Distributed transaction management for distributed system

Business, Management

The concept of transactions and its application has found wide and often indiscriminate usage. Transaction Management deals with the problems of keeping the database in a consistent state even when concurrent accesses and failures occur, (Ozsu et al. , 1991). 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, (Ozsu et al. , 1991). Thus, a transaction is a unit of consistency and reliability. The properties of transactions will be discussed later in the properties section. In large enterprises, the model for distributed database applications has moved away from the client-server model to a multi-tier model with large database application software forming the middle tier.

A distributed transaction updates data on two or more network connected computer systems. Implementing robust applications in a distributedenvironmentis difficult because these applications are subject to many types of failures, including failures of the client, the servers, and the network. In the absence of distributed transactions, the application program itself must detect and recover from these failures. Distributed transactions eliminate this burden from the application developer.

Distributed database systems (DDBS) are systems that have their data distributed and replicated over several locations; unlike the centralized data

base system (CDBS), where one copy of the data is stored. Data may be replicated over a network using horizontal and vertical fragmentation similar to projection and selection operations in Structured Query Language (SQL). Both types of database share the same problems of access control and transaction management, such as user concurrent access control and deadlock detection and resolution.

On the other hand, however, DDBS must also cope with different problems. Access control and transaction management in DDBS require different rules to monitor data retrieval and update to distributed and replicated databases. Oracle, as a leading Database Management Systems (DBMS) employs the two-phase commit technique to maintain a consistent state for the databases. Since organizations tend to be geographically dispersed, a DDBS fits the organizational structure better than traditional centralized DBS.

Each location will have its local data as well as the ability to get needed data from other locations via acommunicationnetwork. Moreover, thefailureof one of the servers at one site won’t render the distributed database system inaccessible. The affected site will be the only one directly involved with that failed server. In addition, if any data is required from a site exhibiting a failure, such data may be retrieved from other locations containing the replicated data. The performance of the system will improve, since several machines take care of distributing the load of the CPU and the I/O.

Also, the expansion of the distributed system is relatively easy, since adding a new location doesn’t affect the existing ones. Properties of Transactions A Transaction has four properties that lead to the consistency and reliability of

a distributed data base. These are Atomicity, Consistency, Isolation, and Durability, (Ozsu et al. , 1991). Atomicity. This refers to the fact that a transaction is treated as a unit of operation. Consequently, it dictates that either all the actions related to a transaction are completed or none of them is carried out.

For example, in the case of a crash, the system should complete the remainder of the transaction, or it will undo all the actions pertaining to this transaction. The recovery of the transaction is split into two types corresponding to the two types of failures: the transaction recovery, which is due to the system terminating one of the transactions because of deadlock handling; and the crash recovery, which is done after a system crash or a hardware failure. Consistency. Referring to its correctness, this property deals with maintaining consistent data in a database system.

Consistency falls under the subject of concurrency control. For example, “ dirty data” is data that has been modified by a transaction that has not yet committed. Thus, the job of concurrency control is to be able to disallow transactions from reading or updating “ dirty data. ” Isolation. According to this property, each transaction should see a consistent database at all times. Consequently, no other transaction can read or transaction. If this property is not maintained, one of two things could happen to the data base, as shown in Figure 2: a.

Lost Updates: this occurs when another transaction (T2) updates the same data being modified by the first transaction (T1) in such a manner that T2 reads the value prior to the writing of T1 thus creating the problem of loosing

this update. b. Cascading Aborts: this problem occurs when the first transaction (T1) aborts, then the transactions that had read or modified data that has been used by T1 will also abort. Durability. This property ensures that once a transaction commits, its results are permanent and cannot be erased from the database.

This means that whatever happens after the COMMIT of a transaction, whether it is a system crash or aborts of other transactions, the results already committed are not modified or undone. Two Phase Commit Protocol The Two-Phase Commit Protocol (2CP) has two types of node to complete its processes: the coordinator and the subordinate, (Mohan et al. , 1986). 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. (Capitalization is used to distinguish between technical and literal meanings of some terminologies) The 2PC may be carried out with one of the following methods: Centralized 2PC, Linear 2PC, and Distributed 2PC, (Ozsu et al. , 1991). The Centralized Two-Phase Commit Protocol

In the Centralized 2PC shown in Figure 3, communication is done through the coordinator’s process only, and thus no communication between

subordinates is allowed. The coordinator is responsible for transmitting the PREPARE message to the subordinates, and, when the votes of all the subordinates are received and evaluated, the coordinator decides on the course of action: either abort or COMMIT. This method has two phases: 1. First Phase: In this phase, when a user wants to COMMIT a transaction, the coordinator issues a PREPARE message to all the subordinates, (Mohan et al. , 1986).

When a subordinate receives the PREPARE message, it writes a PREPARE log and, if that subordinate is willing to COMMIT, sends a YES VOTE, and enters the PREPARED state; or, it writes an abort record and, if that subordinate is not willing to COMMIT, sends a NO VOTE. A subordinate sending a NO VOTE doesn’t need to enter a PREPARED state since it knows that the coordinator will issue an abort. In this case, the NO VOTE acts like a veto in the sense that only one NO VOTE is needed to abort the transaction. The following two rules apply to the coordinator’s decision, (Ozsu et al. , 1991): a.

If even one participant votes to abort the transaction, the coordinator has to reach a global abort decision. b. If all the participants vote to COMMIT, the coordinator has to reach a global COMMIT decision. 2. Second Phase: After the coordinator reaches a vote, it has to relay that vote to the subordinates. If the decision is COMMIT, then the coordinator moves into the committing state and sends a COMMIT message to all the subordinates informing them of the COMMIT. When the subordinates receive the COMMIT message, they, in turn, move to the committing state and send an acknowledge (ACK) message to the coordinator.

When the coordinator receives the ACK messages, it ends the transaction. If, on the other hand, the coordinator reaches an ABORT decision, it sends an ABORT message to all the subordinates. Here, the coordinator doesn’t need to send an ABORT message to the subordinate(s) that gave a NO VOTE. The Linear Two-Phase Commit Protocol In the linear 2PC, as depicted in Figure 4, subordinates can communicate with each other. The sites are labeled 1 to N, where the coordinator is numbered as site 1. Accordingly, the propagation of the PREPARE message is done serially.

As such, the time required to complete the transaction is longer than centralized or distributed methods. Finally, node N is the one that issues the Global COMMIT. The two phases are discussed below: First Phase: The coordinator sends a PREPARE message to participant 2. If participant 2 is not willing to COMMIT, then it sends a VOTE ABORT (VA) to participant 3 and the transaction is aborted at this point. If participant 2, on the other hand, is willing to commit, it sends a VOTE COMMIT (VC) to participant 3 and enters a READY state.

In turn, participant 3 sends its vote till node N is reached and issues its vote. Second Phase: Node N issues either a GLOBAL ABORT (GA) or a GLOBAL COMMIT (GC) and sends it to node N-1. Subsequently, node N-1 will enter an ABORT or COMMIT state. In turn, node N-1 will send the GA or GC to node N-2, until the final vote to commit or abort reaches the coordinator, node The Distributed Two-Phase Commit Protocol In the distributed 2PC, all the nodes communicate with each other. According to this protocol, as Figure 5 shows, the second phase is not needed as in other 2PC methods.

Moreover, each node must have a list of all the participating nodes in order to know that each node has sent in its vote. The distributed 2PC starts when the coordinator sends a PREPARE message to all the participating nodes. When each participant gets the PREPARE message, it sends its vote to all the other participants. As such, each node maintains a complete list of the participants in every transaction. Each participant has to wait and receive the vote from all other participants. When a node receives all the votes from all the participants, it can decide directly on COMMIT or abort.

There is no need to start the second phase, since the coordinator does not have to consolidate all the votes in order to arrive at the final decision. Oracle database management System: The two phase commit The Oracle database is a distributed database management system, which employs the two-phase commit to achieve and maintain data reliability. The following sections explain Oracle’s two-phase implementation procedures. The Session Tree In each transaction, Oracle constructs a session tree for the participating nodes.

The session tree describes the relations between the nodes participating in any given transaction. Each node plays one or more of the following roles: 1. Client: A client is a node that references data from another node. 2. Database Server: A server is a node that is being referenced by another node because it has needed data. A database server is a server that supports a local database. 3. Global Coordinator: The global coordinator is the node that initiated the transaction, and thus, is the root of the session tree. The operations performed by the global coordinator are as follows:

• In its role as a global coordinator and the root of the session tree, all the SQL statements, procedure calls, etc. , are sent to the referenced nodes by the global coordinator. Instructs all the nodes, except the COMMIT point site, to PREPARE • If all sites PREPARE successfully, then the global coordinator instructs the COMMIT point site to initiate the commit phase • If one or more of the nodes send an abort message, then the global coordinator instructs all nodes to perform a rollback. 4. Local Coordinator: A local coordinator is a node that must reference data on another node in order to complete its part.

The local coordinator carries out the following functions (Oracle8): • Receiving and relaying status information among the local nodes • Passing queries to those nodes • Receiving queries from those nodes and passing them on to other nodes • Returning the results of the queries to the nodes that initiated them. 5. Commit Point Site: Before a COMMIT point site can be designated, the COMMIT point strength of each node must be determined. The COMMIT point strength of each node of the distributed database system is defined when the initial connection is made between the nodes.

The COMMIT point site has to be a reliable node because it has to take care of all the messages. When the global coordinator initiates a transaction, it checks the direct references to see which one is going to act as a COMMIT point site. The COMMIT point site cannot be a read-only site. If multiple nodes have the same COMMIT point strength, then the global coordinator selects one of them. In case of a rollback, the PREPARE and COMMIT phases are not needed and thus a COMMIT point site is not selected. A transaction is considered to be committed once the

COMMIT point site commits locally. Two-Phase Commit and the Oracle Implementation The transaction manager of the Oracle8 database necessitates that the decision on what to do with a transaction to be unanimous by all nodes. This requires all concerned nodes to make one of two decisions: commit and complete the transaction, or abort and rollback the transaction (Oracle8). The Oracle8 engine automatically takes care of the commit or rollback of all transactions, thus, maintaining the integrity of the database. The following will describe the two phases of the transaction manager.

1. PREPARE Phase (PP): The PP starts when a node, the initiator, asks all participants, except the commit point site, to PREPARE. In the PP, the requested nodes have to record enough information to enable them either to commit or abort the transaction. The node, after replying to the requestor that it has PREPARED, cannot unilaterally perform a COMMIT or abort. Moreover, the data that is tied with the COMMIT or abort is not available for other transactions. Each node may reply with one of three responses to the initiator. These responses are defined below: a.

Prepared: the data has already been modified and that the node is ready to COMMIT. All resources affected by the transaction are locked. b. Read-only: the data on the node has not been modified. With this reply, the node does not PREPARE and does not participate in the second phase. c. Abort: the data on the node could not be modified and thus the node frees any locked resources for this transaction and sends an abort message to the node that

referenced it. 2. COMMIT Phase (CP): Before the CP begins, all the referenced nodes need to have successfully PREPARED.

The COMMIT phase begins by the global coordinator sending a message to all the nodes instructing them to COMMIT. Thus, the databases across all nodes are consistent. Failure of the Two-Phase Commit A major problem with the two-phase commit occurs when one of the nodes participating in a distributed transaction fails while the transaction is in the PREPARED state. When the failure is for a prolonged period of time, then the data locked on all the other nodes won’t be available for other transactions. This will cause a lot of transactions to rollback due to deadlocks.

Oracle DBMS, in a new version, introduced an advanced queuing technique to deal with the problem of deadlock. The authors hope to address this technique in another paper in the near future. Exemplified: Distributed Database System Figure 6 illustrates the steps Oracle8 performs in order to PREPARE, Select the COMMIT Point Site, and COMMIT. The example in the figure depicts a company that has several branches located in different cities numbered 1 to 7. Each site has to have access to most of the data in the company in order to check on the status of purchase orders, material acquisition, and several other issues.

Since new projects are awarded and older projects are completed, project sites tend to change locations. Also, depending on the size and duration of a project, different COMMIT point strength can be assigned and thus, in the same area, different COMMIT point sites can be chosen, for a given location, over a period of time. In this example, City 1 is the head office and thus

posses the highest COMMIT point strength. The other sites are assigned the COMMIT point strength based on the Dollar volume of the project.

Higher monetary value for a project requires more resource allocation, and as such, will lead to more transactions executed against the data for that project. Since the amount of data involved is large, each site will have the portion of the database pertaining to its operations replicated and stored on a local server. Any transaction will at least affect the database at the head office and one of the sites. If, for example, a material rate, description of an item, accomplished progress, or purchase order is entered, a transaction is initiated that will affect the database at the head office and the database at the concerned site.

Other modifications, such as those involving employee transfer or equipment transfer from one site to another, will affect two or more sites. The following discussion explains the steps that entail in processing a distributed transaction: An employee is to be transferred from City 2 to City4. The transaction is initiated by City 1 by a personnel employee. The affected sites need to participate in the transaction. The processes that transfer one employee from one site to another should be grouped under one transaction so that either all or none of the processes are carried out.

Figures 7a-d depict the sequence of these activities. An explanation of these steps follows: 1. Since City 1 is initiating the transaction, it becomes the root of the session tree, i. e. the global coordinator. Since City 1 updates data in City 2 and City 4, it becomes a client. Since City 1 updates data on City 2 and City 4, the two nodes become database servers. 2. When the application

issues the COMMIT statement, the two-phase commit is started. 3. The global coordinator determines the COMMIT point site. 4. The global coordinator issues the PREPARE statement to all nodes except the COMMIT point site.

If any of the nodes cannot PREPARE, the transaction is aborted; otherwise, a PREPARED message is sent to the node that referenced it. 5. The global coordinator instructs the COMMIT point site to COMMIT. The COMMIT point site commits the transaction locally and records the transaction in its local redo log. 6. The COMMIT point site informs the global coordinator that it has committed and the global coordinator informs the other nodes by sending the COMMIT message. 7. When all the transactions have committed, the global coordinator informs the COMMIT point site to “ forget” about the transaction.

The COMMIT point site, after “ forgetting” about the transaction, informs the global coordinator, and the global coordinator, in turn, “ forgets” about the transaction. Use of the Two-Phase Commit The authors hope that this paper will encourage academicians to explain the concept of transaction management in distributed databases in database courses. The topic becomes particularly important with the introduction of OracleAcademicInitiative, where Oracle Corporation is donating its software to selected universities in the US.

The topic could be introduced towards the end of the first undergraduate database course, or in a graduate database course, where students had a first course in their undergraduate study. The paper provide a starting point

with its complete coverage of transaction management in distributed DBMS and the example on how Oracle implements the two-phase commit method. Transaction management is an old concept in distributed data base management systems (DDBMS) research. However, Oracle was the first commercial DBMS to implement a method of transaction management: the two-phase commit.

Though it was very difficult to obtain information on Oracle’s implementation of this method, the authors finally were able to collect enough information to write this paper. Many organizations do not implement distributed databases because of its complexity. They simply resort to centralized databases. However, with global organizations and multi-tier network architectures, distributed implementation becomes a necessity. It is hoped that this paper to will assist organization in the implementation of distributed databases when installing Oracle DBMS, or encourage organizations to migrate from centralized to distributed DBMS.

Universities could also contribute to this process by having graduates with the knowledge of Oracle DBMS capabilities. With Oracle making so much effort on incorporating this and other advanced features in its database software, academicians should also play a major role in exposing students to these advanced features. Aftergraduation, these students may assist organizations in applying these techniques to the real world. When we started this paper, we wanted it to be with practical significance rather basic theoretical research. We hope that we have accomplished this objective.