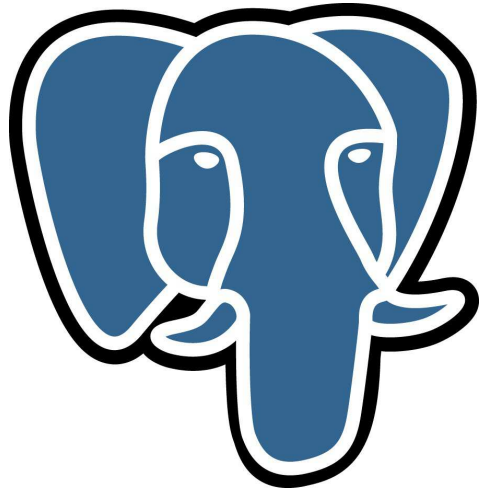
Replikace v PostgreSQL

CSPUG, Praha

Tomáš Vondra (tv@fuzzy.cz)

Czech and Slovak PostgreSQL Users Group



19.4.2011

Agenda

- Účely replikace
- Varianty replikace
- Historie replikace v PostgreSQL
- Zabudovaná replikace
- Externí nástroje
- Srovnání s dalšími DB

Účely replikace

- high-availability
- škálování výkonu (read vs. write)
  - load balancing
  - query partitioning
- migrace a upgrade bez výpadku
  - jiná verze stejné databáze
  - úplně jiná databáze
- rychlejší přístup přes WAN
  - lokální kopie na pobočkách
  - road warriors / kopie pro mobilní zařízení

Varianty replikace

- master-master vs. master-slave
- synchronní vs. asynchronní
- fyzická vs. logická
- warm standby vs. hot standby
- způsob implementace
  - xlog (log file shipping vs. streaming)
  - trigger
  - statement-based

Master vs. slave

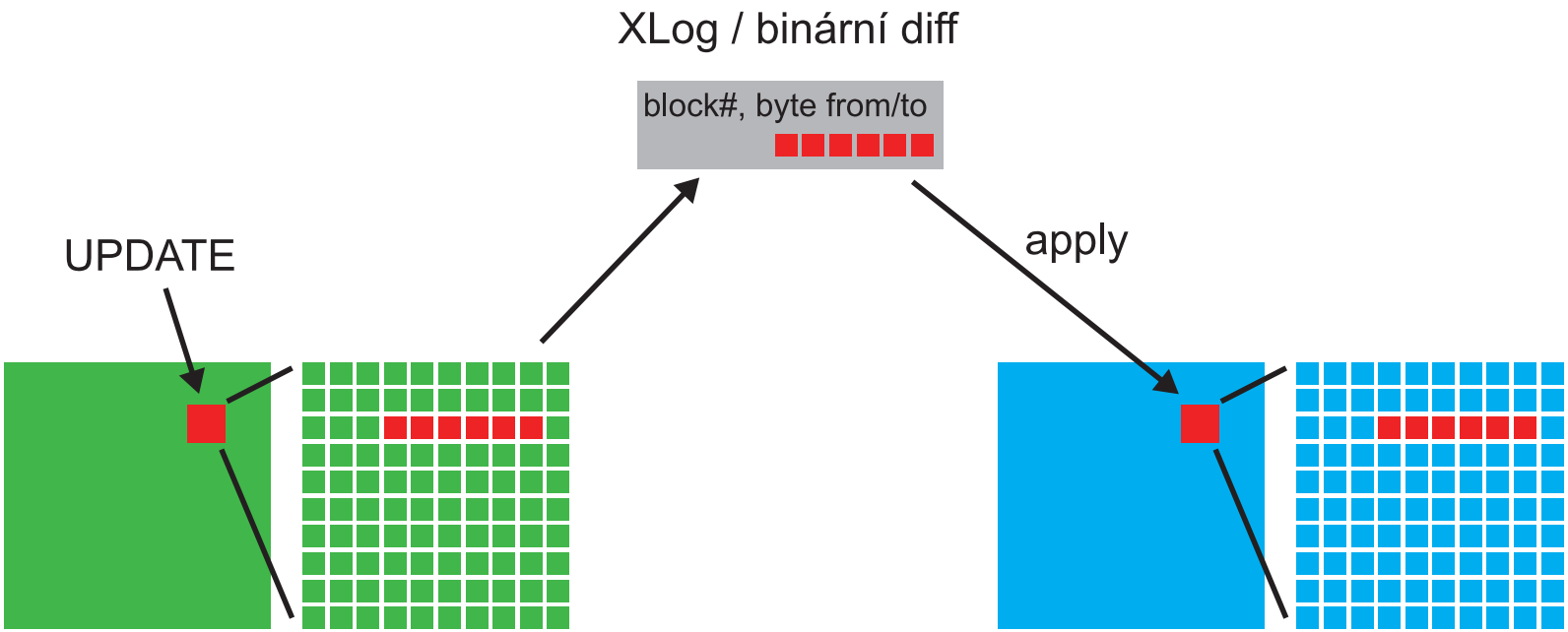
- **master**
  - autoritativní zdroj informací
  - zpracovává požadavky na změny, předává je dále
- **slave**
  - změny se přejímají z master databáze, jinak read-only
  - v podstatě jen "kopie" master databáze
- **master-slave**
  - jednodušší "jednosměrná" replikace
  - read scalability
- **master-master**
  - obousměrná replikace - nutno řešit kolize
  - write scalability

Synchronní vs. asynchronní replikace

- **synchronní**
  - commit čeká na potvrzení dokončení replikace
  - pomalejší, ale vždy konzistentní jako celek
- **asynchronní**
  - na dokončení replikace se nečeká, zapíše se jen lokálně
  - rychlejší, ale může dojít k nekonzistenci mezi originálem a replikou (replika je "pozadu")
- **semi-synchronní**
  - kompromis mezi spolehlivostí a výkonem
  - více replik, čeká se jen na potvrzení z první

Fyzická replikace

- binární kopie datových bloků (xlog)
- masteru do XLogu zapíše "binární diff" (změň byte X v bloku Y na Z)
- slave přečte a aplikuje na bloky (v podstatě recovery)

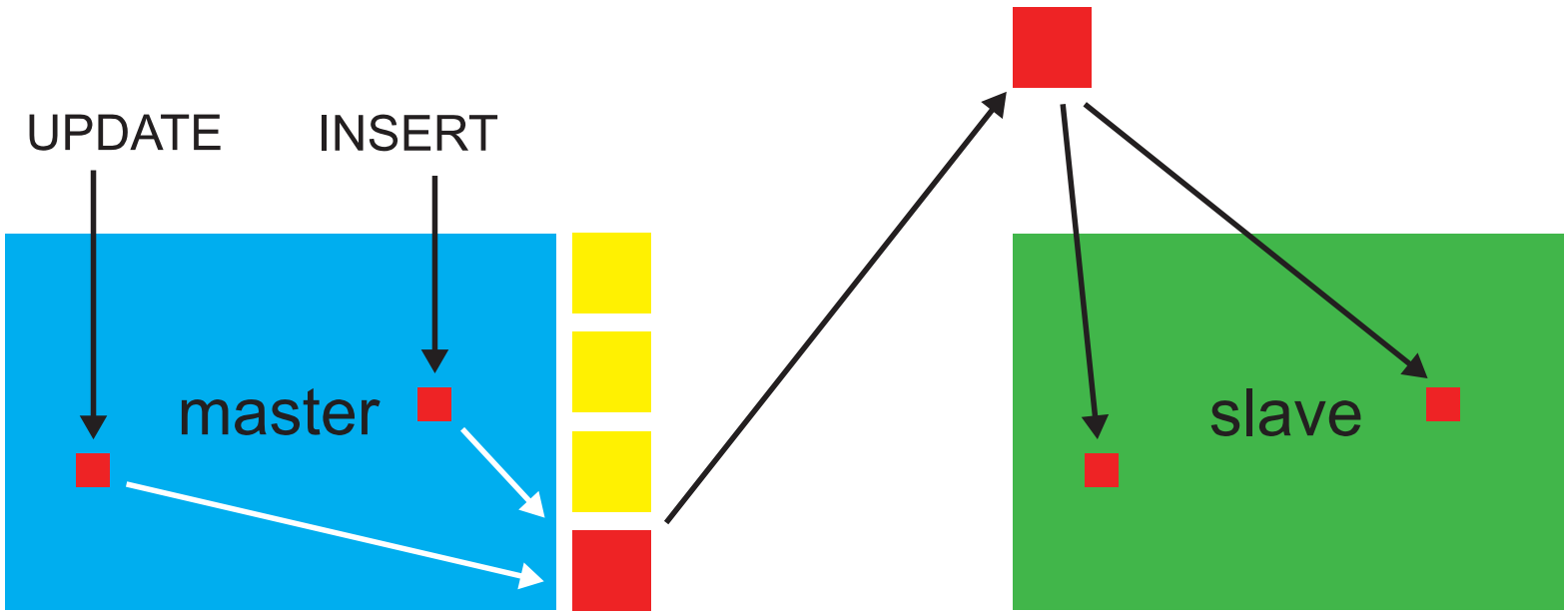


Fyzická replikace

- **klady**
  - minimální overhead (1%)
  - velmi jednoduché na nastavení
- **zápory**
  - jen kompletní databáze
  - stejná verze DB (formát)
  - stejný HW (zejména CPU architektura)
  - jen master-slave (nemožnost řešení konfliktů)

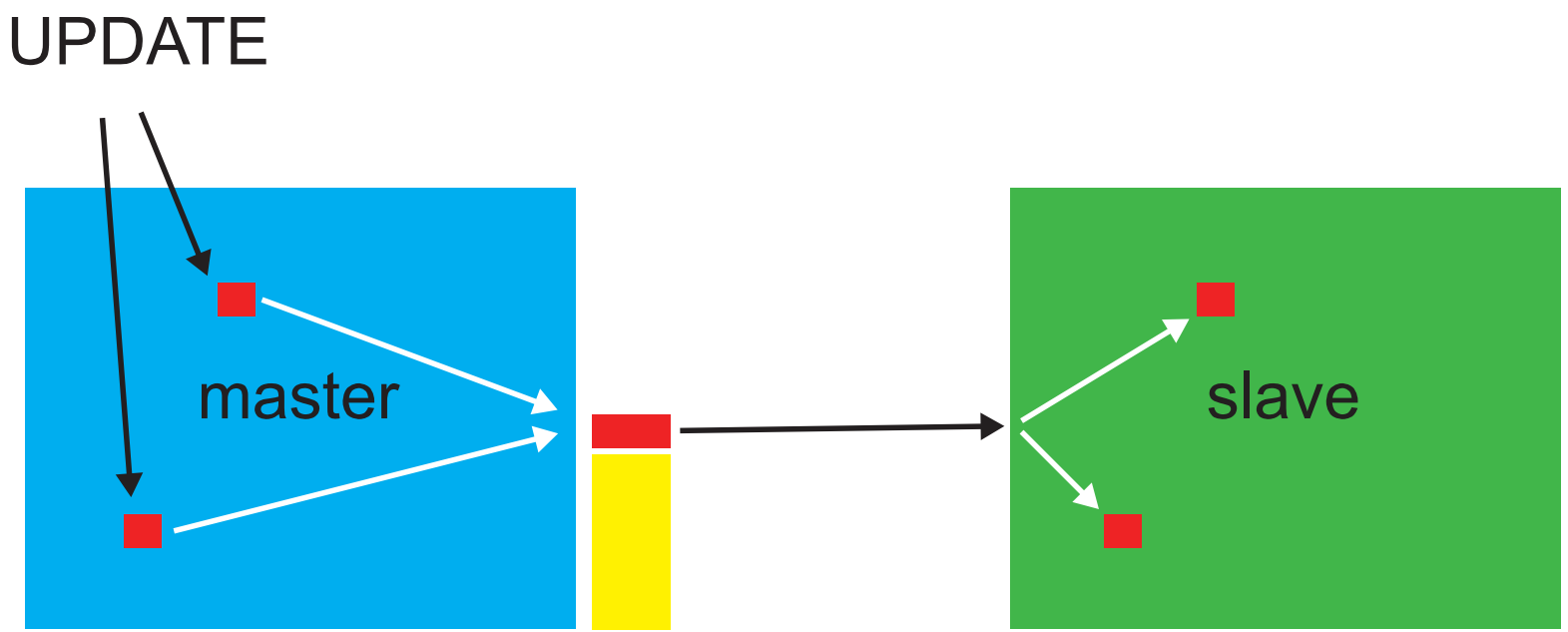
Fyzická replikace / Log file shipping

1. UPDATE znamená změnu několika datových bloků
2. každá změna generuje záznam v transakčním logu
3. při zaplnění segmentu (16MB) se archivuje (NFS, ...)
4. archivovaný segment se aplikuje na slave databázi
5. výsledkem je binární kopie



Fyzická replikace / Streaming replication

1. UPDATE znamená změnu několika datových bloků
2. každá změna generuje záznam v transakčním logu
3. záznamy se (asynchronně) přenáší do slave databáze
4. změny se aplikují
5. výsledkem je (opět) binární kopie



Warm standby vs. Hot standby

- **warm standby**
  - databáze nastartovaná v “recovery módu”
  - přijímá z master databáze změny a aplikuje je
  - slouží jen pro HA - nelze se připojit a spouštět dotazy
- **hot standby**
  - databáze nastartovaná v “read-only módu”
  - lze spouštět read-only dotazy (nedostane XID)

Logická replikace

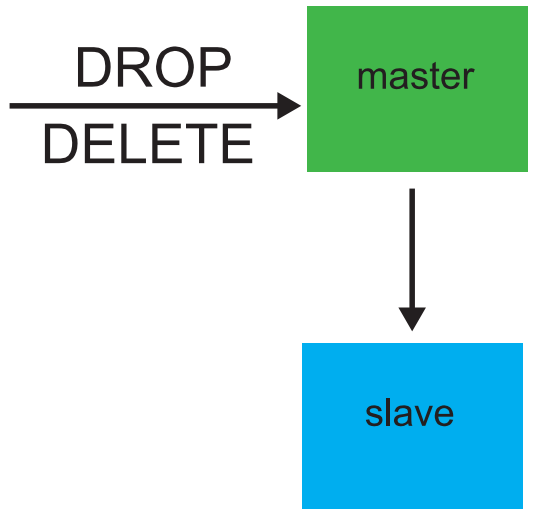
- ne aplikace binárních změn bez znalosti struktury dat
- aplikace logických operací (INSERT/UPDATE/DELETE)
- **klady**
  - replikace jen části databáze (např. jedna tabulka)
  - replikace do jiné verze / jiné DB (uprage a migrace)
  - umožňuje multi-master replikaci
- **zápory**
  - náročnější na nastavení i na zdroje
  - nutnost řešení konfliktů (specifické dle aplikace)

Logická replikace / způsoby implementace

- log všech SQL / opakovaná aplikace na repliku
- zpětná interpretace xlog záznamů
- triggery
- proxy zachycující SQL

Replikace **nenahrazuje** zálohování

- replikuje se všechno (včetně omylů)
- hacker, idiot, unit test omylem na produkční DB
- DROP DATABASE, DELETE, ...



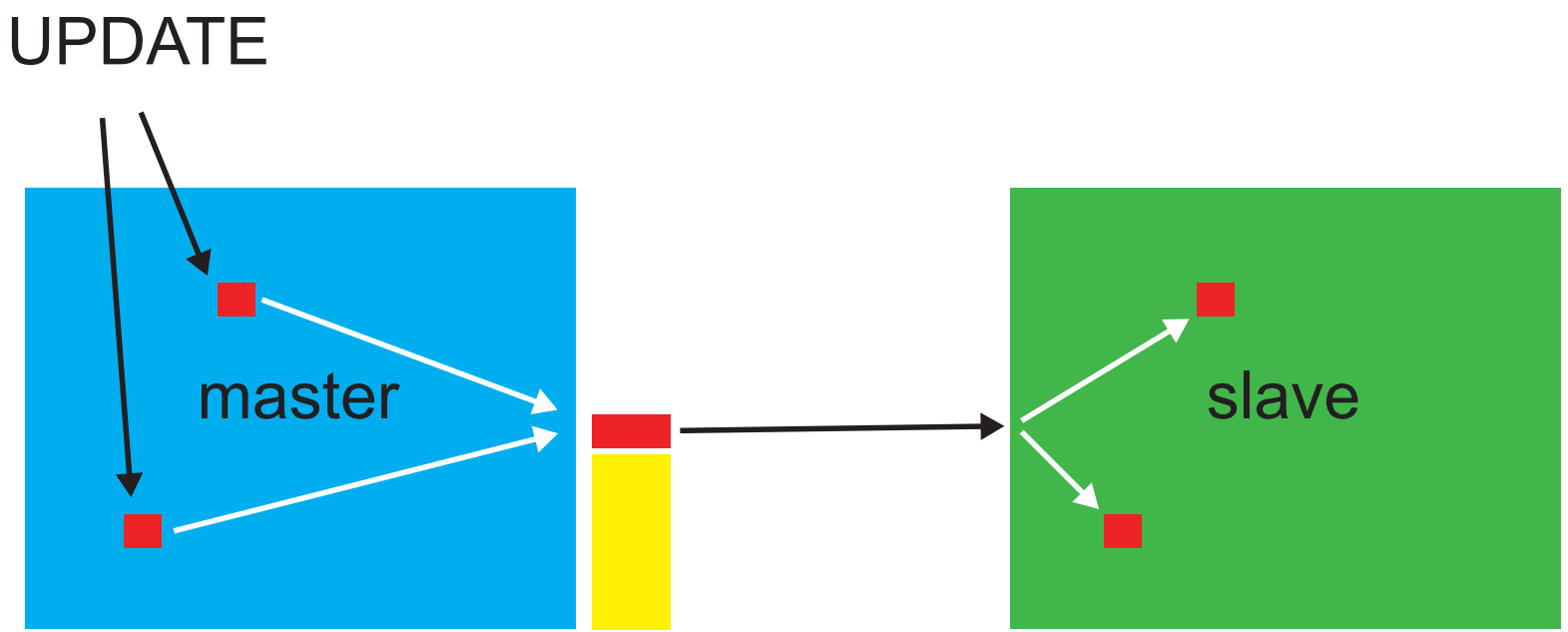
- Zálohujte! Zálohujte! Zálohujte!

Historie, současnost a budoucnost

Historie replikace v PostgreSQL

- "Unixová mentalita" core teamu
  - menší flexibilní nástroje, možnost kombinace
  - mnoho možností implementace - raději externě
  - odpor k přidávání takových vlastností
- **do verze 8.4** (včetně)
  - XLog file shipping replikace + "warm standby" (HA)
  - zajímavé externí nástroje (Bucardo, Londiste, slony-I, ...)
- **verze 9.0**
  - asynchronní XLog streaming replikace
  - možnost "hot standby"
- **verze 9.1**
  - synchronní XLog streaming replikace

Zabudovaná replikace



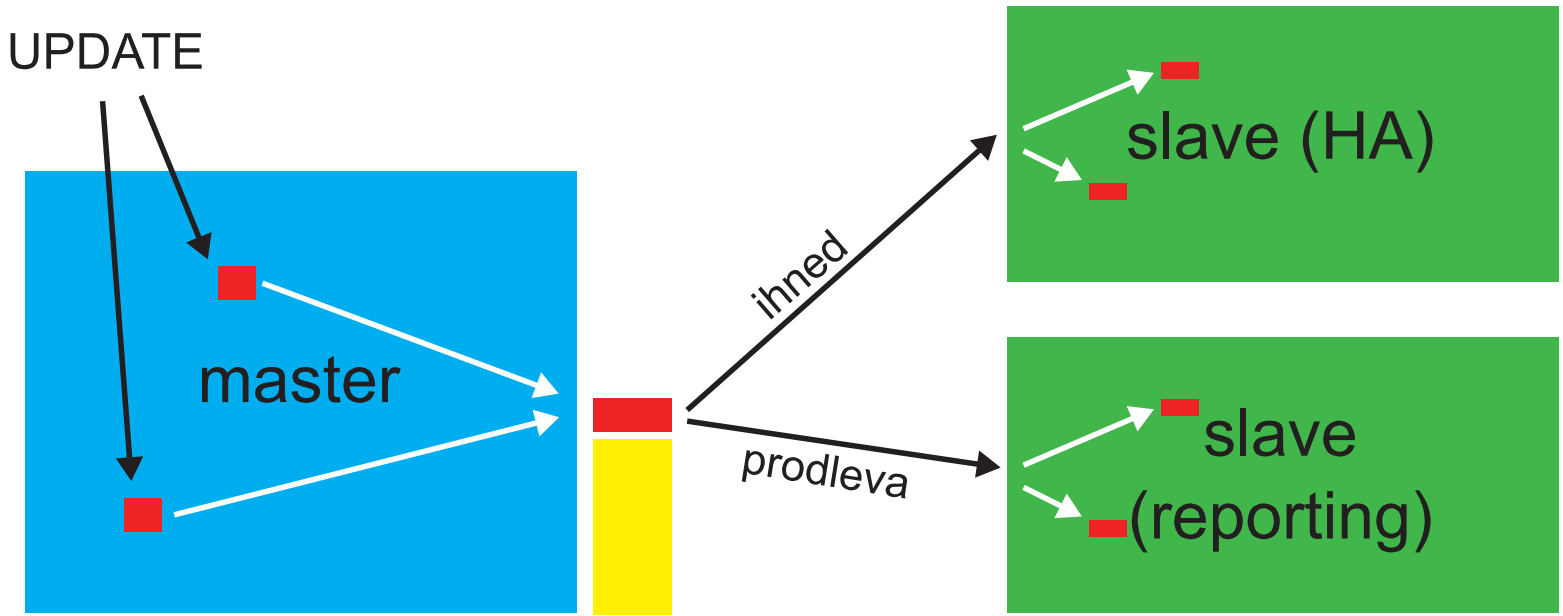
- fyzická (a)synchronní replikace
- rozpor mezi nároky na HA a reporting (zabíjení queries)
- trochu problém při úmrtí mastera (s více slavy)

HA vs. reporting / query cancellation

- **high-availability**
  - cílem je minimální delay oproti masteru (kvůli failoveru)
- **reporting**
  - dlouho běžící dotazy nad velkými datovými objemy
  - dotaz potřebuje blok který se změnil $\longrightarrow$ je zabit
  - rychlá aplikace změn $\longrightarrow$ větší pravděpodobnost zabití
- **zajímavá nastavení**
  - vacuum_defer_cleanup_age (master)
  - hot_standby_feedback (standby)

HA vs. reporting / query cancellation

- řešení - dva slaves, jeden pro HA a druhý pro reporting



Demo

1. vytvoříme a nakonfigurujeme mastera
2. vytvoříme slave, napojíme na master
3. provedeme něco na masterovi
4. podíváme se jak se to zpropagovalo na slave
5. zkusíme nějaké dotazy nad slave (read / write)
6. zastavíme mastera
7. provedeme failover

Demo / master

postgresql.conf

```
listen_addresses = '127.0.0.1'
port = 5432

# archivní režim
wal_level = hot_standby
max_wal_senders = 10

# archivní režim
archive_mode = on
archive_command = 'cp %p /var/pg9/archive/%f'
```

pg_hba.conf

```
# IPv4 local connections
host    replication  repuser  127.0.0.1/32  trust
```

Demo / slave

postgresql.conf

```
listen_addresses = '127.0.0.1'
port = 5433
hot_standby = on
```

recovery.conf

```
standby_mode = 'on'
primary_conninfo = 'host=127.0.0.1 port=5432 user=repuser'

# ukončení recovery (touch)
trigger_file = '/var/pg9/failover'

# načtení z archivu logů
restore_command = 'cp /var/pg9/archive/%f "%p"'
```

Nevýhody zabudované replikace

1. slave je jen pro čtení
   - nenaplníte si TEMP tabulku (problém pro reporting)
   - nenačtete hodnotu ze sekvence
   - nelze udělat standardní zálohu
2. ne úplně elegantní monitoring
   - lag replikace na slave se dá monitorovat přes “ps”
   - výrazně se zlepší ve verzi 9.1
3. nelze dělat kaskádu (všichni visí na jednom masterovi)

Externí nástroje

	typ	technika	M/M	M/S	sync	async
PostgreSQL 9.0	fyzická	xlog	ne	ano	ne	ano
PostgreSQL 9.1	fyzická	xlog	ne	ano	ano	ano
Londiste	logická	triggers	ne	ano	ne	ano
Bucardo	logická	triggers	ano	ano	ne	ano
slony-I	logická	triggers	ne	ano	ne	ano
pgpool-II	logická	proxy	ano*	ne*	ano	ne
Postgres-XC	cluster	-	ano	ne	ne	ano

* u proxy kategorie jako master nebo slave nemají úplně smysl

Londiste

- napsáno Skype, součást SkyTools (i další nástroje)
- implementováno v Pythonu (jako skoro vše ve Skype)
- PgQ - vlastní implementace fronty
- jen master/slave replikace (logická)
- http://wiki.postgresql.org/wiki/Londiste_Tutorial
- http://wiki.postgresql.org/wiki/Skytools

Bucardo

- http://bucardo.org/
- triggery a démon - implementováno v Perlu (PL/Perl)
- založeno na LISTEN/NOTIFY
  - transakční notifikace zabudované přímo do DB
  - jednoduchá komunikace sessions přes frontu
- nedokáže replikovat DDL (nejsou DDL triggery)
- master to master - aktuálně jen dva mastery
- master to many slaves

pgpool-II

- http://pgpool.projects.postgresql.org/
- používá proxy koncept (statement-based middleware)
- spojuje několik pokročilých vlastností
  - connection pooling
  - replikace (včetně online recovery)
  - load balancing (rozhazování queries na repliky)
  - parallel queries (distribuované tabulky)
- několik módů, ne vždy je možno vše (parallel vs. failover)
- pokud chcete HA řešení, jsou asi jednodušší nástroje

slony-I

- http://slony.info/
- master-slave replikace (max. 20 subscriberů)
- založeno na triggerech a C funkcích
- plusy
  - 5 let zkušeností z provozu
  - téměř kompletní řešení (failover, provisioning, ...)
- mínusy
  - fronta událostí je řešena přes tabulku (nutno VACUUM)
  - vyšší overhead než řešení s jinak řešenou frontou
  - komplexní - složité nastavení, obtížné řešení problémů

Oracle & MySQL

Jiné DB / Oracle

- **DataGuard**
  - používá XLog, dva módy - "Redo Apply" a "SQL Apply"
  - Redo Apply - fyzická replikace (= streaming replikace)
  - SQL Apply - logická replikace, obohacený XLog, různá omezení (ne všechny objekty, ne všechny datové typy)
  - Active Data Guard (další $) umožňuje "hot standby"
- **Streams**
  - logická replikace, postavená nad Advanced Queueing
  - obecně nástroj pro distribuci informací (ne jen replikace)
- **GoldenGate**
  - log-based logická replikace pro heterogenní prostředí (Oracle, DB2, MSSQL, MySQL, ...)
  - Oracle doporučuje jako náhradu za Streams

Jiné DB / MySQL

- asynchronní logická master-slave replikace (od 5.5 semi-synchronní)
- postaveno na tzv. “binlogu” (statement-based log)
- **statement-based (SBR)**
  - loguje kompletní SQL příkazy (které změnily data)
  - ne všechny SQL příkazy jsou “bezpečné”
- **row-based (RBR)**
  - logují se finální změny jednotlivých řádků
  - bezpečnější ale větší objem dat než SBR
- **mixed-based (MBR)**
  - SBR nebo RBR podle typu eventu
- MySQL Cluster (NDB engine) - synchronní replikace založená na 2PC (ne na binlogu)

Odkazy / obecné

- Replication @ wikipedia

  http://en.wikipedia.org/wiki/Replication_(computer_science)

- MySQL 5.5 Replication

  http://dev.mysql.com/doc/refman/5.5/en/replication.html

  http://dev.mysql.com/doc/refman/5.5/en/replication-sbr-rbr.html

  http://dev.mysql.com/doc/refman/5.5/en/replication-rbr-usage.html

- Drizzle

  http://docs.drizzle.org/replication.html

  http://code.google.com/p/protobuf/

- Oracle Data Guard

  http://en.wikipedia.org/wiki/Oracle_Data_Guard

- Oracle Streams

  http://www.oracle.com/technetwork/database/features/data-integration/default-159085.html

- Oracle GoldenGate

  http://www.oracle.com/technetwork/middleware/goldengate/overview/index.html

Odkazy / PostgreSQL

- Replication, Clustering, and Connection Pooling

  http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling

- Replication solutions for PostgreSQL (Peter Eisentraut)

  http://www.slideshare.net/petereisentraut/replication-solutions-for-postgresql

- 9.0 Streaming Replication vs Slony (Steve Singer)

  http://scanningpages.wordpress.com/2010/10/09/9-0-streaming-replication-vs-slony/

- PostgreSQL / WAL config

  http://www.postgresql.org/docs/current/static/runtime-config-wal.html

  http://developer.postgresql.org/pgdocs/postgres/runtime-config-wal.html

- PostgreSQL / Comparison of Different Solutions

  http://developer.postgresql.org/pgdocs/postgres/different-replication-solutions.html

- repmgr

  https://github.com/greg2ndQuadrant/repmgr