Replikation mit PostgreSQL 9.0

FrOSCon 2010 – 21.-22.08.2010

Andreas 'ads' Scherbaum

Web: http://andreas.scherbaum.la/
E-Mail: andreas[at]scherbaum.biz
PGP: 9F67 73D3 43AA B30E CA8F 56E5 3002 8D24 4813 B5FE

21.-22.08.2010

Was ist PostgreSQL?

- Relationale Datenbank unter BSD Lizenz
- Weltweit aktive Community
- Zahlreiche Features und Funktionen (Foreign Keys, Transaktionen, Trigger)
- Läuft auf zahlreichen Betriebssystemen und diverser Hardware
- Weitgehendes Einhalten der SQL-Standards – Dokumentation der Abweichungen
- Im Schnitt pro Jahr ein Minor-Release mit neuen Features
- Version 9.1 wird derzeit fleißig entwickelt

Der Dozent

- Name: Andreas Scherbaum
- Selbstständig im Bereich Datenbanken, Linux auf Kleingeräten, Entwicklung von Webanwendungen
- Arbeit mit Datenbanken seit 1997, mit PostgreSQL seit 1998 oder 1999
- Gründungsmitglied der Deutschen und der Europäischen PostgreSQL User Group
- Board of Directors – European PostgreSQL User Group

Was bedeutet dieses komische .la?

`http://andreas.scherbaum.la/`

- `.la` ist die TLD von Laos
- `.la` wird von Los Angeles genutzt und verwaltet
- `LA` ist das KFZ-Kennzeichen von Landshut/Niederbayern
- Dort habe ich längere Zeit gewohnt und meine Frau fand die Domain schön

Ziele dieses Workshops

- Installation von PostgreSQL 9.0beta4 (automatisiert)
- Konfiguration der Master Datenbank
- Erstellen eines Base Backups der Master Datenbank
- Einspielen des Base Backups als Slave
- Konfiguration der Slave Datenbank und der Replikation
- Starten des Slave
- Testen der Replikation

Datenbank kompilieren

- Unter folgender URL gibt es ein Paket zum Workshop
- http://andreas.scherbaum.la/writings/froscon-2010-skripts.tar.bz2

Datenbank kompilieren

- Auspacken mit:

Beispiel (Entpacken)

```
cat froscon-2010-skripts.tar.bz2 | bunzip2 | tar x
cd build/
```

Beispiel (Kompilieren)

```
make build
```

Datenbank kompilieren

- Im Verzeichnis `install` wird PostgreSQL 9.0beta4 installiert
- Im Verzeichnis `master/data` und `slave/data` wird jeweils eine Datenbank initialisiert

- Beide Datenbanken bleiben weitgehend unkonfiguriert
- Nur Änderungen für die Replikation werden vorgenommen

Streaming Replication + Hot Standby

- PostgreSQL 9.0 enthält Streaming Replication
- PostgreSQL 9.0 enthält Hot Standby

Streaming Replication – der Master

- Archive Logs (16 MB) werden wie gehabt zum Slave übertragen
- Zusätzlich pollt der Slave aktuelle Änderungen vom Master
- Bei zu großem Lag werden (zuerst) die Logfiles eingespielt

- Im Master wird weiterhin der `archive_mode` aktiviert
- Die Anzahl Clients über `max_wal_senders = n` konfiguriert
- Die Verzögerung wird über `wal_sender_delay` in ms eingestellt
- `wal_level` muss auf `archive` oder besser `hot_standby` gesetzt werden

Streaming Replication – der Slave

- `hot_standby = on`
- Archivierung deaktivieren (`archive_mode`, `wal_level`, `max_wal_senders`)
- Ggf. Port ändern (wenn Master und Slave auf der gleichen Maschine laufen)

Hot Standby

- Der Slave kann für (lesende) Anfragen genutzt werden
- Transaktionen auf dem Slave sind komplett, aber ggf. einige Zeit hinter dem Master
- Einige Lockingmodi können verwendet werden, andere nicht
- Two-Phase Commits sind nicht möglich (erfordern einen WAL-Eintrag)
- Nicht möglich: `LISTEN`, `NOTIFY`, `UNLISTEN`

Hot Standby – Konfliktlösungen

- Der Slave wartet `max_standby_archive_delay` bzw. `max_standby_streaming_delay` Sekunden bei einem Konflikt
- Transaktionen und Abfragen, die einen Lock halten, werden beendet
- Idle Transaktionen werden beendet
- Konflikte durch Aufräumarbeiten (Entfernen veralteter Records auf dem Master) informieren die Anfrage, diese beendet sich selbst (ansonsten: falsche Ergebnisse)

Hot Standby – Failover

- Im Failover Fall (Slave beendet Recovery) bleiben alle Verbindungen bestehen
- Transaktionen werden schreibbar

Datenbank konfigurieren – Master

- Datei `master/data/postgresql.conf` im Editor öffnen
- Folgende Änderungen vornehmen:

Beispiel (Einstellungen im Master)

```
listen_addresses = 'localhost,127.0.0.1'
port = 5440
archive_mode = on
archive_command = '/bin/cp %p ../../wal/%f'
archive_timeout = 30
wal_level = hot_standby
max_wal_senders = 2
wal_sender_delay = 10ms
```

Streaming Replication – der Master – pg_hba.conf

- Benötigt (derzeit) einen Superuser Account

Beispiel (Superuser Zugang für Replikation – pg_hba.conf)

```
# TYPE   DATABASE      USER        CIDRADDRESS     METHOD
host     replication   postgres    127.0.0.1/32    trust
host     replication   all         0.0.0.0/0       reject
```

- Hinweis: als erste Zeilen eintragen
- Hinweis: ggf. eigenen Superuser für Replikation erstellen
- Hinweis: statt "postgres" ggf. anderen Nutzer (eigenen Unix Account) einsetzen

Datenbank starten – Master

- Den Master starten
- Hinweis: eigene Shell/Konsole/Terminal verwenden

Beispiel (Datenbank starten)

```
make master-db
```

Beispiel (Datenbankshell öffnen)

```
make master-sh
```

Datenbank verwenden

- Einige schreibende Änderungen durchführen
- Z. B. eine Tabelle erstellen und mit Daten befüllen
- Danach die WAL-Datei wechseln:

Beispiel (neue WAL-Datei beginnen)

```
SELECT pg_switch_xlog();
```

- Im Verzeichnis `wal` sollten sich nun Dateien befinden

Slave erstellen

- Der Slave wird wie bei PITR mit einem Online-Backup aufgesetzt
- Der Master wird mittels `pg_start_backup()` und `pg_stop_backup()` in den Online-Backup Modus versetzt
- Der Slave darf noch nicht laufen!
- Im Workshop Paket befindet sich ein entsprechendes Makefile Ziel

Beispiel (Slave erstellen)

```
make backup
```

Slave – Hinweis

- Hinweis: `pg_standby` kann nicht mehr verwendet werden!

Datenbank konfigurieren – Slave

- Datei `slave/data/postgresql.conf` im Editor öffnen
- Folgende Änderungen zurücksetzen bzw. vornehmen:

Beispiel (Slave konfigurieren)

```
port = 5441
archive_mode = off
#archive_command = ''
#archive_timeout = 30
```

Beispiel (Slave konfigurieren)

```
hot_standby = on
```

Streaming Replication – der Slave

- Normale Konfiguration in `recovery.conf`
- Zusätzlich: `standby_mode = true`
- Zusätzlich: `primary_conninfo = 'host=masterdb port=5432'`
- Zusätzlich: `trigger_file = /tmp/trigger.hs`

Streaming Replication – der Slave – Konfiguration

- `slave/data/recovery.conf` im Editor erstellen

Beispiel (Beispielkonfiguration)

```
restore_command = 'cp ../../wal/%f %p'
standby_mode = 'on'
primary_conninfo = 'host=127.0.0.1 port=5440 user=postgres'
trigger_file = '/tmp/replikation.trigger'
```

- Hinweis: statt "postgres" ggf. anderen Nutzer (eigenen Unix-Account) einsetzen

Datenbank starten – Slave

- Den Slave starten
- Hinweis: eigene Shell/Konsole/Terminal verwenden

Beispiel (Datenbank starten)

```
make slave-db
```

Beispiel (Datenbankshell öffnen)

```
make slave-sh
```


Datenbank verwenden

- Änderungen auf dem Master vornehmen
- Änderungen sofort danach auf dem Slave betrachten
- Schreibende Änderungen auf dem Slave versuchen

Hot Standby – Hilfsfunktionen

- `pg_is_in_recovery()`: liefert true auf einem Standby-System zurück

Beispiel (pg_is_in_recovery())

```
postgres=# SELECT pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 t
(1 Zeile)
```


Hot Standby – Hilfsfunktionen

- `pg_last_xlog_receive_location()`: liefert den letzten empfangenen WAL-Record

Beispiel (pg_last_xlog_receive_location())

```
postgres=# SELECT pg_last_xlog_receive_location();
 pg_last_xlog_receive_location
-------------------------------
 0/DD000000
(1 Zeile)
```

Hot Standby – Hilfsfunktionen

- `pg_last_xlog_replay_location()`: liefert den letzten eingespielten WAL-Record

Beispiel (pg_last_xlog_replay_location())

```
postgres=# SELECT pg_last_xlog_replay_location();
 pg_last_xlog_replay_location
------------------------------
 0/DD000000
(1 Zeile)
```


Hot Standby – Nacharbeiten

- `max_locks_per_transaction` erhöhen, wenn `Prepared Transactions` verwendet werden
- `max_connections`, `max_prepared_transactions` und `max_locks_per_transaction` müssen auf dem Slave gleich oder größer als auf dem Master sein
- `vacuum_defer_cleanup_age` verzögert `VACUUM` auf dem Master und erhöht damit die Laufzeit für Anfragen auf dem Slave (alternativ: Anfrage rückwärts zum Master)

Wichtiger Termin

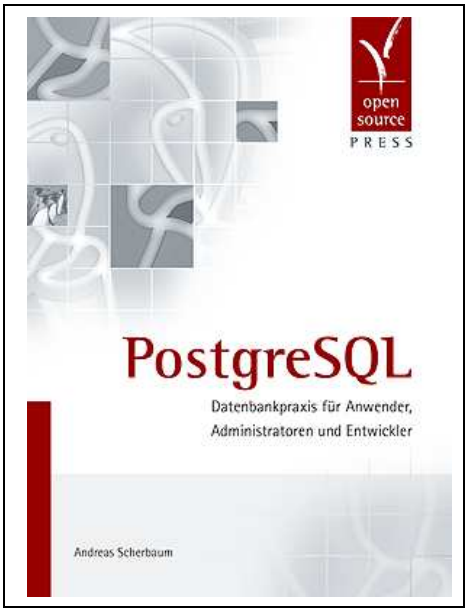
Wichtiger Termin

`PGDay.EU 2010`

- 6.-8. Dezember
- in Stuttgart

Webseite: `http://www.pgday.eu/`

PostgreSQL Buch



`PostgreSQL` – Datenbankpraxis für Anwender, Administratoren und Entwickler

Erschien im Juli 2009 im Verlag Open Source Press
Umfang: ca. 520 Seiten

Ende

http://andreas.scherbaum.la/

Fragen?

Andreas 'ads' Scherbaum <andreas@scherbaum.biz>
PostgreSQL User Group Germany
European PostgreSQL User Group

PostgreSQL Service & Support