About pgReplicatorII

From previous architecture...
We had preserved the philosophy of pgReplicator (I) regarding the way to perform replica: There is a *Logger System* in each replicated database, which logs all the operations performed during transactions. During a replica operation, the information of the all *Logger System*s are collected and analyzed by a *Conflict Resolution Alghotithm*. After that, valid operations are propagated to all sites and non-valid operations are rolled back.
For now, the *Logger System* is still performed via Triggers written in PL/Tcl, as in PgReplicator (I) but in the future we would like to obtain the same information from the postgresql WAL. Conlict Resolution Alghorithms in C++ are the same as in the previous version of PgReplicator but are written in C++ and are parts of the Replica Engine (see below)

Simpler installation and a GUI Replica Manager
Replica Manager is a G.U.I. Utility that can run both in Windows and Linux(i386) environments and does not need any operation on remote databases. All set-up operations, replica definition and so on are performed via this tool (Of course it is still possible to obtain the same result using SQL comands;-). No more tricky Tcl/DP installations, no more multiple installation on all postgresql servers involved in replica: simply enter some information in a grid (eg. hostname, dbname, postgres password, replicator password, postgresql version, etc..) and the tool will set-up *all* the hosts for you. Table Partition Type, conclift resolution alghorithms, and all parameters regarding replica can be set one time only in one shot via the GUI tool.

New Architecture
PgReplicatorII is quite different from the previous PgReplicator. All the code (except for triggers) is written in C++. In pgReplicator (I), the communication among Postgres SQL servers was peer-to-peer and it was hard to manage errors. In PgReplicatorII there is one (or some) process called *Replica Engine,* which controls all the SQL servers during replica. Both the *Replica Manager* and the *Replica Engine* can run on a server that is hosting a replicated database, or in a completely different server running Linux or Windows.
It is still possible to define the *Partition Type* of a single table (ReadOnly, ReadWrite, ReadWrite for the only Local Partition). We can also define a *Replica Group* that is a collection of replicated tables with the same *Conflict Resolution Alghotithm*. Moreover, a *Replica Cluster* defines one or more Replica Group.

Minor Enhancements
Some improvements has been added, for instance now you can replicate a table among different databases on the same host. BLOB support is native and more affordable. The whole system is developed on the current stable version of postgresql (7.3.4 in the time I am writing). Most of the specific Postgresql code is stored in a structure where you can specify a postgres verision for each server, so it will be *very* easy to mantain pgReplicatorII with following versions of postgresql.

Future Planning
In short, a new alpha-version (proof of concept) should be realased. I would be glad if someone will test it and tell me his impressions.
In medium time, thanks to the *Replica Group* objects we would like to test a non stopping replication able to mantain referential integrity. We would also permit some databases to be offline during replica operations, according to particular Conflict Resolution Alghorithms.

Features Resume:

General:

1. "Store and forward" asynchronous data replication.
2. Large Object replication natively supported
3. No Postgresql patch required
4. Based on PostgreSQL catalog and inheritance
5. Tested on 7.3.x versions.
6. Easy replica administration by GUI interface (Windows & Linux)

Data Ownership Models:

1. Table level replicated database set
2. Table level data ownership model
3. Master/Slave, Update Anywhere , Workload Partitioning (and more) data ownership models are supported.

Conflicts Resolution Algorithms:

1. Table level conflict resolution algorithm
2. Automatic forwarding and rolling-back transactions during database synchronization, according to the table conflicts resolution algorithm.
3. Several Conflict Resolution Algorithms are provided..

Replica Process

1. Point-to-multipoint architecture during replica process. (Replica-Engine to multiple-database-servers)
2. Replica process can be either scheduled or triggered by a maximum-operations threshold. Operation counters can be disabled maximizing performances
3. Operations that were rolled back by the conflict resolution algorithms are logged and can be submitted again.
4. Users whose operations were rolled back by the conflict resolution algorithms can be alerted via e-mail, and get transaction details (table level warning option) in "human readable" format.
5. Status of the replica process can be monitored via GUI or with tail command

Tests:

1. We are in alpha stage, sorry ;-).

Auxiliary tools provided:

1. Table restructure utility (simple as editing a text file). Useful for dropping columns or changing columns type
2. Database Functions to obtain some sort of "sequencies for distributed databases"