ISSN : 2231-0711

Chirra Priyanka et al | IJCSET(www.ijcset.net) | July 2015 | Vol 5, Issue 7,264-266

Distributed Real-Time Database System Model and Performance Metrics Based on QoS

Chirra Priyanka[1], K Praveen Kumar[2]

[1,2]*Department of Computer Science and Engineering,*
Chaithanya Institute of Technology and Science, warangal, INDIA

Abstract: In Distributed algorithm an algorithm for anonymous distributing of private facts and figures among parties is evolved. This method is utilised iteratively to accredit these nodes ID figures extending from 1 to N . This assignment is anonymous in that the identities received are unknown to the other constituents of the group. This allotment of successive numbers permits more complex facts and figures to be distributed and has submissions to other difficulties in privacy preserving data excavation, collision avoidance in communications and circulated database access. The needed computations are circulated without utilising a trusted central authority. The QOS such as end-to-end hold up, collision avoidance, Best-effort service and traffic shaping is achieved in circulated system.

Keywords: Quality of Service(Qos), Distributed system, Data sharing.

I. INTRODUCTION

Distributed System is a assemblage of autonomous computers linked by a mesh utilising programs to produce an integrated computing facility. The circulated scheme is decentralized one. The services of distributed scheme are

•Eposted letters-Electronic posted letters
•Netnews –group discussion on lone subject
•Multimedia teleconferencing over mesh
• WWW-World wide web

1.1 Service model of Distributed System

Computers can perform diverse purposes and each unit in a circulated scheme may be to blame for only a set number of purposes in an organization. We address the concept of service models as taxonomy of scheme configurations.

1.2 Centralized model of Distributed system

A centralized form is one in which there is no networking. All facets of the submission are hosted on one appliance and users exactly attach to that appliance. This is epitomized by the classic mainframe time-sharing system. The computer may comprise one or more CPUs and users communicate with it by terminals that have a direct (e.g., serial) connection to it.The main difficulty with the centralized form is that it is not effortlessly scalable. There is a limit to the number of CPUs in a scheme and eventually the entire scheme desires to be upgraded or restored. A centralized scheme has a problem of multiple entities arguing for the identical asset.

1.3 Client-server model

The client-servermodel is a popular networked model consisting of three components. A serviceis the task that a particular machine can perform. For example, offering files over a network, the ability to execute certain commands, or routing data to a printer. A serveris the machine that performs the task (the machine that hosts the service). A machine that is primarily recognized for the service it provides is often referred to as a print server, file server, et al. The clientis a machine that is requesting the service. The labels clientand serverare within the context of a particular service; a client can also be a server. A particular case of the client-server model is the workstation model, where clients are generally computers that are used by one user at a time (e.g. a PC on a network).

1.4 Focus of Resource Sharing

Users are so used to the advantages of asset sharing that They may effortlessly overlook their significance. We regularly share hardware assets such as printers, facts and figures assets such as files, and resources with more exact functionality such as seek engines. Looked at from the point of view of hardware provision, we share equipment such as printers and computer disks to reduce charges. But of far larger implication to users is the distributing of the higher-level assets that play a part in their submissions and in their everyday work and communal activities. For example, users are concerned with distributing data in the form of a distributed database or a set of web pages – not the disks and processors on which they are applied likewise, users believe in periods of shared resources such as a seek motor or a currency converter, without consider for the server or servers that supply these. In practice, patterns of asset distributing vary broadly in their scope and in how closely users work simultaneously. At one extreme, a seek engine on the world wide web presents a facility to users all through the world, users who need not ever come into contact with one another exactly. At the other farthest, in computer-supported cooperative working (CSCW), a assembly of users who help exactly share assets such as articles in a small, closed assembly. The pattern of distributing and the geographic circulation of particular user's works out what mechanisms the scheme must provide to coordinate users' actions.We use the period service for a distinct part of a computer scheme that organises a collection of associated resources and presents their functionality to users and submissions. For example, we get get access to to shared documents through a document service; we send articles to printers through a printing service; we purchase goods through an electrical devices payment service. The only get access to we have to the service is by the set of procedures that it trade goods.

For demonstration, a document service presents read, write and delete procedures on files.The detail that services constraint asset access to a well-defined set of procedures is in part standard software engineering perform. But it furthermore reflects the physical organization of circulated schemes. Assets in a circulated system are physically encapsulated inside computers and can only be accessed from other computers by means of communication. For productive sharing, each resource must be organised by a program that boasts a communication interface endowing the asset to be accessed and revised reliably and consistently. The period server's likely familiar to most readers. It mentions to a running program (a process) on a networked computer that accepts demands from programs running on other computers to present a service and answers appropriately. The requesting processes are mentioned to as clients, and the overall approach is renowned as client-server computing. In this approach, demands are sent in notes from purchasers to a server and answers are dispatched in notes from the server to the purchasers. When the client drives a demand for an operation to be carried out, we say that the purchaser invokes an operation upon the server. A entire interaction between a client and a server, from the issue when the client drives its request to when it receives the server's response, is called a remote invocation.The identical method may be both a purchaser and a server, since servers occasionallyinvoke procedures on other servers. The terms 'client' and 'server' request only to the roles performed in a single demand. purchasers are active (making requests) and servers are passive (only waking up when they receive requests); servers run relentlessly, while purchasers last only as long as the applications of which they pattern a part.Note that while by default the terms 'client' and 'server' refer to processes rather than the computers that they execute upon, in everyday parlance those periods also mention to the computers themselves.

II QUALITY OF SERVICE (QOS)

Quality of Service (QoS) refers to the capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET, and IP-routed networks that may use any or all of these underlying technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail. QoS technologies provide the elemental building blocks that will be used for future business applications in campus, WAN, and service provider networks.

Fundamentally, QoS enables you to provide better service to certain flows. This is done by either raising the priority of a flow or limiting the priority of another flow. When using congestion-management tools, you try to raise the priority of a flow by queuing and servicing queues in different ways. The queue management tool used for congestion avoidance raises priority by dropping lower-priority flows before higher-priority flows. Policing and shaping provide priority to a flow by limiting the throughput of other flows. Link efficiency tools limit large flows to show a preference for small flows.

A tool box, and many tools can accomplish the same result. A simple analogy comes from the need to tighten a bolt: You can tighten a bolt with pliers or with a wrench. Both are equally effective, but these are different tools. This is the same with QoS tools. You will find that results can be accomplished using different QoS tools. Which one to use depends on the traffic. You wouldn't pick a tool without knowing what you were trying to do, would you? If the job is to drive a nail, you do not bring a screwdriver.

QoS tools can help alleviate most congestion problems. However, many times there is just too much traffic for the bandwidth supplied. In such cases, QoS is merely a bandage. A simple analogy comes from pouring syrup into a bottle. Syrup can be poured from one container into another container at or below the size of the spout. If the amount poured is greater than the size of the spout, syrup is wasted. However, you can use a funnel to catch syrup pouring at a rate greater than the size of the spout. This allows you to pour more than what the spout can take, while still not wasting the syrup. However, consistent overpouring will eventually fill and overflow the funnel.

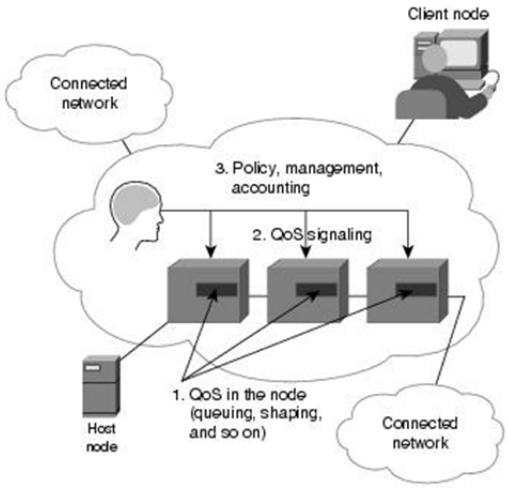

Fig.1.A basic QoS Architectutre

A. Basic QoS Architecture

The basic architecture introduces the three fundamental pieces for QoS implementation

QoS identification and marking techniques for coordinating QoS from end to end between network elements

QoS within a single network element (for example, queuing, scheduling, and traffic-shaping tools)

QoS policy, management, and accounting functions to control and administer end-to-end traffic across a network

III. REAL-TIME DATABASE MODEL AND PERFORMANCE METRICS

Before we present our QoS algorithm, we first introduce the distributed real-time database system model and the performance metrics considered in this paper.

A. Real-time Database Model

In this paper, we consider a distributed real-time database system which consists of a group of main memory real-time database systems connected by a Local Area Network(LAN). For the high performance of main memory accesses and the decreasing main memory cost, main memory databases have been increasingly used for data management real-time applications. We focus our study on medium scale distributed databases (in the range of 5 to 10 sites), since the load balancers need full information from every sites to make accurate decisions. Several applications that require distributed real-time data services fall in that range. For example, a ship-board control system which controls navigation and surveillance consists of 6 distributed control units and to general control consoles located throughout the platform and linked together via a ship-wide redundant Ethernet to share distributed real-time data and coordinate the activities. We leave it as the future work to make our solution applicable to large scale distributed real-time applications with 100s sites involved, using only a partial information from a subset of sites.

In this paper, we apply firm deadline semantics in which transactions add value to the application only if they finish within their deadlines. Hence, tardy transactions (transactions that have missed their deadlines) are aborted upon their deadline miss. Firm deadline semantics are common in several real-time database applications. A late commit of a real-time transaction may incur the loss of profit or control quality, resulting in wasted system resources, due to possible changes in the market or control status. Our objective is to provide QoS guarantees for real-time data services in those applications.

B. Data Composition

In our system model, data objects are divided into two types, namely, temporal data and nontemporal data. Temporal data are the sensor data from physical world. In ship-board control application, they could be ship maneuvering data such as position, speed and power; in stock trading, they could be real-time stock prices. Each temporal data object has a validity interval and is updated by periodic sensor update transactions. Non-temporal data objects do not have validity intervals and therefore there are no periodic system updates associated with them. Non-temporal data do not change dynamically with time.

In our distributed real-time database system model, a local site is called a node. Each node hosts a set of temporal data objects and non-temporal objects. The node is called the primary node for those data objects. Each node also maintains a set of replicas of temporal data objects hosted by other nodes. The fresh value of temporal data objects are periodically submitted from outside to their primary nodes and propagated to the replicas. In our replication model, temporal data objects are fully replicated and the replicas are updated as soon as the fresher data are available. Non-temporal data objects are not replicated because replicating non-temporal data objects will not improve the system performance when the read/write ratio is not high. For instance, replicating real-time stock quotes would be appropriate in stock trading, since a significant portion of user transactions only read the data.

C. Transaction Model

In our system, transactions are divided into two types, system update transactions and user transactions. System update transactions are temporal data (sensor data) update transactions and temporal data replica update transactions. User transactions are queries or updates from applications. User transactions are divided to different service classes, e.g., class 0, 1 and 2. The lower the service class number, the higher the priority the transaction has during the execution. Class 0 is the service class that has the best quality of service guarantee.

Transactions are represented as a sequence of operations on data objects. The operation of system update transaction is always write. For user transaction, the operation on non-temporal data objects could be read or write while operation on temporal data could only be read. There is certain execution time associated with each operation and the execution time of a transaction is the sum of the execution time of all its operations. Operations of one transaction is executed in sequential fashion. One operation can not be executed unless all previous operations are finished.

IV. CONCLUSION

The Distributed scheme is a decentralized one .The security is one of the handicaps of Distributed scheme. In this paper mainly concentrated on supplying security in circulated system. The user authentication is accomplished by utilising AIDA algorithm and note authentication is accomplished by Hash cipher algorithms. If the user wants to get access to the circulated system the first the user authentication is performed afterwards client get access to the distributed server. The circulated server provides the service to the authorized user. The service is supplied based assets, time slot etc. By this we achieve the quality of service such End-to-end delay, traffic shaping, congestion avoidance.

REFERENCES

[1] G. Sindhu,"Distributed System: Privacy Data Sharing And Achieving Qos", IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 5, Oct-Nov, 2013.

[2] D. Jana, A. Chaudhuri, and B. B. Bhaumik, "Privacy and anonymityprotection in computational grid services," Int. J. Comput. Sci. Applicat., vol. 6, no. 1, pp. 98–107, Jan. 2009

[3] Karr, "Secure statistical analysis of distributed databases, emphasizing what we don't know," J. Privacy Confidentiality, vol. 1, no. 2, pp. 197–211, 2009.

[4] J.W. Yoon and H. Kim, "A perfect collision-free pseudonym system," IEEE Commun. Lett., vol. 15, no. 6, pp. 686–688, Jun. 2011

[5] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Toolsfor privacy preserving distributed data mining," ACM SIGKDD Explorations

[6] Newsletter, vol. 4, no. 2, pp. 28–34, Dec. 2002. J. Wang, T. Fukasama, S. Urabe, and T. Takata, "A collusionresistantapproach to privacy-preserving distributed data mining," IEICE Trans.Inf. Syst. (Inst. Electron. Inf. Commun. Eng.), vol. E89-D, no. 11, pp. 2739–2747, 2006