Asynchronous Replication for PostgreSQL

Slony-I on Microsoft Windows

Dave Page
Slony-I

- Asynchronous Replication for PostgreSQL
- Developed by Jan Wieck of Afilias
- Now a community project
Uses

- Load balancing
- Redundancy/Failover
- Remote/distributed servers
- Upgrades
Architecture
Flexible Topology
Object types

- Data
- Sequence values
- Schema changes
Porting team

- Hiroshi Saito
  - pgAdmin, psqlODBC, Npgsql, pgInstaller

- Dave Page
  - pgAdmin, PostgreSQL, psqlODBC, pgInstaller, Npgsql, pgWeb

- Magnus Hagander
  - PostgreSQL Server, pgInstaller, pgWeb

- Andreas Pflug
  - pgAdmin, PostgreSQL Server
Provided the initial ‘quick n dirty’ port.

Organised the project
‘Ducttape’ test suite

New regression test suite

Build system/Makefiles

Patch/CVS management
Magnus

- Code port
  - slonik – based on Hiroshi’s work

- slon
  - Reuse pgpipe from PostgreSQL
  - Service control code
  - Event logging
GUI Management using pgAdmin
Slony-I
Asynchronous Replication for PostgreSQL

Porting Slony-I

Magnus Hagander
Porting overview

- The bad
  - Designed for Unix
  - Relied on Unix tools and architecture

- The good
  - Portable between Unixes
  - Based on PostgreSQL build system
Porting Slonik - easy

- No shell utils available
  - Slonik 1.1 uses SED
  - Hiroshi already fixed
- Path issues
  - Find the "share" directory
  - Windows compatible paths
Porting Slon - easy

- Pthreads
  - Find library to link with
- Winsock
  - Simple initialization issue
- Pipes
  - Steal from PostgreSQL
- Signals
  - Ignore!
Porting slon – a bit more work

- Eventlog integration
  - Centralised logging already
  - Eventlog when service, stdout when console
  - Create message library

- Versioning metadata
  - Steal most from PostgreSQL
  - Decimal version number in config.h
Porting slon – most work

- fork()
- Service integration
- Two problems, one solution
Slon – Windows architecture

Service Control Manager

rc.slony

slon watchdog

slon engine

rc.xyz

slon watchdog

slon engine

CreateProcess() CreateProcess()
Porting – end result

- Slon runs on the commandline
  - Only for testing/debugging!
- Single service, multiple engines
  - One config file per engine
  - Paths stored in registry, add/remove with slon commandline
- Multiple services, multiple engines
  - Different versions of slon
Simple slony replication

DEMO
Base table creation

db1
CREATE TABLE t (  
    name text NOT NULL PRIMARY KEY)  
INSERT INTO t VALUES ('Dave')  
INSERT INTO t VALUES ('Magnus')

db2
CREATE TABLE t (  
    name text NOT NULL PRIMARY KEY)
Installing Slony

```
slon -regservice
slon -addengine c:\slony\db1.conf
slon -addengine c:\slony\db2.conf
slon -listengines

db1.conf
log_level=1
log_timestamp=false
cluster_name='test'
conn_info='host=127.0.0.1 user=postgres
dbname=db1'
```
# Create slony cluster
cluster name = test;
node 1 admin conninfo = 'host=127.0.0.1
    user=postgres database=db1';
node 2 admin conninfo = 'host=127.0.0.1
    user=postgres database=db2';
init cluster (id=1, comment='Node 1')
# Create set of tables with one table
create set (id=1, origin=1)
set add table (set id=1, origin=1, id=1, fully qualified name = 'public.t')
# Create node for second engine
store node (id=2, comment='Node 2');

# Create paths between the two nodes
store path (server=1, client=2,
    conninfo='host=127.0.0.1 user=postgres dbname=db1');
store path (server=2, client=1,
    conninfo='host=127.0.0.1 user=postgres dbname=db2');
store listen (origin=1, provider=1, receiver=2);
store listen (origin=2, provider=2, receiver=1);

# Subscribe the slave
subscribe set (id=1, provider=1, receiver=2,
    forward=no)
Asynchronous Replication for PostgreSQL

Graphical management of Slony-I

Andreas Pflug
pgAdmin III architecture

- C++
- wxWidgets 2.6
- Native libpq PostgreSQL connection
- Native Windows and GTK2 look and feel
- For PostgreSQL 7.3 and above
- Some helper programs and modules
Slony-I installation: modules

- Performed by Windows installer
- Performed by make;make install from source
- Use identical Slony-I versions on all servers!
- PostgreSQL servers may have different versions and run on different operating systems
Slony-I installation: Create cluster

- First node in cluster
- Node: database with installed cluster and running slon process
- Uses Slony-I creation scripts
- See pgAdmin's slony path option
Slony-I installation: Join cluster

- Create node and copy replication configuration from existing node
- Installs software in current database from existing cluster
Slony-I installation: paths

- Path: describes how a slon process connects to other nodes
- Libpq connect string
Listen: instructs a node to poll events from other nodes
Slony-I cluster status

- See node statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last event</td>
<td>452</td>
</tr>
<tr>
<td>Last event timestamp</td>
<td>10/05/2005 07:54:41 PM</td>
</tr>
<tr>
<td>Last acknowledged</td>
<td>71</td>
</tr>
<tr>
<td>Last ack timestamp</td>
<td>10/05/2005 06:52:09 PM</td>
</tr>
<tr>
<td>Last response time</td>
<td>0.004615 s</td>
</tr>
<tr>
<td>Outstanding acks</td>
<td>381</td>
</tr>
<tr>
<td>No ack for</td>
<td>01:02:31.512139</td>
</tr>
<tr>
<td>Hanging event</td>
<td>72</td>
</tr>
<tr>
<td>Command</td>
<td>SYNC</td>
</tr>
</tbody>
</table>
Slony-I replication sets

- Set: collection of tables and sequences
- All table and sequence data originating on one node
Table needs unique index, PK preferred

pgAdmin doesn’t offer tables without unique index

Select triggers on the table that Slony-I should disable on slave nodes
Table and sequences must be present in slave node before subscribing!

Subscribed sets can’t be modified; use merge set instead.
Slony-I DDL script replication

- Use replication to execute changes to subscribed tables to insure master and slave have identical definitions
- May replicate any DDL script
Slony-I switch over

- Gracefully exchange master and slave role to a set between two nodes
- Both nodes must be fully functional
- Function "move set"
Slony-I fail over

- Master node has failed
- Failover tries to restore as much data from slave nodes as possible
- Designate a new master out of the previous slaves
- Not yet supported in pgAdmin III V1.4
pgAdmin future

● Coming in V1.6:
  ● Slony-I failover support
  ● Set creation wizard
  ● Health analysis improvements
Slony-I and pgAdmin Conclusion

- Most Slony-I functions accessible through easy-to-use GUI
- At-a-glance view on cluster health
- Integrated with other administrative tasks

PostgreSQL has integrated replication!