Overview of Replication

GreatBridge™

Great Minds Are Open

A Work in Progress!

Agenda

- Overview of Replication Technology
- Overview of Commercial Replication Products
- Overview of PostgreSQL Replication Projects
- Customer Request
- Discussion

Overview of Replications

- Transaction Processing
- ACID
- Replication Types

GreatBridge

Transaction Processing

Transaction–The execution of a program that performs and administrative function by accessing a shared database

Transaction Processing Application–A collection of transaction programs designed to automate a given business activity

GreatBridge

Critical Properties of a Transaction–ACID

Atomic–Successful transaction commit; failed transaction abort

Consistent–Each transaction is programmed to preserve database consistency

Isolated–Each transaction executes as if it were running alone

Durable–The result of a committed transaction is guaranteed to be on stable storage

Replication Types

- Hot Fail Over–not really replication (But customers...)
- Read Only – (master/slave)
  - Read from many/local system and write to one master while the master updates the slaves.
- Peer to Peer –
  - Read and write from any/local system while each system is responsible for sending changes to other systems in the distribution.
  - Two types: Synchronous, Asynchronous

Hot Fail Over

- Hot Fail Over–the ability to fail a database from from a primary standalone server to secondary server.
  - Usually done locally
  - Hardware/Software (OS) solution
  - Multi–ported RAID
  - Heartbeat
  - WAL services must exists

Read Only Replication

- Pre database – A process that intercepts client data and sends writes to multiple systems.
- Post database – When the master database is updated, the slave(s) are updated via log event or trigger.

Peer–to–Peer

- Asynchronous (Lazy/Store and Forward) – Post commit sends information to all other systems involved in the distribution.
  - Trade Off: Data Synchronization and Conflict resolution.
- Synchronous (Eager) – Pre commit send information to all other systems in the distribution and verifies a commit or a roll back on each transaction for the entire distribution.
  - 2 Phase Commit vs. 2 Phase locking
  - Trade Off : Performance and Scalability.

P2P Synchronous

- Pessimistic – Synchronous replication in which all transactions read or write are propagated to every system in the distribution in the same order.
- Optimistic – Synchronous replication in which only writes are propagated to every system in the distribution in the same order. (All reads are performed locally.)

Overview of Commercial Replication Products

- Oracle
- Informix
- Sybase
- IBM DB2
- BEA TUXEDO
- Times Ten Data Replication

Oracle 8 Replication

- Types:
  - Basic replication– is a read only, which can use snapshots, increments or copy the entire database manually for a specified time interval.
  - Asynchronous Replication– Oracle has some automatic conflict resolution mechanisms via highest time stamp, but a DBA must decide which database is the correct copy.
  - Synchronous Replication– Oracle warns "this type of replication is highly dependent on system and network availability because it can function only when all sites in the system are available.

All events are initiated by "internal triggers" which preserves the user data trigger settings, and can can be either serial or parallel propagation via deferred transactions.

Informix Enterprise Replication

- Types:
  - Asynchronous replication
    - solution which uses a log post–processing method.
    - Parallel processing is used to filter transaction logs for marked replica transactions, which improves performance.
    - A conflict detection and resolution scheme incorporates either time–stamp, user–defined, or ignore.
    - Parsed queries are propagated to other database types.

IBM DB2

- Types:
  - Read Only
  - Asynchronous
    - The application is responsible for conflict detection and resolution.

Either Trigger based or Log based propagation

Sybase Replication Server

- Types: Called "Publish–and–Subscribe"
  - Read Only
    - Master/Slave
  - Peer–to–Peer (Asynchronous)
    - Update everywhere "lazy" configurations
    - In the update everywhere configuration, updates are may take place at any site, and conflict resolution has to be done by the application.are obtained from the log as soon as the log records are stored to disk in an effort to minimize the time that the copies are inconsistent.

In the update everywhere configuration, updates are may take place at any site, and conflict resolution has to be done by the application.

The transactions are parsed queries that still need to be processed on the remote systems.

GreatBridge

BEA TUXEDO Middleware

- Type
  - Transaction Processing
    - A transaction processing application would sit in between the application and the databases.
    - This requires the application to use the TP API to ensure replication, so access to the databases have to be restricted.
    - Using per–service load factors or prioritization, the transaction manager will send the request to the least busy server.

TimesTen Data Replication

- Type
  - Asynchronous replication system that monitors transaction logs to propagate queries to other systems.
    - It has store and forward, recovery, and peer to peer that uses time–stamp conflict resolution.

Overview of PostgreSQL Replication Projects

- Usogres (Tetsuichi Hosokawa)
- Rserver (Posgres, Inc.)
- PG Link (Joseph Conway)
- PostgreSQL Replicatior (IRCCS)
- Postgres–R(ETH Zurich)

Usogres

- Developer:Tetsuichi Hosokawa
- Type: Read Only, Pre–Process
- License: GNU
- Description:
  - Real–time Backup Utility
  - Replicates data via pre–postmaster
  - Supports only main server and backup server

Rserve

- Developer: Postgres Inc. (in CVS Contrib tree)
- Type: Read Only, Post–process
- License: BSD ?
- Description:
  - Uses Perl5 and PG–Perl interface (demo uses TCL and PG–TCL interface
  - Has Stored Procedures to set up Master and Slave databases
  - Uses a SyncID to keep track of the slave data updates
  - Supports Snapshots
  - Have to populate a Replication table on the master database to capture updates via trigger then updates can be manually synchronized via command line or by number of snapshots
  - Only one slave and no fail over support

PG Link

- Developer: Joseph Conway , 2001
- Type: Read Only, Post–process
- License: BSD
- Description:
  - Uses a C function which captures trigger events for inserts, updates, and deletes.
  - Each replicated table needs these triggers for each column.
  - Does support more than one slave
  - No fail over support

PostgreSQL Replicator

- Developer: Open Source project on Source forge (IRCCS)
- Type: Peer–to–Peer, Asynchronous
- License: GNU
- Description:
  - Creates a set of Replication Scheme Tables (RST) to store replication rules
  - Uses stored procedures to dynamically capture triggers and auxiliary tables, and act as an interface between the DBA and the replication engine.
  - TCL replication daemon running on each system in the distribution allows database synchronization to be started from any site at any time.
  - Robust update conflict detection and resolution mechanism
  - Uses triggers written in PL/TCL. The TCL daemon uses PostgreSQL connectivity API and TCL/DP libraries for communication over TCP/IP.

Postgres–R

- Developer: Information and Communication Systems Group, ETH Zurich, Switzerland
- Type: Peer–to–Peer Synchronous
- License: Unknown
- Description:
  - Uses Ensemble/Transis for group communication. (Ensemble Open Source)
  - Uses shadow copies to enforce isolation.
  - Its an extension to PostgreSQL, so all of the functionality is still available.
  - There is a remote back end for every local back end.
  - Uses either tuple changes to decrease processing on remote systems or query strings if the tuple changes are too much to propagate.
  - Currently supports one copy serialization isolation via 2–phase–locking.
  - There is a code branch that supports recovery and partial replication, but the recovery is only entire database synchronization. The partial implementation is simple and limited to a master/slave receiver driven approach.
  - There is no support for Asynchronous replication for off–line or WAN scenarios.

Postgres–R Overview

- Based on PostgreSQL 6.4.2
- Muticast communications
- 2–phase locking as concurrency control method.
- Branches
  - Original Source Tree
    - approx. 2,744 line changes in 40 files.
  - Recovery
  - particle replication
  - Ensemble, plus optimizations
    - remote backend pool of processes.
    - Approx. 17,050 line changes in 65 files.

GreatBridge

Major Changes

- heap/heapam.c – Added a simple tuple level lock mechanism.
- transam/xact.c – Changes for transaction aborts/termination, and new code for sending and waiting for write sets.
- executor/execMain.c – Changes to InitPlan, EndPlan, ExecutePlan for lock enhancements. The write set is prepared in ExecAppend, ExecReplace, and ExecDelete.
- postmaster.c – Signal changes
- lmgr.c – Many changes for currency control.
- multi.c – Enhanced with more lock modes for tuple locking.
- proc.c and lock.c – New queue for granted locks, remote transaction processing, and shadow copies.
- tcop/postgres.c – Changes in control of execution.

GreatBridge

New Files

- pg_ensemble*.c – group communication system.
- replicaManager.c – The heart of the replication component. Main tasks are socket management, process management, and send the right message across the right socket.
- Writeset.c – functions for marshaling and unmarshaling the write set.
- rmgrLib.c Socket management
- libpq/bufferedSock.c – provides buffered sockets.

GreatBridge

Replication Project

- Great Bridge Researchers
  - Darren Johnson, Senior Software Engineer (djohnson@greatbridge.com)
  - Frank Byrum, Chief Scientist (byrum@greatbridge.com)
- http://www.greatbridge.org/project/pgreplication/projdisplay.php

GreatBridge