Replicated Data in a Distributed Environment

Malcolm Colton
Sybase, Inc.

SYBASE Replication Server™

Replication Server is a forthcoming Sybase product that dynamically maintains subsets of data in a distributed environment, providing several transaction models to maintain loose consistency. Replication Server is constrasted with existing products which provide location-transparent reads and high-consistency coordinated commits Replication Server makes it possible to build systems that are much more robust in the face of system component failures. By moving transactions rather than data, and by locating data at the point of processing, it maximizes the use of network bandwidth, enabling the deployment of robust, high-performance applications at a lower cost than with traditional approaches.

1: Introduction

Classical distributed database theory [1] posits seven major capabilities in distributed systems:

- A location-transparent *query processor* such that an application could remain unaware of data location.
- A *distributed data dictionary* to provide information about data available anywhere in the system
- A *distributed concurrency mechanism* to coordinate multi-user access at multiple sites
- A *distributed commit mechanism* to synchronize data changes at all sites
- Support for *replicated tables* to allow copies of the same table to be held at more than one side
- Support for *partitioned tables* to enable a single table to be split across many sites
- An update *propagation mechanism* to maintain replicated tables

To data no commercial relational database system has provided all these features, and it is becoming clear that this list does not describe the only possible architecture for a distributed system. In response to demand from customers attempting to build robust distributed applications, Sybase has developed a product called Replication Server™ that offers a different approach to building distributed systems - one that has significant advantages in many applications. While the more traditional approach emphasizes providing computational access to data no matter what its location, Replication.

Server addresses the problem of replicating data to the point at which it is to be used.

2: Replication Server

A Replication Server system is made up of LANs connected (typically) by WAN. Each LAN has at least one data server and the client programs that access them. Each LAN also has at least one Replication Server to manage movement of data to and from the data servers on its LAN. (It is possible to share REplication Servers between LANs, but that mode of operation will not be described here).

Each piece of data has a "primary" site which acts as the definitive copy of the data. The owner of each piece of data controls replication access to the data: who can replicate it, which tables and columns can be replicated and which columns can be used as keys and row version numbers.

At remote sites, users can enter "subscriptions" to primary data. A subscription lookks very like a SELECT statement: it defines the columns to be replicated and the restrictions to apply in order to select the rows. Once a subscription is entered, the primary site data that matches the subscription is copied to the remote site. From now on, the Replication Server will ensure that this replicate copy is kept as up to data as possible. Replication Server also

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0464...$1.50

supports the idea of a "dynamic subscription": when a program requests data that does not match a subscription, Replication Server will detect this event and enter a temporary subscription in every regard except that it expires after a certain period of disuse.

3: Transaction Model

Transactions are delivered by a reliable store-and-forward mechanism that guarantees that every transaction will ultimately arrive at its destination. Under normal operations the Replication Servers will attempt to move transactions as fast as possible. It is possible. however, to override this mode, and delay transactions to be applied in batches
At the application level the programmer has a choice of a number of transaction models:

- Local First
- Primary First
- Version-controlled
- Two-Phase Commit

3.1: Local First Update

The method which is most robust in the face of component and network failure is toupdate the local copy first. The Replication Server will ensure that changes to the local copy propagate to the primary copy where they will be applied and then replicated to other copies. Of course, this is a loose consistency protocol which allows for the creation of inconsistencies. It is possible that by the time the transaction is applied at the primary, it has become invalid. In such a case, and inconsistency report will be transmitted back to the source of the update via the Replication Servers. At the local sit, rules can be put in place to handle typical inconsistencies introduced by this transaction mechanism.

This transaction model would be chosen for low-value transactions and transactions where the cost of such an error is very much less than the cost of failing to perform the transaction. It is also suitable for the case where the chance of inconsistency is very low. The benefit of this method is that transactions can commit even when the primary site is unavailable.

3.2: Primary First Update

For transactions with a higher probability of failure, or for transaction where the cost of errors exceeds the cost of being unable to perform the transaction, Replication Server allows the primary copy to be updated first. Once the update succeeds at the primary, it propagates to all replicates, including the sit of origin of the transaction. This transaction model reduces the risk of inconsistency, but at the cost of a lower availability.

3.3: Version Controlled Update

In order to prevent lost updates, a remote site can ask to replicate the "version number" of each row. this control number is incremented each time the row is modified. the remote site can include the version number as part of the qualifying clause of its UPDATE statement. Thus, if a row has been changed since it was last copied to the replicate, the update will fail and an error message will be returned to the remote site. This optimistic concurrency model is useful for applications where there are occasional collisions over access to the primary data.

3.4: Two-Phase Commit

For most applications, two-phase commit will represent an unnecessary overhead

because an application will typically replicate all the necessary data to the local site and the perform a Local first transaction, the pieces of which migrate tot he various primary sites. However, there are some high-value transactions which require a tightly consistent simultaneous update of all primary sites. In this case, two-phase commit is the appropriate protocol.

4: Robustness

With current distributed database applications, it is very difficult to desynchronize a sit after it has been down for some period while transactions continued in other parts of the system. Replication Server automates this task. all transactions for an unavailable data server are saved reliably until the server recovers, and the automatically propagated to it. Thus a damaged site desynchronizes itself without any intervention.

5: Flexibility

Replication Server product provides a range of choices for building distributed applications so that the application architecture can fit the business problem, rather than forcing the business to fit into a Procrustean technological bed. Transaction protocol choices can be made by comparing the value of the transaction with the lost opportunity cost of being unable to conduct business.

6: Performance

Most data access in a typical application is read access. In a Replication Server system, all read access is local to the LAN, and so performs at high speed. Because the Replication Server moves transactions rather than data, only changes are moved between servers, so the WAN is optimized as well.

7: Heterogeneity

Replication server supports heterogeneous data servers. Each non-Sybase server is described with a server class definition which includes routing information as well as a mapping of the simple SQL DML statements to strings executable at the foreign server. Foreign data is defined in terms of virtual tables, columns and keys. If a data server is to act as a primary site, the developer must provide a means for extracting the transactions from the foreign data source and making them available to the Replication Server. A toolkit is provided for building this interface. Hence non-Sybase and even non-relational data can be included in a Replication Server system.

References

[1] C.J. Date, An Introduction to Database Systems, Volume I (Menlo Park, CA: Addison-Wesley Publishing Company, 1991).