International Journal of Mathematical Archive-3(1), 2012, Page: 203-209
Available online through www.ijma.info ISSN 2229 – 5046

A NEW DYNAMIC COORDINATORS SELECTION ALGORITHM IN DISTRIBUTED DATABASE SYSTEMS

Omar Saad*, Mona Abbass, Mohamed kouta and Ashraf Darwish

Department of computer science, Faculty of Science, Helwan University, Cairo, Egypt
E-mail: omarsd@gmail.com

(Received on: 15-12-11; Accepted on: 05-01-12)

ABSTRACT

Distributed database management systems (DDBMSs) pose different problems when accessing distributed and replicated databases. Particularly, access control and transaction management in DDBS require different mechanism to monitor data retrieval and update to databases. Current trends in multi-tier client/server networks make DDBMS an appropriate solution to provide access to and control over localized databases. One of the basic problems in DDBMS is two phase commit protocol that depends on one coordinator which responsible for executing the transaction to other participant in the network. In this paper we present a new algorithm in order to select dynamically more than one coordinator and sort them according to their availability in any arbitrary network. The proposed algorithm is applied and showed the improvement of the DDBMS in any communication network.

***Keywords**: Distributed database systems, two phase commit protocol, Concurrency control and Distributed transaction processing.*

INTRODUCTION:

Database systems have become among the most important and widely used computer applications. The increased reliance on database systems by many applications and users requires that they deal with a variety of demands involving great amounts of data. Such applications could be for banking, airline reservation or many other on-line information systems (Luk´aˇs 2007).

Consequently, a database management system (DBMS) should provide high performance, high availability, reliability and maintainability.

The data objects of a DBMS may be distributed among geographically dispersed locations. This distribution is required in organizations with different operating units, each having a local database that is shared with other sites. Such DBMSs are connected by a communication network and called DDBMSs. DDBMSs provide high availability and reliability by the distribution and possible replication of data among many sites. Each site has an independent DBMS. A transaction in a DDBMS may require the use of data not available on its local site; such a transaction is called a global transaction.

A global transaction splits into a number of sub transactions; one for each site containing the objects of its read and write sets (Bipin 1996).

Distributed systems are used to increase the computational speed of problem solving. These systems use a number of computers which cooperate with each other to execute tasks (MingXiong 2002). The control of distributed algorithms requires one node to act as a coordinator. If the coordinator crashes or fails for any reason, a new coordinator should be automatically selected to keep the network working. The graph center problem solves this problem by substituting the failed coordinator by a new deserved one. It starts when one or more nodes discover that the coordinator has failed. It terminates when the remaining nodes know who the new coordinator is as in (Mohammed 2010). This paper presents a new algorithm to select dynamically more than one coordinator and can sort them based on their availability in any communication network.

The remainder of this paper is organized as follows. Section 2 overviews briefly the basic definitions and main concepts related to DDBMSs. Section 3 introduce the related work. Consequently Section 4 presents the proposed algorithm. The proposed algorithm is implemented and the results are analysed in section 5. Finally section 6 draws the conclusions and future work.

***Corresponding author: Omar Saad*, *E-mail: omarsd@gmail.com**

BASIC CONSEPT AND BACKGROUND:

Distributed Database:

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (distributed DBMS) is the software system that permits the management of the distributed database and makes the distribution transparent to the users .The term distributed database system (DDBS) is typically used to refer to the combination of DDB and the distributed DBMS. Distributed DBMS are similar to distributed file systems in that both facilitate access to distributed data (Hela 1997;   Gray 1993).

Transaction:

A transaction consists of a series of operations performed on a database. The important issue in transaction management is that if a database was in a consistent state prior to the initiation of a transaction, then the database should return to a consistent state after the transaction is completed. This should be done irrespective of the fact that transactions were successfully executed simultaneously or there were failures during the execution. Thus, a transaction is a unit of consistency and reliability. A Transaction has four properties, atomicity, consistency, isolation, and durability that lead to the consistency and reliability of a distributed data base (Graz 2004).

Reliability in Distributed Database Systems:

DDBSs are potentially more reliable since there are multiple copies of each system component, which eliminates single points-of-failure, and data may be replicated to ensure that access can be provided in case of system failures. Distributed reliability protocols maintain the atomicity and durability properties by (a) ensuring that either all of the actions of a transaction that is executing at different sites complete successfully (called commit) or none of them complete successfully (called abort), and (b) ensuring that the modifications made to the database by committed transactions survive failures. The first requires atomic commit protocols, while the second requires recovery protocols (Hela 1997).

The correctness of a distributed protocol lies in its atomicity.The main issue to be considered is that, a distributed transaction is a set of independent transactions executing at different sites (being processed by their respective transaction mangers), but the result of transaction should be a consensus among all. Thus the transaction is initiated by a coordinator, which ensures the above property. The participating sites execute the transaction initiated by the coordinator independently on their local copies of database and commit/abort as per the final verdict of the coordinator (Date 1997). In addition to maintain the communication between the two, proper Logs have to be generated which are to be later used by the recovery manager, in case a site failure occurs? Several designs have been suggested for the problem. The two phase commit protocol one of the most important protocols which can be used to solve this problem.

Two Phase Commit Protocol:

This is the most common atomic commit protocol two phase commit (2PC) which has two types of node to complete its processes: the coordinator and the subordinate (Yousef 2004). The coordinator's process is attached to the user application, and communication links are established between the subordinates and the coordinator. The two-phase commit protocol goes through, as its name suggests, two phases. The first phase is a PREPARE phase, whereby the coordinator of the transaction sends a PREPARE message. The second phase is decision-making phase, where the coordinator issues a COMMIT message, if all the nodes can carry out the transaction, or an abort message, if at least one subordinate node cannot carry out the required transaction. The two phase commit protocol has various advantages such as: (a) it ensures atomicity even in the presence of deferred constraints, (b) it ensures independent recovery of all sites, (c) since it takes place in two-phase, it can handle network failures, disconnections and in their presence assure atomicity.

Presumed Abort Two Phase Commit:

This protocol is a variant of 2PC in which on abortion the coordinator does not force write abort decision, just sends abort messages and the participants too do not write logs in case of abort (Ansha 2004). In the absence of information about a transaction in its protocol database, a presumed abort (PrA) coordinator presumes the transaction has aborted. PrA makes the abort presumption systematically to further reduce the costs of messages and logging. Once a transaction has aborted, its entry is deleted since a missing entry denotes the same outcome. No information needs be logged about such transactions because their protocol database entries need not be recovered. We must guarantee that the protocol database always contains entries for committed transactions which have not yet completed all phases of 2PC.These entries must be recoverable across coordinator crashes. PrA deletes the protocol database entries for committed transactions when 2PC completes in order to limit the size of the database, and the same garbage collection strategies are also possible. A coordinator need not make a transaction's entry stable before its commit because an earlier crash aborts the transaction, and that is the presumed outcome in the absence of information. Only a commit outcome needs to be logged (with a forced write).Since there is no entry in the protocol database for an aborted transaction, there is no entry in need of deletion, and hence no need for an ACK of the ABORT outcome message. This protocol can reduce abortion cost as compared to 2PC, both at coordinator and participant end (Mohan 1986).

Presumed Commit Protocol:

This protocol is another variant of 2PC in which missing information about a transaction is presumed to be committed. However the coordinator force writes an initiation Record before sending prepare to ensure that the missing information of the Transaction is not misinterpreted as a commit after the coordinator's failure. 2PC and PrC consist of a voting phase and a decision phase. However, PrC reduces the cost of committing a transaction by not requiring that the participants to force write a commit log record to a stable storage and to acknowledge a commit decision during the decision phase. PrC achieves this by making an explicit commit presumption about the outcome of transactions in the absence of information about the transactions. That is, after recovering from a failure, when a participant in the execution of a distributed transaction inquires the coordinator of the transaction (i.e. the site where the transaction has been initiated) about the status of the transaction, the coordinator responds with a commit decision if it has no recollection about the transaction (Yousef 1997).

Fault Discovery: network connectivity:

One of the most fundamental network properties in distributed systems is connectivity. Consider an undirected graph G on n nodes, which is initially connected. Suppose, for a parameter $\varepsilon > 0$, we are interested in detecting ε partitions: failures of network elements after which there are two subsets of nodes A and B, each of size at least ε n, such that no node in A has a path to any node in B. For a parameter $k > 0$, we wish to be able to detect any ε-partition that is caused by the failure of up to k (adversarial chosen) nodes or edges, and record the occurrence of such an event. Note that increasing k allows an adversary more power, so handling larger values of k represents a sequence of successively more difficult problems. Here is a general approach for doing this, motivated by the type of analysis discussed above. We place "monitoring agents" at a subset D of the nodes of G, and each of these agents periodically engages in communication with all the others. Now, if at some point in time, there are nodes u, v D such that the agents at u and v have no path connecting them, we can record a fault in the network (Jon 2002).

Recovery Control: database recovery:

When a distributed database node crashes, its main memory contents are lost, and must somehow be recovered. Traditionally, the most common methods to keep on stable storage, a log of all update performed on the node's data. When a node recovers, it reads and applies all the log entries to the database. Since the log can grow arbitrarily large, logging is nearly always accompanied by the taking of periodic check pointier. Complete database images written to stable storage. Check pointing optimizes the time needed the most recent checkpoint have to be read during the recovery process. For recovery, since only the log records postdating the most recent checkpoint have to be read during the recovery process.

In distributed databases, the consistency of the recovered data must also be considered. If immediate consistency is enforced, once the node is reintegrated in the system, its data must be consistent with the data on the other nodes. Eventual consistency is required; the reintegrated node must still be in a state that is eventually consistent, i.e., a state that would become consistent after a bounded time period in a quiescent system (Sanny 2000).

Traffic Analysis in Distributed Systems:

Network traffic analysis could be defined as: "the inference of information from observation of the network traffic data flow". Analysis in general, and hence network traffic analysis, can be categorized by time (or frequency) criteria and by the purpose of the analysis.

(a)Time based analysis categorization: Regarding time and frequency criteria, any network traffic analysis can be classified in one of the following three categories: real-time analysis, batched analysis and Forensics analysis. The first two categories are not event orientated analysis in the sense that analysis is performed continuously, and not when a particular event occurs like forensics analysis does (Marc 2010).

(b)Real-time analysis: This analysis is performed over the data as it is obtained, or using small batches often called buffers to efficiently analyse data. The response time of this kind of analysis, understood as the time elapsed between a certain event occurs and is computed or detected, is low thanks to the low delay obtaining data and the fact that real-time analysis are fully automated. Real-time analysis though, has usually high computational resources requirements (Anu 2008).

Network Monitoring:

Network monitoring tasks have been taken place in computer networks since first networks where starting to be used. Network monitoring could be defined as the use of a system that constantly monitors a computer network for slow or failing components and that notifies the network administrator in case of problems.

Over the years, two different kinds of techniques mainly have been found effective for monitoring purposes:

(i) Agent based monitoring: agent based monitoring relies on a piece of software Running on the network devices that should be monitored (e.g. hosts and routers), called agent. This piece of software collects information from the device, such as the connectivity state of its network interfaces, link performance like through puts and any other information that may be of interest, and send them to a management platform through the same network or through a dedicated management network (Wells 2010).

(ii) Agent less monitoring: does not rely on agents collecting information from each of the hosts of the network under surveillance, but on analysing network traffic obtained directly from the network. In this sense this kind of systems typically supervises network traffic in terms of connection throughputs, packet routing information, and TCP window state to estimate congestion, and host services. This kind of systems may be totally passive systems, and hence do not interfere on the traffic flowing in the network or be also an active system, in which the monitoring system is able to deliberately inject packets to force devices to respond to them obtaining information by capturing and analysing devices responses. The weakness of these kinds of monitoring systems is that not all the information can be gathered from the network data observation, especially information related to of particular hardware and software parameters on the hosts which agents are able to supply. Most IT administrators agree that agent based monitoring and agent less network monitoring is complementary (Jon 1981).

RELATED WORKS:

In several researches, such as (David 1993), the authors developed a new form of presumed commit that reduces the number of log writes while preserving the reduction in message, bringing both these costs below those of presumed abort. The penalty for this is the need to retain a small amount of crash related information forever.

Most two-phase commit protocols, such as presumed-abort and presumed-commit, handle all committing transactions uniformly. In (Gopi 2002), author developed the presumed-either protocol that uses a dynamic strategy. Transactions which can be resolved efficiently using presumed commit are resolved that way. For other transactions, presumed abort is used. Presumed-either exploits the fact that transactions are processed concurrently and that log records for one transaction May be piggybacked into the log by log writes generated by others. Presumed-either offers a potential performance gain with little risk. In the worst case, committing a transaction using presumed either is no more costly than committing under the widely implemented presumed-abort protocol. Furthermore, presumed either has the attractive property that it is likely to perform best in high-performance systems that flush log pages frequently to stable storage.

In hard real-time control applications, components execute concurrently on distributed nodes and must coordinate to perform the control task under timing constraints. Timed atomic commitment is used to enforce the deadline on the decision and performance of commitment actions. The communication between nodes during atomic commitment is done by exchanging O (N*) messages, resulting in the degradation of the performance of real-time systems. The protocol also has to be non-blocking because strict timing constraints have to be met. Therefore the authors in (Johnny 1996) designed a consensus-based protocol that uses fewer messages than the protocol that requires every node to send messages to every other node and is non-blocking. One interesting property of the protocol is that it is naturally non-blocking. This is because there are time-out facilities at each step of the protocol.

THE PROPOSED ALGORITHM:

As appear from the previous sections, we noted that there is still the coordinator failure problem doesn't solve so far. In this paper we propose a new algorithm to solve this problem. The proposed algorithms are based on finding the nodes that represent the centers of the graph, where locating centers of graphs has a wide variety of important application because placing a common resource at a center of a graph minimizes costs of sharing the resource with other locations, and then we sort these centers according to their availability on the network, where The definition of availability is the probability that a device will perform a required function without failure under defined conditions for a defined period of time.

There are two main factors that are involved in the calculation of availability: Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR). MTBF is obtained from the data sheets of the equipment. MTTR is the average time to fix and restore the device in order to be put back into service. Once MTBF and MTTR are known, the availability of the component can be calculated using the following formula:

A = MTBF/ MTBF+MTTR:

The above discussion has focused on the availability of a single device but networks involve many devices all of which must be functioning in order to maintain the operation of the intended application. A dependent, serial system in which the end-to-end availability is a function of the availability of each item in the series. The network availability can be calculated by the following formula:

Network Availability = device 1 availability x device 2 availability x.... device n
availability The flowchart of the proposed model is given in figure 1 ,where a network is a finite, connected, undirected simple (i.e., no parallel edges and no self-loops) graph G (V, E) of n vertices and m edges, where V and E are the vertex and edge sets, respectively. A weighted graph G is a graph in which a weight w (e) is associated with every edge e € E. we assume that all edge weights are positive, where the cost here is the availability of each node in the network A path of a graph G is an alternating sequence of distinct vertices and edges, beginning and ending with vertices. The proposed model in figure (1) is divided into three phases.

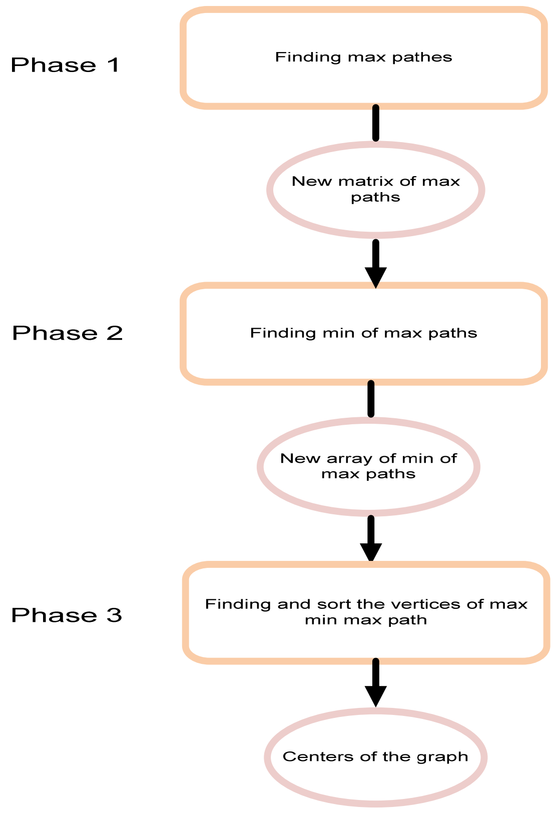

Figure: 1. The flowchart of the proposed graph center algorithm.

The proposed algorithm can be described as follows:

Input: A graph G (V, E)

Output: The centers of G.

Method: The algorithm consists of three phases:

Phase 1:
For each vertex v in G we will find the max paths (max availability) to all other vertices where the length of a path is the multiple of the weights of the edges in the path. A path from vertex u to v is a max path if there is no other path from u to v with higher length.

Phase2:
For each vertex v in G we will find the min path (min of max availability) from the max paths to all others vertices.

Phase3:
Finding centers of G which are the vertices with max of min max path (max of min max availability) and then sort them. The first center of a Graph G is the vertex that has the max of min max paths to all other vertices in G called (Max Min Max problem).

EXPREMENTAL RESULT AND ANALYSIS:

In this section we briefly describe the three scientific computational phases used in Experiments and present the results obtained for these phases after conducting the experimental tests. The goal of these experiments is to determine the efficiency and scalability of the proposed algorithm.

Let's consider a simple graph in figure (2) to demonstrate how to find the graph centers.

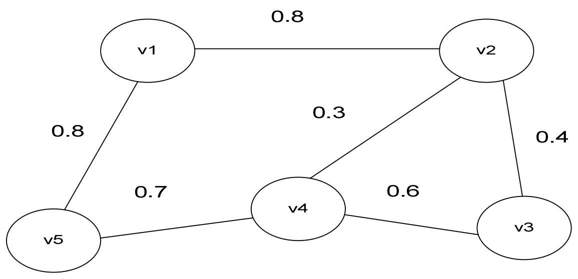

Figure: 2. Undirected connected Graph.

In figure (2) there are five vertices (v1, v2, v3, v4, v5) in the graph .So the input here will be the following matrix:

$$\begin{bmatrix} 0 & .8 & 0 & 0 & .8 \\ .8 & 0 & .4 & .3 & 0 \\ 0 & .4 & 0 & .6 & 0 \\ 0 & .3 & .6 & 0 & .7 \\ .8 & 0 & 0 & .7 & 0 \end{bmatrix}$$

Firstly we find for each vertex v in G all paths to all other vertices in the graph G and determine the max paths as in the following matrix:

$$\begin{bmatrix} 0 & .8 & .3 & .5 & .8 \\ .8 & 0 & .4 & .4 & .6 \\ .3 & .4 & 0 & .6 & .4 \\ .5 & .4 & .6 & 0 & .7 \\ .8 & .6 & .4 & .7 & 0 \end{bmatrix}$$

Secondly we find for each vertex v in G the min path from max paths to all other vertices in the graph G as in figure(3).

Then we find and sort the vertices according to max min paths in G so the result will be as in figure (4).

The first center in this example is v2 and the second one is v4 and the third one is v5.

.3	.4	.3	.4	.4

Figure: 3. Min path of max paths for each vertex v.

V2	V4	V5	V1	V3

Figure: 4. the centers of the graph G.

CONCLUSION AND FUTURE WORK:

One of the most fundamental problems in distributed systems is the coordinator failure. In this paper we presented a new algorithm to select more than one coordinator depending on their availability in the network and sort them such that the first coordinator is the node of max availability in the network. If a coordinator failure creates network partitions, the second one is chosen to be the new coordinator. When a node recovers from a crash, becomes a node without coordinator and so starts a new selection to find its coordinator. To show the effectiveness of the proposed algorithm; some applications are tested. Several possible extensions to improve the work are proposed here.

For the future work; an algorithm can be designed to solve the coordinator failure in different topologies such as meshes networks and can be investigated in a wireless communication environment.

REFERENCES:

Luk´aˇs H, Ludˇek M (2007), Nonblocking Distributed Replication of Versioned Files, journal of software, VOL. 2, NO. 5. pp. 16-23.

Bipin C, Boutros S (1996), Performance of a two-phase commit protocol. Information and Software Technology 38. pp. 581-599.

MingXiong A, Krithi R, Jayant R, John A (2002), mirror: a state-conscious concurrency control protocol for replicated real-time databases.Information Systems 27. pp. 277-297.

Mohammed R, Ahmad S, Fahad A (2010), Leader Election Algorithm in 2D Torus Networks with the Presence of One Link Failure. The International Arab Journal of Information Technology, Vol. 7, No. 2. PP. 105-115.

Helal A, Heddaya A, Bhargava B(1997), Replication Techniques in Distributed Systems.Kluwer Academic Publishers.

Gray J , Reuter A(1993), Transaction Processing: Concepts and Techniques. Morgan Kaufmann.

Ghazi A, Ronny S(2004) Transaction Management in Distributed Database Systems: the Case of Oracle's Two-Phase Commit, Journal of Information Systems Education, Vol. 13(2), PP 95-104.

Date J, Darwen H(1997), A Guide to the SQL Standard - A User's Guide to the Standard Database Language SQL, 4th edition, Addison-Wesley.

Yousef J, Panos K (2004), 1-2 PC: The one-two phase atomic commit protocol.

Ansha V, Mittal K(2004),One and Two Phase Commit protocols, Project Report, KReSit IIT, Bombay, India.

Mohan C, Lindsay B, Obermarck R (1986), "Transaction Management in the R* Distributed Database Management System," ACM Trans. Database Systems, vol. 11, no. 4, pp. 378-396.

Yousef J, Panos K, Steven P(1997), enhancing the performance of presumed commit protocol University of Pittsburgh PA 15260 pp. 131-133.

Jon K (2002), Detecting a Network Failure. Internet Mathematics Vol. 1, No. 1. pp. 37-56.

Sanny G, Sten F (2000), on recovery and consistency preservation in distributed real- time database systems. University of Skövde.

Marc S (2010), A framework for network traffic analysis using GPUs,UniversitatPolit`ecnica de Catalunya (UPC), Barcelona.

Anu P (2008), Minnesota Internet Traffic Studies (MINTS), University of Minesota. Wells W (2010), Simple Network Management Protocol, from http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol.

Wells W (2010), Simple Network Management Protocol, from http://en.wikipedia.org/Wiki/Simple_Network_Management_Protocol.

Jon P (1981),Transmission Control Protocol, DARPA Internet program protocol specificiation (TCP RFC), Information Sciences Institute University of Southern California4676 Admiralty Way Marina del Rey, California 90291.

David L, Butler L (1993), A New Presumed Commit Optimization for Two Phase Commit, DEC Cambridge Research Lab one Kendall Square, Bldg 700 Cambridge, MA 02139, pp. 630-640.

Gopi K,Kenneth S (2002), The Presumed-Either Two-Phase Commit Protocol. IEEE transactions on knowledge and data engineering, VOL. 14, NO. 5, pp. 1190-1196.

Johnny S, Soma M (1996), ANonblocking Timed Atomic Commit Protocol for Distributed Real-Time Database Systems, J. SYSTEMS SOFTWARE 34 pp. 161-170.

****