The PostgreSQL Replication Protocol

Tools and opportunities

Postgres Open 2011
Chicago, IL

Magnus Hagander
magnus@hagander.net
PostgreSQL Replication

- Added in PostgreSQL 9.0
- Based on streaming WAL (Transaction Log)
- Starts from *base backup*
- Uses standard recovery code
- Layered on top of regular protocol
Parts of the puzzle

- Connection processing and startup
- The PostgreSQL protocol
- The replication specific protocol
- pg_basebackup
Normal client connection

1. TCP connection established (5432)
Normal client connection

1. TCP connection established (5432)
2. fork()
Normal client connection

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
Normal client connection

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
4. Get database/username/options
Normal client connection

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
4. Get database/username/options
5. Perform authentication
Normal client connection

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
4. Get database/username/options
5. Perform authentication
6. Select database
Normal client connection

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
4. Get database/username/options
5. Perform authentication
6. Select database
7. Enter query processing loop
Replication client

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
4. Get database/username/options (fixed)
5. Perform authentication
6. Select database
7. Enter query processing loop
Replication client

1. TCP connection established (5432)
2. fork()
3. SSL negotiation
4. Get database/username/options
5. Perform authentication
6. Start **walsender**
What's the walsender?!

- Special purpose PostgreSQL backend
- Not connected with a database
- Only accepts simple queries
- Returns mix of resultsets and streams
- 9.0: only basic log streaming
  - Client connects, requests WAL streaming starting at position <x>
The PostgreSQL protocol

- Very simple
- Always TCP
- Message-based, bi-directional
- Optionally SSL encrypted
  - Entire stream wrapped
A message

| Message Type (byte) | Message Length (32-bit) | Message... |
Standard query exchange

ReadyForQuery

Z <size> <Transaction Status>

Q 13 SELECT 1,2,3

SimpleQuery

RowDescription

T <size> <col1>,<col2>,<col3>

DataRow

D <size> 1,2,3

CommandTag

C <size> SELECT
Streaming replication

- ReadyForQuery
- START_REPLICATION 0/0
- SimpleQuery
- CopyOutResponse
- W 0,0
- CopyData
- d <xlog data>
- CopyData
- d <xlog data>
Advances in 9.1

- Synchronous replication
  - (not going to cover that)
- Hot Standby Feedback Loop
  - (not going to cover that)
- Walsender “micro language”
Walsender micro-language

- Full grammar in walsender mode
- Few commands, few options
- Still very picky about formats
- Not designed for manual consumption
- Foundation for future improvements
Walsender in 9.1

- IDENTIFY_SYSTEM
- START_REPLICATION <position>
- BASE_BACKUP
  [LABEL 'label']
  [PROGRESS]
  [FAST]
  [WAL]
  [NOWAIT]
Base backups

- Single-command base backups
- No need for separate pg_start_backup()/pg_stop_backup()
  - Can still control backup label
  - Can still control fast/slow checkpoint
- Not a silver bullet
  - Old method is still there!
Base backups

- Still not for manual consumption
- Use `bin/pg_basebackup`
- Integration in third party modules and applications
Streaming base backups

- Tar format stream
  - Easy to stream
  - No global archive header
  - Alignment-at-512-bytes cheap
- One tar stream per tablespace
- Sequential transmission
Streaming base backups

- **ReadyForQuery**
- **Q** 24  BASE_BACKUP LABEL 'foo'
- **RowDescription**
- **DataRow**
- **CommandTag**

**SimpleQuery**

**Z** <size> <Transaction Status>

**T** <size> spcoid, spclocation, size

**D** <size> <tablespace information>

**C** <size> SELECT
Streaming base backups

CopyOutResponse

CopyData

CopyData

CopyDone

CopyOutResponse
Using pg_basebackup

- pg_basebackup
  -D <directory>
  -F<p|t>
  -c <fast|spread>
  -l <label>
  -Z

- Plus all “standard” libpq client options
Progress reporting

- Add -P to the commandline
- Expensive!
  - Scans all tablespaces twice
- Inexact – but gives a good hint
Base backups and WAL

- Restore from base backup requires WAL archiving
  - Complex to set up and monitor
- Append WAL to command, or use -x
- walsender includes required WAL files at end of tar file
- Use wal_keep_segments!
Future improvements
Streaming WAL archive

- Log archiving still uses archive_command
- 16Mb-blocks, or archive_timeout
- Replication protocol already does this
- `pg_receivexlog`
Prevent WAL cycling

- WAL cycled normally during backups
- In `-x` mode, might still be needed
- If cycled too soon, backup fails
WAL streaming during backup

- Combine streaming wal archive with `pg_basebackup`
- During backup, log is streamed in parallel
- Less WAL to keep on master
Relocatable tablespaces

- Currently, only $PGDATA can be moved
- In theory...
- Support moving other tablespaces
- Both for streaming and regular base backups!
Incremental backups

- “rsync” style?
- Using LSN?
- Decrease size of log archive without more full backups
Thank you!

Questions?

Twitter: @magnushagander
http://blog.hagander.net/
magnus@hagander.net