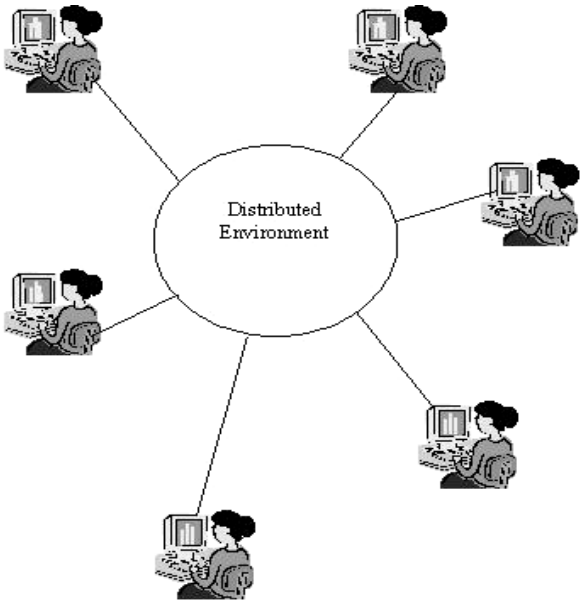

Fig-2: Logical view of a Distributed Database System

2. Openness

Openness is not only one of the important attributes that distinguishes DDB systems from other systems, but also the main objective that most developers would like to achieve. It is important to note that DDB systems tend to be extended and improved all the time. This can become a problem if the original system is not designed for this purpose and new components do not integrate with the existing components.

For example, when a new component is implemented, the system either fails to function or has to be reconstructed entirely to merge with the new component. Emmerich proposed that this problem is caused because of the differences in data representation of interface types on different processors [2]. To overcome this problem, Tanenbaum and Steen says that standard rules which describe the syntax and semantics of the services provided through interfaces have to be published [5]. These standard rules are formalized in an IDL, which includes the details of a particular function, that is, the syntax of services, and the features which these services provide, in other words, the semantics of interfaces. The aim of having detailed interfaces of components is to make sure that new components follow the standard rules of the original system [5].

Openness also includes the flexibility of a system. DDB system should be flexible enough to be extended in a straight forward manner. In addition, its architecture should also be capable of accommodating additional features easily as well as removing the redundant features. Standard and well-defined interfaces are essential in order to attain the flexibility.

3. Latency

In DDB systems, the time that takes to set up the communication is known as latency [3]. Apart from latency delay, there is transmission delay which is calculated by the length of the message and the data transfer rate, which is the time it takes to transfer the data of the message. In short, message transmission time is the total of latency delay and the transmission delay. Figure-3 shows the delay in sending a message.

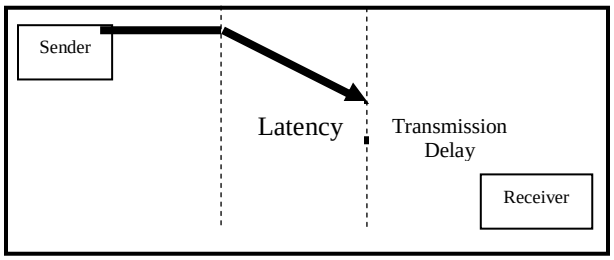

Fig-3: Message Transmission Time

The following are the various issues related to latency.

1. Latency indicates the physical limitations of communication between sender and the receiver. They can be propagation delay, insufficient bandwidth, network timeout, machine crashed, or connection refused.
2. Taylor, Redmiles, and Hoek point out that latency introduces uncertainty since the system fails to update the change to the clients [6].

The issue is identified when latency becomes high; this means a request has to wait for a long time before being executed. DDB system developers wish to minimize latency. According to Taylor et al. this can be achieved by using asynchronous notifications of resource changes and routing directly to relevant agencies [6].

4. Global Clock

Unlike centralized database systems which have a central server to control the system time, DDB systems do not have a global time. In a DDB system, although clocks are synchronized, each individual clock does not have the exact time because of being run at different rate. This may cause some difficulties in the context of process cooperation. For example, a request

that is sent after another request may be assigned an earlier time may lead to an incorrect and undesirable effect.

The time in DDB systems is only known within a given precision and a clock may be synchronized with a more trusted clock but it is impossible to run an absolute value for clocks due to transmission and execution [1]. This difficulty is addressed by using the Universal Time Coordinated (UTC) servers or the Internet Network Time Protocol (NTP). According to the NTP's documentation, NTP is a fault tolerant protocol which uses the UTC that allows the developers to synchronize computer clocks to national standard time via internet. NTP automatically selects the best of several available time sources to synchronize.

5. Security

Like any other systems, DDB systems require high level of security in order to prevent the occurrence of numerous threats [4]. The security threats that can affect the system can be categorized into four types as listed in the table-1.

Table-1: Security Threats

Interception	Threat appears when a service or data is accessed by an unauthorized party.
Interruption	Threat relates to a service or data that is made unavailable and inaccessible to other parties.
Modification	Threat pops up when an attempt is made to illegally modify the data so that it appears to be different from the original data.
Fabrication	Includes generating additional data or activity that would normally not exist.

As can be seen from these threats, security plays an important role in building and maintaining DDB systems. Tanenbaum and Steen identified that security in DDB system can be divided in two main parts viz., the secure communication channel and the authenticated access [5]. These are achieved by a number of security methods such as encryption, authentication, authorization, and auditing.

1. Firstly, encryption is the most common method that is used in computing security. Data is encrypted and is not easily understood by unauthorized users.
2. Secondly, authentication is a method to verify the identity of a user. Normally, each user is assigned by a username or a password, or both. There are also other ways to identify a user. Only an authenticated user is able to access the system.
3. Thirdly, authorization ensures that an authenticated user can only perform certain tasks depending on the permission that is given to a particular user. For example, an administrator of a forum can delete a post, but not a standard member. Finally, the auditing method is used to map out the tasks that each user performs while using the system. This information is stored in audit logs which can help to identify intruders.

6. Heterogeneity

When a distributed service is designed for one type of computer and one type of network, its portability is restricted to that computer and network.

This becomes one of the major difficulties and eliminates the efficient and effective utilization of resources (Turnbull, 1987). Thus, maintaining heterogeneity in DDB systems is necessary.

However, each computer in the network may have different memory sizes, network protocols, and I/O bandwidth. These are categorized into three main areas viz., computer hardware heterogeneity, network heterogeneity, and software heterogeneity as depicted in Table-2.

Table-2: Heterogeneity in DDB System

Hardware Heterogeneity	Nesterenko and Jin (2002) described the hardware heterogeneity as the difference in computer architectures of the components in DDB system such as instruction sets and data representations.
Network Heterogeneity	Network heterogeneity is the diversity of transmission media, signaling, protocols, and network interfaces.
Software Heterogeneity	Software heterogeneity relates to the difference in operating systems and application programs. These factors affect the DDB systems in terms of scalability, resource sharing, and openness. Nevertheless, support for heterogeneity remains a mostly unsolved problem (Nesterenko & Jin, 2002).

7. Conclusion

A distributed system is a collection of independent computers that appear to its users as a single coherent system. For establishing any testing process with a DDB, it is necessary to concentrate on Openness, Latency, Global Clock, Security and Heterogeneity of a Distributed Database.

Openness deals with improvement and extensions of a DDB system over time. Latency deals with latency delay and transmission delay which is calculated by the length of the message and the data transfer rate to transfer the data of the message. DDB systems require a high level of security in order to prevent a number of threats like interception, interruption, modification and fabrication. The Heterogeneity of a DDB system focus on probability of using a DDB system varies with different network protocols, memory sizes and bandwidths.

References:

[1]. Burback, R. L. R. (1998). *No Global Clock.*

[2]. Emmerich, W. (1997). *Distributed System Principles.* UK: University College London.

[3]. Hillston. J. (2002). *Distributed Systems* UK: The University of Edinburgh.

[4]. Pfleeger. S. (1997). *Software Engineering: Theory and Practice*. America: Prentice Hall.

[5]. Tanenbaum, A. S., & Steen, M. V. (2002). *Distributed Systems – Principles and Paradigms.* America: Prentice-Hall.

[6]. Taylor, R. N., Redmiles, D. F., & Hoek, A. V. D. (2006). *Decentralized Architectures*

[7]. R. Mahammad Shafi, Dr. B. Kavitha (2011) – *A Framework for Designing and Testing a Distributed Database System*, Proceedings of International Conference on Advances in Mathematical and Computational Methods.

[8]. G. Couloris, J. Dollimore, and T. Kinberg, Distributed Systems – Concepts and Design, 4th Edition, Addison-Wesley, Pearson Education, UK, 2001

[9]. Jim Waldo, Geoff Wyant, Ann Wollrath, and Sam Kendall, A note on distributed computing. Technical Report SMLI TR-94-29, Sun Microsystems Laboratories, Inc., 1994.

AUTHOR PROFILE

Dr. R. Mahammad Shafi received the Doctoral Degree in Computer Science and Engineering from University of Allahabad, Allahabad in 2010. Master's Degree in Computer Applications from University of Madras, Chennai in 1998 and M.Tech Degree in Software Engineering from Vinayaka Mission Research Foundation, Deemed University, Salem in the year 2005. Currently he is working as a Professor and Head, Department of MCA, Sree Vidyanikethan Engineering College, A. Rangampet, Near Tirupati, A.P., India. His areas of research interest include Software Engineering, Software Testing Methodologies and Distributed Databases.

Mrs. Nirmala Mungamuru received her M.Tech- Information Technology in 2004. Currently she is working as a Lecturer, Department of Computer Science, Eritrea Institute of Technology, Asmara, Eritrea, North East Africa. She has blended her wide experience of 13 years in teaching and research in the field of Computer Science and Information Technology. Her areas of research interest include Distributed Databases, Software Engineering and Software Testing Methodologies. She is a member of IAENG, CSI, IETE, ISTE and many journal membership bodies.