PostgreSQL upgrade včera, dnes a zítra

Zdeněk Kotala

Revenue Product Engineer

Sun Microsystems

Agenda

- Úvod
- Catalog upgrade
- Storage upgrade
- Ostatní

Úvod

Cíle

- Minimální výpadek
- Žádné další potřebné místo
- Bez potřeby staré verze
- Jednoduché k použití

Včera

- pg_dump/pg_restore
  - Dlouhá odstávka závislá na velikosti databáze
  - Nutné další místo na dump
  - Nevhodné pro velké databáze
  - Univerzální
- Slony
  - Možné upgradovat za chodu
  - Konfigurace není jednoduchá
  - Potřeba dalšího místa případně serveru

Dnes

- pg_migrator, pg_upgrade.sh
  - Pouze pro 8.1 na 8.2 nebo 8.3 na 8.4
  - Spíše dočasné řešení
  - Nedokáže upgradovat tabulky se smazanými sloupečky

Zítra

- Jednoduché “pg_ctl -D /var/postgres/ upgrade”.
- V některých případech bude nutné pustit pre_upgrade skript na staré verzi.
- Výpadek nutný k upgradu jen několik minut.
- První přístup k datům bude v některých případech pomalejší a bude generovat I/O.

Catalog upgrade

Co je katalog

- Control file
- Flat file
- Struktura adresářů
- Systémové tabulky (pg_catalog)
- Konfigurace

Současné řešení

- Pg_migrator nebo pg_upgrade.sh
  - Pouze pro 8.1->8.2, 8.3->8.4
  - Problém s tablespace – nutné udržet data na stejném svazku.
  - Problém s TOAST tabulkami (TOAST pointer)
  - Problém se zrušenými sloupečky v tabulkách
  - Závisí na privátním rozhraní

Jak pg_upgrade.sh pracuje*

1) Dump metadata
2) Uložení mapy relací (relfilenode<->name)
3) Export dat z control file
4) Nový databázový klastr (initdb)
5) Provede se zmražení databázového klastru
6) Kopie CLOG
7) Nastavení control file (XID,OID,XLOG ...)
8) Vytvoření databází, uživatelů ...

*Jednoduchá verze bez tablespaců

Jak pg_upgrade.sh pracuje

9) Ochrana TOAST tabulek (musí mít stejné relfilenode)
10) Vytvoření tabulek, pohledů ...
11) Upravení relfilenode pro TOAST tabulky a indexy
12) Kopírování a přejmenování datových souborů
13) Hotovo

Jak by měl upgrade vypadat

```
pg_ctl -D /var/postgres upgrade
```

Výstup při upgradu

```
check directory /var/postgres ... ok (version 822)
check subdirectories ... ok
creating template1 database in /tmp/pokus/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
upgrading pg_global database ... ok
upgrading template0 ... ok
upgrading postgres ... ok
upgrading super_db ... ok
```

Katalog

- Struktura
  - Pomocí postgres.bki se inicializuje template1 a pak se katalog rozkopíruje postupně do ostatních databází
  - Staré datové soubory katalogů je nutné uschovat, pro konverzi dat
- Obsah
  - Uživatelská metadata se převedou do nového formátu
  - Problém podchytit změny během vývoje, vyžaduje větší nároky na vývoj

Storage upgrade

Datové struktury

BLCKSZ

PageHeaderData

TOAST_MAX_CHUNK_SIZE

ItemIdData

*MaxItemSize

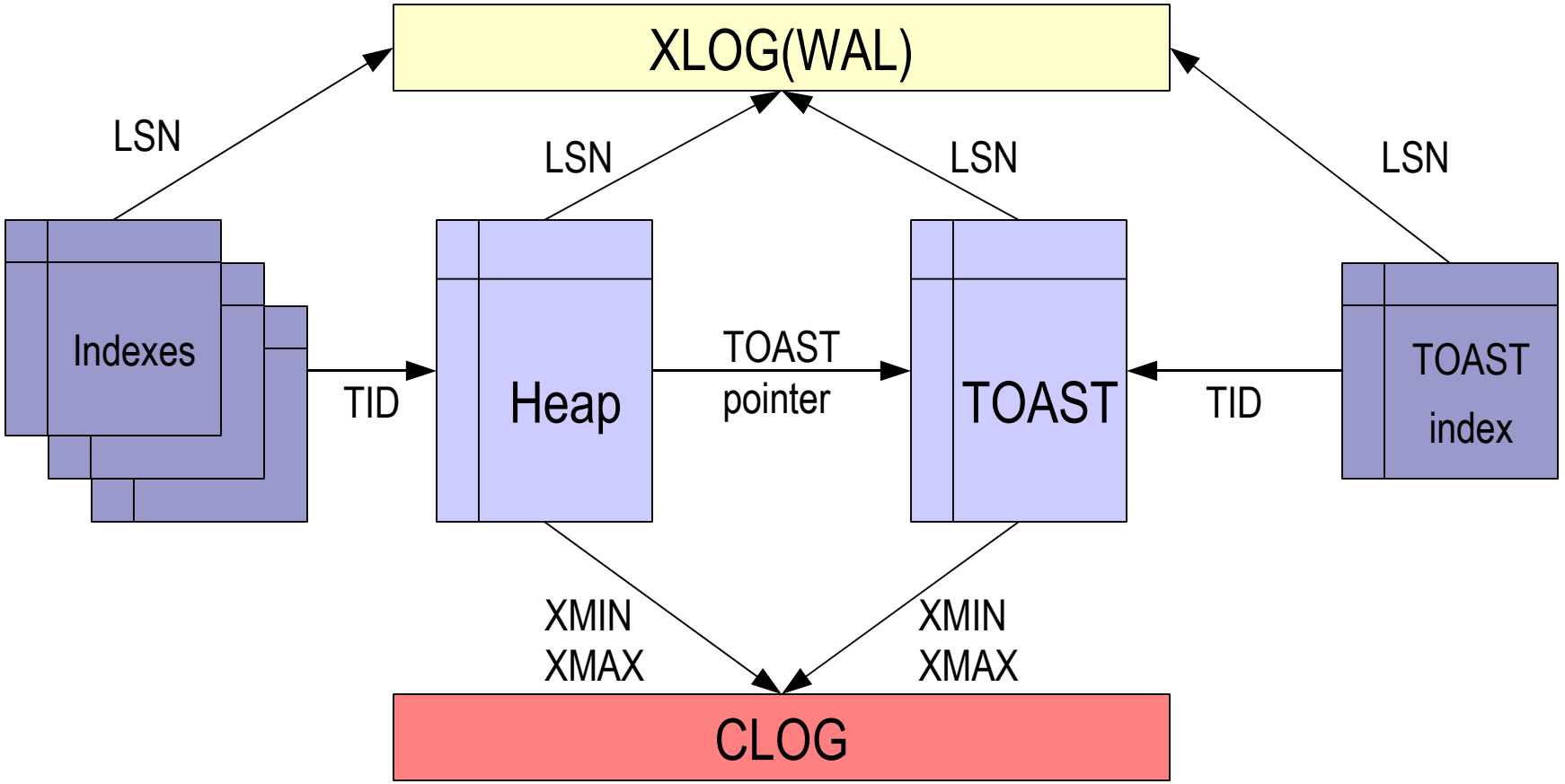
IndexTupleData

*OpaqueData

varatt*

HeapTupleHeaderData

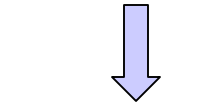
Graf závislosti datového úložiště



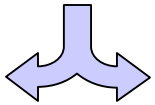
Problémy k řešení

- Žádná vazba NESMÍ být porušena!
- Všechny data musí být zkonvertovány při čtení
  - Zkonvertována data se musí vejít opět na stránku, v případě, že může dojít k této situaci, je nutné pustit preupgrade fázi na staré databázi, který zajistí místo
  - Pokud tuple obsahuje TOAST data, musí se provést také konverze těchto dat.
  - TOAST tabulky nejdou konvertovat přímo, poněvadž neobsahují informaci o datovém typu. Navíc jedna stránka může být sdílena více různými daty.

Jak to funguje



ReadBuffer() ⟺ smgr_rd()



Vytvoř novou stránku PageInit()

- Načti tuple ze staré stránky
- Zkonvertuj tuple
- Ulož tuple na novou stránku PageAddItem()

Nahraď starou stránku novou

Zapiš WAL

A jak vypada kód

- Konvertor “hook” v ReadBuffer_common

```
{
        smgrread(reln->rd_smgr, blockNum, (char *) bufBlock);
        /* Page Layout Convertor hook. We assume
          that page version is on same place. */
        if( plc_hook &&  PageGetPageLayoutVersion(reln,bufBlock)
                != PG_PAGE_LAYOUT_VERSION )
        {
                plc_hook(reln, (char *)bufBlock);
                bufHdr->flags |= (BM_DIRTY | BM_JUST_DIRTIED);
                log_newpage(&reln->rd_node, blockNum ,bufBlock);
        }
}
```

Space reservation

- Space reservation slouží k rezervaci volného místa, které bude nutné při konverzi stránky do nového formátu
- Velikost potřebného místa závisi na relaci
- Součást preupgrade fáze a je nutné pustit před upgradem
- Neovlivní dostupnost staré databáze
- Kód bude backporotván, jakmile vyjde nová verze

Ostatní

Uložené procedury

- Změny v PL jazycích
  - Většinou jsou zpětně kompatibilní
  - Případně je možné doručit starou verzi PL jazyka a rozlišovat verze
  - Problém s procedurami psaných v C, nutné rekompilovat
    - Uživatelské datové typy

Tsearch2

- Změny ve FTS konfiguraci nebo ve slovnících vyžaduje regenerovat tsvectors attributy. Bohužel není k dispozici závislost mezi tsvector attributem a originálním zdrojem.

A co dál?

References

http://pgfoundry.org/projects/pg-migrator/

http://src.opensolaris.org/source/xref/sfw/usr/src/cmd/postgres/postgresql-upgrade/

http://wiki.postgresql.org/wiki/In-place_upgrade

PostgreSQL upgrade včera, dnes a zítra

Zdeněk Kotala

zdenek.kotala@sun.com