PostgreSQL 9.0 Streaming Replication under the hood

Heikki Linnakangas

Copyright 2009 EnterpriseDB Corporation. All rights Reserved.

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

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Existing solutions

EnterpriseDB®
The Enterprise Postgres Company

- pg_standby
  - WAL File-based
  - Have to wait for full segment to fill up
- Slony
  - Trigger-based

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Record-based log-shipping < 9.0

EnterpriseDB®
The Enterprise Postgres Company

- “PostgreSQL directly supports file-based log shipping as described above. It is also possible to implement record-based log shipping, though this requires custom development.” - PostgreSQL user manual
- Poll using pg_xlogfile_name_offset()
- Skytools

- Doesn't work nicely with Hot Standby

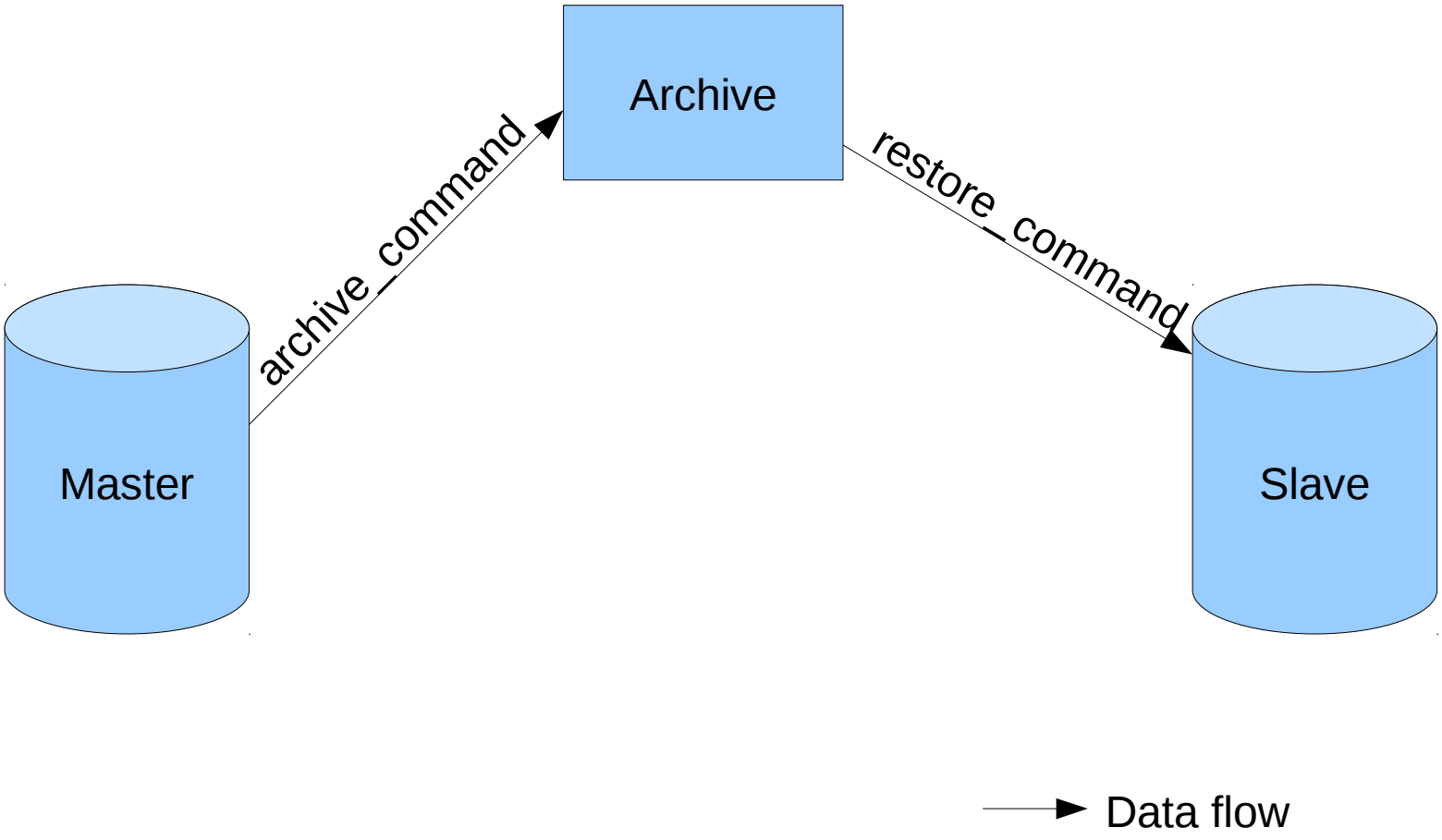
pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Streaming Replication

- WAL records are shipped from master as they're generated
- Similar to record-based log shipping in previous versions:
  - Asynchronous
  - Needs file-based archiving too
- But:
  - No custom development required
  - Works well with Hot Standby

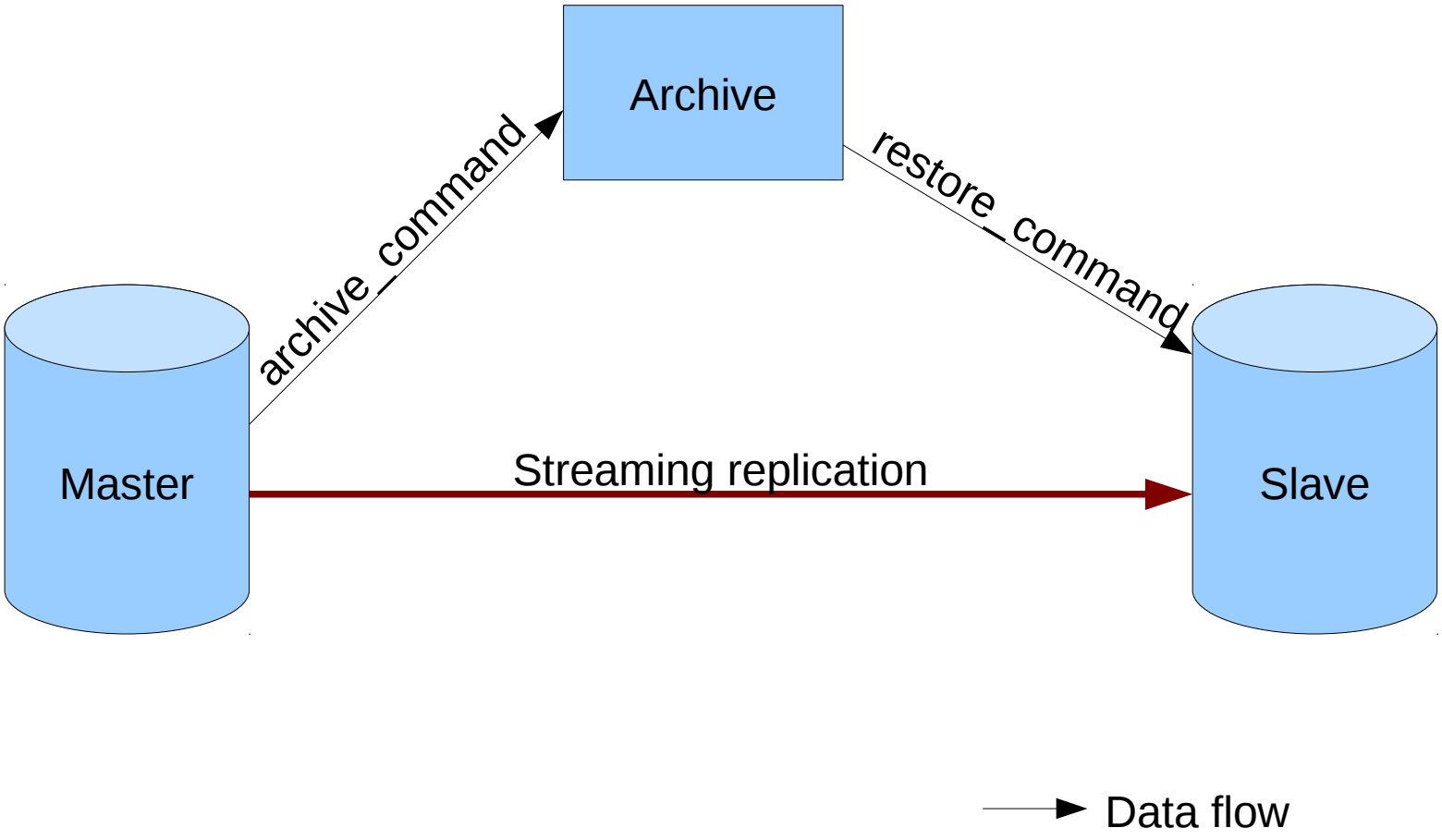
pyright 2009 EnterpriseDB Corporation. All rights Reserved.

File-based replication



pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Streaming replication



1. **Set up master and WAL archiving**
2. Set up standby server

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Set up master

EnterpriseDB®
The Enterprise Postgres Company

```
$:~/pgsql.cvshead$ bin/initdb -D data-master
The files belonging to this database system will be
owned by user "hlinnaka".
This user must also own the server process.

...

Success. You can now start the database server using:

    bin/postgres -D data-master
or
    bin/pg_ctl -D data-master -l logfile start
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Configure master

EnterpriseDB®
The Enterprise Postgres Company

- Open `postgresql.conf` in master
- Enable archiving:

```
archive_mode = on
archive_command = 'cp -i %p
/home/hlinnaka/pgsql.cvshead/walarchive/%f < /dev/null'
```

- Enable replication connections from standby:

```
max_wal_senders = 5
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Start master

EnterpriseDB®
The Enterprise Postgres Company

- Start master and take base backup:

```
$ bin/pg_ctl start -D data-master
server starting
$ bin/psql postgres -c "SELECT pg_start_backup('mybackup', true)"
 pg_start_backup
-----------------
 0/1000020
(1 row)

$ cp -a data-master/ data-standby
$ bin/psql postgres -c "SELECT pg_stop_backup()"
 pg_stop_backup
----------------
 0/10000D0
(1 row)
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough

1. Set up master and WAL archiving
2. **Set up standby server**

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Prepare standby

EnterpriseDB®
The Enterprise Postgres Company

- Clean up clutter left over from master:

```
$ rm data-standby/postmaster.pid data-standby/pg_xlog/*
```

- Change port in postgresql.conf:

```
port = 5433
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Prepare standby

EnterpriseDB®
The Enterprise Postgres Company

- Create `data-standby/recovery.conf`:

```
restore_command = 'cp
/home/hlinnaka/pgsql.cvshead/walarchive/%f
%p'
standby_mode = 'true'
primary_conninfo = 'host=localhost
port=5432'
trigger_file='/tmp/standby-trigger'
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Start standby

EnterpriseDB®
The Enterprise Postgres Company

- **$ bin/postmaster -D data-standby**

```
LOG:  database system was interrupted; last known up at 2010-02-03 16:34:24 EET
LOG:  starting archive recovery
LOG:  restore_command = 'cp /home/hlinnaka/pgsql.cvshead/walarchive/%f %p'
LOG:  standby_mode = 'true'
LOG:  primary_conninfo = 'host=localhost port=5432'
LOG:  trigger_file = '/tmp/standby-trigger'
LOG:  restored log file "000000010000000000000003" from archive
LOG:  automatic recovery in progress
LOG:  initializing recovery connections
LOG:  redo starts at 0/3000020
LOG:  consistent recovery state reached at 0/4000000
LOG:  database system is ready to accept read only connections
LOG:  restored log file "000000010000000000000004" from archive
LOG:  restored log file "000000010000000000000005" from archive
cp: cannot stat
`/home/hlinnaka/pgsql.cvshead/walarchive/000000010000000000000006': No such
file or directory
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Standby running

EnterpriseDB®
The Enterprise Postgres Company

```
$ ps ax| grep postgres
17451 pts/4    S      0:00 /home/hlinnaka/pgsql.cvshead/bin/postgres -D data-master
17455 ?        Ss     0:00 postgres: writer process
17456 ?        Ss     0:00 postgres: wal writer process
17457 ?        Ss     0:00 postgres: autovacuum launcher process
17458 ?        Ss     0:00 postgres: archiver process   last was 000000010000000000000005
17459 ?        Ss     0:00 postgres: stats collector process
17573 ?        Ss     0:00 postgres: startup process   recovering 000000010000000000000006
17576 ?        Ss     0:00 postgres: writer process
17578 ?        Ss     0:00 postgres: stats collector process
17584 ?        Ss     0:00 postgres: wal receiver process   streaming 0/6014708
17585 ?        Ss     0:49 postgres: wal sender process hlinnaka 127.0.0.1(56056)
streaming 0/6014708
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Standby running

EnterpriseDB®
The Enterprise Postgres Company

Master

```
$ psql postgres
psql (8.5devel)
Type "help" for help.

postgres=# CREATE TABLE foo (id
int4);
CREATE TABLE
```

Standby

```
$ PGPORT=5433 psql postgres
psql (8.5devel)
Type "help" for help.

postgres=# SELECT * FROM foo;
 id
----
(0 rows)
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: It works!

EnterpriseDB®
The Enterprise Postgres Company

Master

```
postgres=# INSERT INTO foo VALUES
(1);
INSERT 0 1
```

Standby

```
postgres=# SELECT * FROM foo;
 id
----
  1
(1 row)
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Walkthrough: Summary

EnterpriseDB®
The Enterprise Postgres Company

1. Set up master and WAL archiving
    1. Set up Continuous WAL Archiving
    2. Set max_wal_senders in postgresql.conf
    3. Start up and take base backup

2. Set up standby server
    1. Restore base backup
    2. Create recovery.conf
    3. Start standby

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

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

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Failover

EnterpriseDB®
The Enterprise Postgres Company

```
LOG:  trigger file found: /tmp/standby-trigger
FATAL:  terminating walreceiver process due to administrator
command
LOG:  redo done at 0/60147C0
LOG:  last completed transaction was at log time 2000-01-01
02:21:05.209196+02
cp: cannot stat
`/home/hlinnaka/pgsql.cvshead/walarchive/000000010000000000000006':
No such file or directory
cp: cannot stat
`/home/hlinnaka/pgsql.cvshead/walarchive/00000002.history': No such
file or directory
LOG:  selected new timeline ID: 2
LOG:  archive recovery complete
LOG:  autovacuum launcher started
LOG:  database system is ready to accept connections
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

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

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

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

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

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'
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Advanced: Look, no archive!

EnterpriseDB®
The Enterprise Postgres Company

- WAL archiving is still required
  - To take an online backup
  - To allow standby to catch up if the connection is lost for a long time
- You **can** do streaming replication without a shared archive, if you don't care about the above

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Advanced: replacing pg_standby

EnterpriseDB®
The Enterprise Postgres Company

- You can set standby_mode='true', without streaming replication
- WAL file-based replication, like pg_standby, but without pg_standby

```
restore_command = 'cp
/home/hlinnaka/pgsql.cvshead/walarchive/%f
%p'
standby_mode = 'true'
trigger_file='/tmp/standby-trigger'
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

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
```

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Protocol

- Walreceiver process in standby connects using libpq
  - Using replication=on
- Primary server 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

pyright 2009 EnterpriseDB Corporation. All rights Reserved.

Future

- Allow base backup to be taken and transferred through the streaming connection
- Allow reliable operation without archive
- Synchronous mode
- Cascading slaves
- Archiving from slave
- Stand-alone tools to stream WAL or take a base backup

pyright 2009 EnterpriseDB Corporation. All rights Reserved.