Slony versus Developers

Christopher Browne
Afilias Canada
Postgres Open 2011

Agenda

- Overview of replication systems
- How does Slony work?
- Changes Slony Can't Capture
- Clever things we do with Slony

Overview of Replication Systems

- What is replication?
  - *Dynamically* duplicating activity from one DB into another DB
    - INSERTs
    - UPDATEs
    - DELETEs

Why Replication?

- Failover
  - Includes “avoiding failure”: *maintenance*
- Division of work
  - load balancing
  - doing *different* work on a replica
- Fast upgrades
  - Create replica on new version, “fail” over

Major Technologies

- Transaction log (WAL) capture
  - Built-in for Postgres 8.0+, steadily improving...
- Trigger-based replication
  - Slony, *Londiste, Bucardo, eRServer...*
- Statement capture
  - Exists on DB2, MySQL(tm), not on Postgres
  - Problematic
    - Nondeterministic updates, interleaving

WAL Rep - Shortcomings

- ***ZERO*** variation of schema on replica
  - Indices on replicas - inconvenience or fatal?
  - No altered behavior on replicas - *e.g.* - triggers
- Pure read-only access
  - Reports cannot use temporary tables
- No good for upgrades
  - Can't replicate across versions or architectures

Compare to Slony

- Complex to configure
- Clusters fragile – more moving parts
- DDL not replicated automatically
- ~15% write perf. degradation

- Sophisticated failover options
- Mods possible on subscribers
- Concurrent use of subscribers
- Fine for rapid Postgres upgrade

Strength = Weakness

- WAL is below-the-water magic
- Developers aren't likely to accidentally touch anything below the waterline
- Slony uses visible database features
- Developers might accidentally change stuff out from under Slony

How does Slony work?

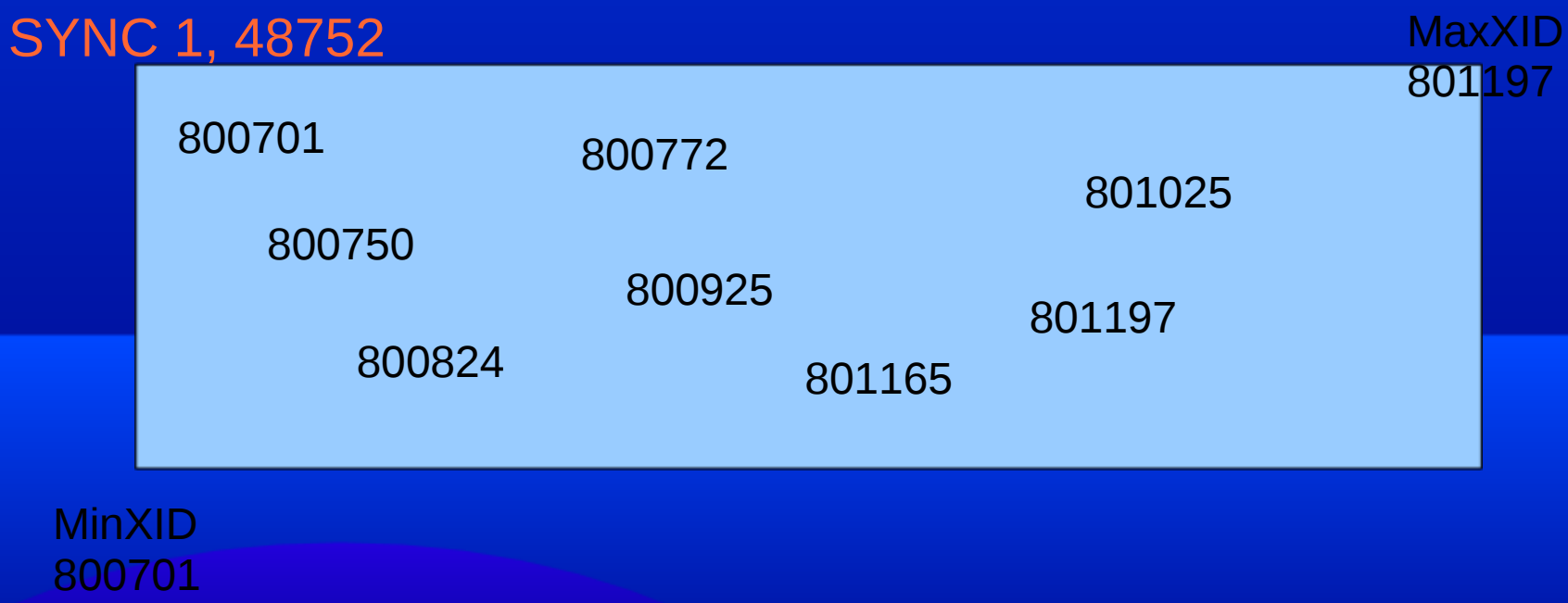
- Slonik – configuration tool - SQL-ish language
- Slon – C daemon for each node
- Slony Schema
  - Configuration & state in DB tables
  - Stored functions used by slon/slonik
- Triggers on each replicated table capturing INSERT/UPDATE/DELETE

Replication Trigger Functions

- log_trigger('cluster', table_id, 'kvkkvk')
  - Captures INSERT/UPDATE/DELETE + txid + sequence
  - Indirect changes also work:
    - Stored functions that do INSERT/UPDATE/DELETE fire log_trigger
    - Triggers that fire stored functions …
- denyaccess()
  - prevents corrupting replica

What's in a SYNC?

- active XID
- List of O/S XIDs: Active Transactions



Bank Reconciliation

- Outstanding transactions list, end of last month
- Start of Month
- Transactions on our books this month
- Outstanding transactions list, end of this month

Slony/Londiste Replication

- Outstanding transactions list, end of last SYNC/Tick point
- Start of new SYNC/Tick
- Transactions issued during this SYNC
- Outstanding transactions list, end of this SYNC/Tick

Devs Challenge: Schema Change

- New table? Not too hard...
  - Use psql to load DDL on all nodes
  - CREATE SET
  - SET ADD TABLE/SEQUENCE
  - SUBSCRIBE SET
  - Can be routinely done while system **active**
- **Easier with WAL replication**
  - **It just replicates *everything*...**

Schema Modification II

- ALTER TABLE
  - Slony can't capture this automatically
  - Must set up a script to run via slonik EXECUTE SCRIPT
  - Requires locks on tables, which often mandates outages
  - Failure to use EXECUTE SCRIPT likely to cause ***failure of the cluster!!!***
- WAL replication...
  - Just replicates... Everything...

Changes NOT Capturable by Slony

- ALTER TABLE
- CREATE TABLE
- DROP TABLE
- *TRUNCATE TABLE *(well, 'til PG 8.4)*
- **Capturing ALTER TABLE is Very Hard**

WRONG Upgrade Approach

- Write a program embedding DDL changes
  - If run directly, it will break replication ***badly***
  - If program queries schema to figger out changes, DBAs mayn't be able to figger out what should be in EXECUTE SCRIPT
  - DBAs cannot control changes going into production, perhaps inducing failure of S-O, SAS 70, ISO9K audit
  - People get fired. (Or perhaps should be.)

What are these standards?!?

- Sarbanes Oxley - legal response to Enron
- SAS 70 - Service Organization Audit
- ISO 27001 - Information Security Standard
- ISO 9001 - Quality Management Systems

Sorts of things involved

- Verifying that processes are documented
  - Do changes get deployed in controlled ways?
  - Can all changes be traced back to establish they were properly controlled?

Audit Mechanisms

- Ticketing system (RT, Bugzilla, Mantis, ...)
- System level logging
  - Unix logs, Postgres logs
- Policies
  - Who's allowed access?
  - Documented processes to implement changes (create user, alter schema,...)
- Sometimes, capture audit logs in DB

The point

- They won't tell you what your policies should be
- They want you to have policies
  - Written down
  - Evaluable
  - Evaluable *so that they can audit that they have been applied*

Trigger-Based Performance Wins

- There's a performance *win* to be had versus WAL-based
- Essentially in that it's easy to do **consistent** queries against a replica
  - With WAL, need to manage vacuum on the master to avoid data loss
  - With Slony, no special management is needed

Why WAL has a problem here

- Replica is tracking master... OK!
- You start running a 5 hour query against replica...
- Vacuum runs on master, trims old data.
- WAL for that vacuum replicates... OOPS!!!
- Workaround: Open a transaction on the master that runs for 5 hours... Ick!!! :-(
- Alternative: *hot_standby_feedback*, if on 9.1.
  - Still "icky" - holds onto data on master for 5h :-(

Clever(ish) Uses of Slony

- Data Capture on Replicas
  - WHOIS Cache Management
  - DNS Change Capture
    - Cannot be done without “trigger” approach!
  - Eliminates performance "hit" on master node

WHOIS Cache

- WHOIS service feeds off a replica
- Uses cache table containing constructed WHOIS record
- Cache invalidation triggers *on replica*
  - On UPDATE/DELETE on domain/host/contact
  - Capture ID of object for cache manager to trim from cache
  - ***ZERO*** performance impact against master

DNS Processing

- Replica has triggers on crucial tables
- Capture object IDs whenever interesting objects are modified
- DNS state recomputed based on objects that have been modified
- ***ZERO*** processing cost on master node

DNS - Quasi-MultiMaster

- DNS Master != Registry Master
  - DNS work split off of registry master altogether
  - Helpful to performance
  - Complicates failover
  - Need some more Slony changes to *properly* support failover with multiple origins :-(

In Closing...

- Clustering is complex to manage
- Requires discipline, regardless of method
- Slony has costs & benefits vs built-in WAL
- Work *is* ongoing on Slony - v2.1 next week!

Questions?