O'REILLY
ONLamp.com
LAMP: THE OPEN SOURCE WEB PLATFORM

Published on ONLamp.com (http://www.onlamp.com/)
http://www.onlamp.com/pub/a/onlamp/2004/11/18/slony.html
See this if you're having trouble printing code examples

Introducing Slony

by A. Elein Mustain
11/18/2004

Slony is the Russian plural for elephant. It is also the name of the new replication project being developed by Jan Weick. The mascot for Slony, Slon, is a good variation of the usual Postgres elephant mascot, created by Jan.

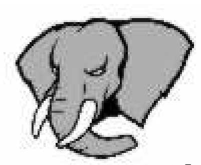
Figure 1. Slon, the Slony mascot.

Slony-I, the first iteration of the project, is an asynchronous replicator of a single master database to multiple replicas, which in turn may have cascaded replicas. It will include all features required to replicate large databases with a reasonable number of replicas. Jan has targeted Slony-I toward data centers and backup sites, implying that all nodes in the network are always available.

The master is the primary database with which the applications interact. Replicas are replications, or copies of the primary database. Since the master database is always changing, data replication is the system that enables the updates of secondary, or replica, databases as the master database updates. In synchronous replication systems, the master and the replica are consistent exact copies. The client does not receive a commit until all replicas have the transaction in question. Asynchronous replication loosens that binding and allows the replica to copy transactions from the master, rolling forward, at its own pace. The server issues a commit to the master client based on the state of the master database transaction.

Cascading replicas over a WAN minimizes bandwidth, enabling better scalability and also enables read-only (for example, reporting) applications to take advantage of replicas.

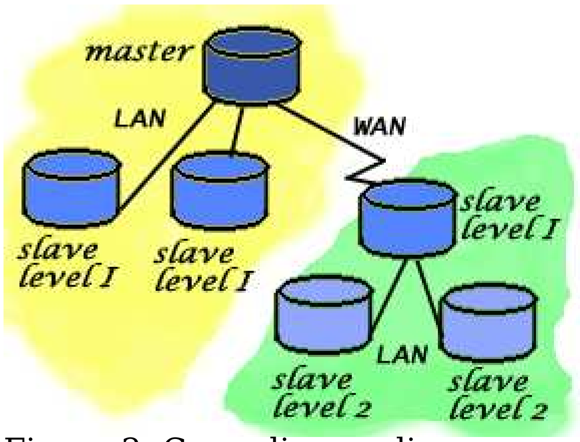

Figure 2. Cascading replicas

Assume you have a primary site, with a database server and a replica as backup server. Then you create a remote backup center with its own main server and its backup replica. The remote primary server is a direct replica, replicating from the master over the WAN, while the remote secondary server is a cascaded replica, replicating from the primary server via the LAN. This avoids transferring all of the transactions twice over the WAN. More importantly, this configuration enables you to have a remote backup with its own local failover already in place for cases such as a data center failure.

Slony's design goals differentiate it from other replication systems. The initial plan was to enable a few very important key features as a basis for implementing these design goals. An underlying theme to the design is to update only that which changes, enabling scalable replication for a reliable failover strategy.

The design goals for Slony are:

1. The ability to install, configure, and create a replica and let it join and catch up with a running database.

    This allows the replacement of both masters and replicas. This idea also enables cascading replicas, which in turn adds scalability, limitation of bandwidth, and proper handling of failover situations.

2. Allowing any node to take over for any other node that fails.

    In the case of a failure of a replica that provides data to other replicas, the other replicas can continue to replicate from another replica or directly from the master.

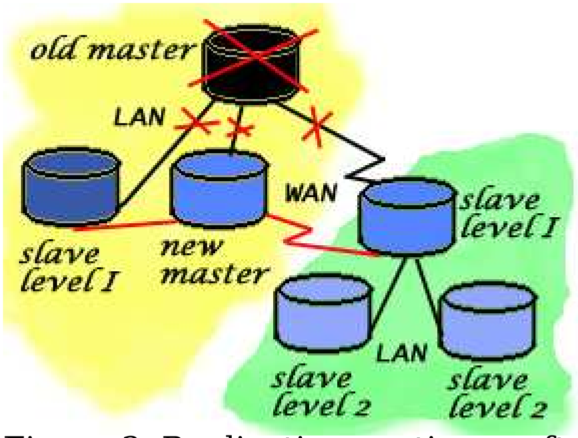

Figure 3. Replication continues after a failure

In the case where a master node fails, a replica can receive a promotion to become a master. Any other replicas can then replicate from the new master. Because Slony-I is asynchronous, the different replicas may be ahead of or behind each other. When a replica becomes a master, it synchronizes itself with the state of the most recent other replica.

In other replication solutions, this roll forward of the new master is not possible. In those solutions, when promoting a replica to master, any other replicas that exist must rebuild from scratch in order to synchronize with the new master correctly. A failover of a 1TB database leaves the new master with no failover of its own for quite a while.

The Slony design handles the case where multiple replicas may be at different synchronization times with the master and are able to resynchronize when a new master arises. For example, different replicas could logically be in the future, compared to the new master. There is a way to detect and correct this. If there weren't, you would have to dump and restore the other replicas from the new master to synchronize again.

It's possible to roll forward the new master, if necessary, from other replicas because of the packaging and saving of the replication transactions. Replication data is packaged into blocks of transactions and sent to each replica. Each replica knows what blocks it has consumed. Each replica can also pass those blocks along to other servers--this is the mechanism of cascading replicas. A new master may be on transaction block 17 relative to the old master, when another replica is on transaction block 20 relative to the old master. Switching to the new master causes the other replicas to send blocks 18, 19, and 20 to the new master.

Jan, said, "This feature took me a while to develop, even in theory."

3. Backup and point-in-time capability with a twist.

   It is possible, with some scripting, to maintain a delayed replica as a backup that might, for example, be two hours behind the master. This is done by storing and delaying the application of the transaction blocks. With this technique, it is possible to do a point-in-time recovery anytime within the last two hours on this replica. The time it takes to recover only depends on the time to which you choose to recover. Choosing "45 minutes ago" would take about one hour and 15 minutes, for example, independent of database size.

4. Hot PostgreSQL installation and configuration.

   For failover, it must be possible to put a new master into place and reconfigure the system to allow the reassignment of any replica to the master or to cascade from another replica. All of this must be possible without taking down the system.

   This means that it must be possible to add and synchronize a new replica without disrupting the master. When the new replica is in place, the master switch can happen.

   This is particularly useful when the new replica is a different PostgreSQL version than the previous one. If you create an 8.0 replica from your 7.4 master, it now is possible to promote the 8.0 to master as a hot upgrade to the new version.

5. Schema changes.

   Schema changes require special consideration. The bundling of the replication transactions must be able to join all of the pertinent schema changes together, whether or not they took place in the same transaction. Identifying these change sets is very difficult.

   In order to address this issue, Slony-I has a way to execute SQL scripts in a controlled fashion. This means that it is even more important to bundle and save your schema changes in scripts. Tracking your schema changes in scripts is a key DBA procedure for keeping your system in order and your database recreatable.

The first part of Slony-I also does not address any of the user interface features required to set up and configure the system. After the core engine of Slony-I becomes available, development of the configuration and maintenance interface can begin. There may be multiple interfaces available, depending on who develops the user interface and how.

Jan points out that "replication will never be something where you type SETUP and all of a sudden your existing enterprise system will nicely replicate in a disaster recovery scenario." Designing how to set up your replication is a complex problem.

The user interface(s) will be important to clarify and simplify the configuration and maintenance of your replication system. Some of the issues to address include the configuration of which tables to replicate, the requirement of primary keys, and the handling of sequence and trigger coordination.

The Slony-I release does not address the issues of multi-master, synchronous replication or sporadically synchronizable nodes (the "sales person on the road" scenario). However, Jan is considering these issues in the architecture of the system so that future Slony releases may implement some of them. It is critical to design future features into the system; analysis of existing replication systems has shown that it is next to impossible to add fundamental features to an existing replication system.

The primary question to ask regarding the requirements for a failover system is how much down time can you afford. Is five minutes acceptable? Is one hour? Must the failover be read/write, or is it acceptable to have a read-only temporary failover? The second question you must ask is whether you are willing to invest in the hardware required to support multiple copies of your database. A clear cost/benefit analysis is necessary, especially for large databases.

References

- General Bits Slony Articles on Tidbits
- The Slony-I Project documentation on GBorg
- Slonik Commands
- Jan Wieck's Original Slony-I Talk and Scripts, July 2004 in Portland, OR,

sponsored by Affilias Global Registry Services
- Information from IRC's #slony on freenode.net
- The Slony1-general@gborg.postgresql.org mailing list

A. Elein Mustain has more than 15 years of experience working with databases, 10 of those working exclusively with object relational database systems.

Return to ONLamp.com.