Built-in Replication in PostgreSQL 9.0

Heikki Linnakangas

Built-in replication

- Streaming Replication
  - Allow WAL records to be streamed to standby as they're generated
- Hot Standby
  - Allow read-only queries in standby server

1 + 1 = 3

Standby mode

EnterpriseDB®
The Enterprise Postgres Company

- In PostgreSQL 9.0, a server can be put into *standby mode*
  - By setting standby_mode=on in recovery.conf

```
# data/recovery.conf file
standby_mode = 'true'
restore_command = 'cp /home/hlinnaka/pgsql.cvshead/walarchive/%f %p'
primary_conninfo = 'host=localhost port=5432'
```

- In standby mode, recovery doesn't end when the end of WAL is reached

Standby mode

- Three sources of WAL:
  1. Archive, via restore_command
  2. Existing files in pg_xlog
  3. Stream from master
- In standby mode, the server repeatedly tries all three sources

File-based Warm standby server

- In master, postgresql.conf:

  wal_level='archive'

  archive_mode='on'

  archive_command='cp -i %p /walarchive/%f < /dev/null'
- In standby, recovery.conf:

  restore_command = 'cp /walarchive/%f %p'

  standby_mode = 'true'

  trigger_file = '/home/postgres/failover_now'

- This replaces pg_standby
  - But old config using pg_standby still works!

Warm-standby, startup

EnterpriseDB®
The Enterprise Postgres Company

~/pgsql.cvshead$ bin/postmaster -D data-standby -p 5433

LOG: database system was interrupted while in recovery at log time 2010-05-15 14:56:37 EEST

HINT: If this has occurred more than once some data might be corrupted and you might need to choose an earlier recovery target.

LOG: entering standby mode

LOG: restored log file "000000010000000000000002C" from archive

LOG: redo starts at 0/2C000020

LOG: restored log file "000000010000000000000002D" from archive

LOG: consistent recovery state reached at 0/2E000000

cp: cannot stat `/walarchive/000000010000000000000002E': No such file or directory

cp: cannot stat `/walarchive/000000010000000000000002E': No such file or directory

cp: cannot stat `/walarchive/000000010000000000000002E': No such file or directory

cp: cannot stat `/walarchive/000000010000000000000002E': No such file or directory

cp: cannot stat `/walarchive/000000010000000000000002E': No such file or directory

Warm-standby, failover triggered

EnterpriseDB®
The Enterprise Postgres Company

LOG: trigger file found: /tmp/triggerfile

LOG: redo done at 0/2D000088

LOG: last completed transaction was at log time 2010-05-15 14:56:39.6291+03

LOG: restored log file "000000010000000000000002D" from archive

cp: cannot stat `/walarchive/00000002.history': No such file or directory

LOG: selected new timeline ID: 2

cp: cannot stat `/walarchive/00000001.history': No such file or directory

LOG: archive recovery complete

LOG: database system is ready to accept connections

LOG: autovacuum launcher started

Hot Standby

EnterpriseDB®
The Enterprise Postgres Company

- You can connect to a standby server and run read-only queries.

- In master postgresql.conf:

  wal_level = 'hot_standby'

- In standby postgresql.conf:

  hot_standby=on

Hot standby, startup

EnterpriseDB®
The Enterprise Postgres Company

~/pgsql.cvshead$ bin/postmaster -D data-standby -p 5433

LOG: database system was interrupted while in recovery at log time 2010-05-15 15:31:25 EEST

HINT: If this has occurred more than once some data might be corrupted and you might need to choose an earlier recovery target.

LOG: entering standby mode

LOG: restored log file "000000010000000000000002F" from archive

LOG: redo starts at 0/2F000020

LOG: consistent recovery state reached at 0/30000000

LOG: database system is ready to accept read only connections

cp: cannot stat `/walarchive/000000010000000000000030': No such file or directory

LOG: unexpected pageaddr 0/28000000 in log file 0, segment 48, offset 0

cp: cannot stat `/walarchive/000000010000000000000030': No such file or directory

cp: cannot stat `/walarchive/000000010000000000000030': No such file or directory

cp: cannot stat `/walarchive/000000010000000000000030': No such file or directory

Streaming Replication

- WAL records are shipped from master as they're generated
- Asynchronous

- You could do record-based log shipping in previous releases, but now:
  - No custom development required
  - Works well with Hot Standby

Streaming replication

EnterpriseDB®
The Enterprise Postgres Company

- In master, postgresql.conf:
  - wal_level='archive'
  - max_wal_senders = 5
  - wal_keep_segments=100
- In master, edit pg_hba.conf to allow standby connections
- In standby, recovery.conf:
  - standby_mode = 'true'
  - primary_conninfo = **'host=localhost port=5432'**

Streaming Replication, startup

EnterpriseDB®
The Enterprise Postgres Company

~/pgsql.cvshead$ bin/postmaster -D data-standby -p 5433

LOG: database system was interrupted while in recovery at log time 2010-05-15 15:31:25 EEST

HINT: If this has occurred more than once some data might be corrupted and you might need to choose an earlier recovery target.

LOG: entering standby mode

LOG: redo starts at 0/2F000020

LOG: record with zero length at 0/2F0000A0

LOG: streaming replication successfully connected to primary

LOG: consistent recovery state reached at 0/30000000

LOG: database system is ready to accept read only connections

Putting it all together

EnterpriseDB®
The Enterprise Postgres Company

Standby with Streaming Replication, backed by a WAL archive

- In master, postgresql.conf:

```
wal_level='hot_standby' # or 'archive'
max_wal_senders=5
archive_mode=on
archive_command='cp -i %p /walarchive/%f < /dev/null'
```

- In standby, recovery.conf:

```
standby_mode = 'true'
restore_command = 'cp /walarchive/%f %p'
primary_conninfo = 'host=localhost port=5432'
```

Summary: Standby options

EnterpriseDB®
The Enterprise Postgres Company

Recovery.conf options:

```
#standby_mode = 'off'
#trigger_file = ''
#restore_command = '' # e.g. 'cp /mnt/archivedir/%f %p'
#primary_conninfo = ''  # e.g. 'host=localhost
port=5432'
```

In standby postgresql.conf:

```
#hot_standby = 'off'
#max_standby_delay = 30s
```

Summary: Configuration options

EnterpriseDB®
The Enterprise Postgres Company

In master postgresql.conf

#wal_level = minimal # minimal, archive, or hot_standby

#max_wal_senders = 0 # max number of walsender processes

#wal_keep_segments = 0 # in logfile segments, 16MB each; 0 disables

#vacuum_defer_cleanup_age = 0 # num transactions by which cleanup is deferred

wal_level

wal_level controls how much information is written to the WAL log:

- 'minimal' – the default
  - Suitable for crash recovery only. WAL archival or streaming replication can't be enabled.
  - Some operations like CREATE INDEX are faster because WAL-logging can be skipped
  - Was previously controlled by archive_mode='off'
- 'archive'
  - Allows WAL archival and streaming replication
  - Hot standby not allowed in the standby
  - Was previously controlled by archive_mode='on'
- 'hot_standby'
  - Like 'archive', but adds extra information about running transactions.
  - Allows hot standby mode in standby servers

max_wal_senders

EnterpriseDB®
The Enterprise Postgres Company

- Controls how many concurrent streaming replication connections allowed from standby servers.
- Set at least to the number of standby servers + safety margin

- In case of network problems, it can take a while for the master to notice that a TCP connection is broken. Allow some wiggleroom for that.

wal_keep_segments

EnterpriseDB®
The Enterprise Postgres Company

- If standby falls behind too much, so that the WAL it needs have already been recycled in the master, the standby cannot continue recovery.
- wal_keep_segments sets the number of WAL files that are retained, in case a standby still needs them
  - Actual number of files retained could be higher, if archiving is set up but not working, or if wal_keep_segments is smaller than checkpoint interval
- There's no safe setting. Bigger is better, but don't run out of disk space.
- Not needed if you use archiving

Hot standby conflicts

EnterpriseDB®
The Enterprise Postgres Company

- A read-only transaction can conflict with WAL replay in a hot standby server
- Example:
  - A vacuum record is being replayed, that removes tuple X. However, tuple is still visible to an old, long-running reporting query. If it's removed, the query will return incorrect results.

- There's two ways to resolve a conflict:
  - Kill the read-only transaction
  - Pause WAL replay until query finishes

Avoiding Hot Standby conflicts

EnterpriseDB®
The Enterprise Postgres Company

- `vacuum_defer_cleanup_age` lets you defer vacuum of recently-deleted/updated tuples in the master.
- Reduces the chance of conflicts in standby
- Causes some bloat in master

Conflict resolution in standby

EnterpriseDB®
The Enterprise Postgres Company

- max_standby_delay = -1
  - Wait until offending query/transaction finishes.
  - Suitable for reporting servers, where queries are more important than staying up-to-date with master
- max_standby_delay = 0
  - Kill offending query/transaction
  - Suitable for High-Availability standby servers, where staying up-to-date is important

- Either way, **monitor** the lag between master and standby, and the number of killed queries

Monitoring

EnterpriseDB®
The Enterprise Postgres Company

pg_last_xlog_receive_location()

- How much WAL have we safely received from master?
- Determines how much data you will lose if the master dies

pg_last_xlog_replay_location()

- How far have we replayed?
- Determines how old data the results to read-only queries are based on

Failover

- Failover can be triggered by creating the trigger file specified in recovery.conf:

  trigger_file='/tmp/standby-trigger'

- New timeline is created
- Existing read-only connections stay connected, and become normal read-write connections

Failover considerations

- Designed to be simple and integrated with 3rd party high availability tools
  - Heartbeat
  - Shoot The Other Node In The Head
- Need to restore from base backup to make the old master as a standby

Security considerations

- All normal libpq authentication methods available
  - SSL
  - Certificate based authentication
- Streaming replication requires a user with superuser privileges
- Edit pg_hba.conf to control access

Example

EnterpriseDB®
The Enterprise Postgres Company

- pg_hba.conf:

```
# TYPE  DATABASE        USER            CIDR-ADDRESS          METHOD

host    replication     rep_user        192.168.1.117         md5
host    replication     all             0.0.0.0/0             reject

# "local" is for Unix domain socket connections only
local   all             all                                   trust
# IPv4 local connections:
host    all             all             127.0.0.1/32          trust
# IPv6 local connections:
host    all             all             ::1/128               trust
```

- psql:

```
CREATE USER rep_user SUPERUSER PASSWORD 'strong'
```

Under the hood

EnterpriseDB®
The Enterprise Postgres Company

- Streaming replication is implemented with two new server processes:
  - Walsender in master
  - Walreceiver in standby

```
$ ps ax
28016 ?        Ss     0:00 postgres: wal receiver
process   streaming 0/7000574
28017 ?        Ss     0:00 postgres: wal sender
process rep_user 127.0.0.1(33104) streaming
0/7000574
```

Protocol

- Walreceiver process in standby connects using libpq
  - Using replication=on
- Master launches a walsender process to serve the connection, instead of a regular backend.
- Walsender accepts a small set of special replication related commands:

- IDENTIFY_SYSTEM
  - Server replies with TLI and system identifier
- START_REPLICATION XXX/XXX
  - Server goes into COPY OUT mode, and starts to stream WAL

Future

- Allow base backup to be taken and transferred through the streaming connection
- Synchronous mode
- Cascading slaves
- Archiving from slave
- Stand-alone tools to stream WAL or take a base backup