Présentation technique du projet PostgreSQL

v.1.0
01/12/2009

Sommaire

- Généralités sur PostgreSQL
- Instance
- Organisation des fichiers
- Processus
- Gestion de la mémoire
- Interne

PostgreSQL

- PostgreSQL a besoin du système d'exploitation
- Il utilise les ressources que le système d'exploitation met à sa disposition
  - Processeurs, Mémoire, Disques
- Il ne cherche pas à se substituer au système d'exploitation
  - Ce n'est pas un système d'exploitation (cache disque, prioritisation processus et I/O)
  - Ce n'est pas un système de fichiers (pas de raw devices)

Instance - Définition

- Regroupe
  - Répertoire des données
  - Processus
  - Mémoire
- Le terme cluster est aussi souvent utilisé

Instance – Création

- Outil de la communauté
  - initdb
- Outils spécifique Debian
  - pg_createcluster, pg_dropcluster
  - pg_ctlcluster
  - pg_lsclusters
  - Wrapper Perl pour initdb

Instance – Un répertoire des données

- Une instance == un répertoire des données
- Initialisé par la commande initdb
- Contient
  - les fichiers de configurations
  - les fichiers de données
  - les journaux de transactions
  - les fichiers de contrôle / statut

Instance – Un numéro de port

- Une instance == un numéro de port
- Connexions TCP/IP
  - port de connexion
- Connexions par la socket Unix
  - nom du socket (.s.PGSQL.<numero port>)

Instance – plusieurs bases

- Une instance peut contenir plusieurs bases de données
- Partagent
  - Les mêmes processus d'administration
  - La même mémoire cache
- Plus simple à configurer que plusieurs instances avec chacune une base
- Mais peut poser des soucis pour les sauvegardes de type fichiers

Instance – plusieurs bases - 2

- Pas de « Use » à la MySQL
  - Une base MySQL correspond à un schéma dans PostgreSQL
- Une connexion à une base
  - on ne peut pas voir/requêter les objets d'une autre base
  - Forte isolation des objets des différentes bases
- Pour requêter sur une autre base
  - Module contrib dblink
  - Langage PL/proxy

Organisation des fichiers

- Répertoire principal des données
  - Généralement appelé PGDATA
  - Option -D pour la plupart des outils

Organisation des fichiers - 2

- Fichiers de configuration
  - postgresql.conf
  - pg_hba.conf
  - pg_ident.conf
- Attention, sous Debian, ils sont déplacés
  - /etc/postgresql/<version majeure>/<nom instance>

Organisation des fichiers - 4

- PG_VERSION
  - Numéro de version majeure
- postmaster.opts
  - Options de lancement du moteur
- postmaster.pid
  - PID, répertoire des données, infos sémaphore

Organisation des fichiers - 5

- Répertoire base
- Contient chaque base de données
- Une base de données == un répertoire
- Le nom du répertoire est un identifiant système
- Trois moyens pour retrouver le nom de la base
  - SELECT datname FROM pg_database
    WHERE oid=<identifiant>
  - oid2name | grep <identifiant>
  - grep <identifiant> $PGDATA/global/pg_database

Organisation des fichiers - 6

- Dans chaque base
  - donc $PGDATA/base/<identifiant base>
- Un fichier par table et index
  - Augmente par bloc de 8 Ko
  - Taille maximum 1 Go (extension .<numero>)
- Deux moyens pour retrouver le nom de la table
  - SELECT relname FROM pg_class

    WHERE relfilenode=<identifiant table>
  - oid2name -d <nom base> | grep <identifiant table>

Organisation des fichiers - 7

- Fichiers associés à chaque table/index
- Un fichier TOAST associé
  - Si le contenu de certaines cellules (varchar, text, bytea) dépasse 2 Ko
  - Identifiant système spécifique
- Un fichier d'extension _fsm
  - Contient tous les espaces libres trouvés par le VACUUM
- Un fichier d'extension _vm
  - Contient la liste des lignes visibles par les processus

Organisation des fichiers - 8

- pg_xlog, liste des journaux de transactions
- Contient une liste de fichiers de 16 Mo
- Nombre maximum de fichiers
  - Environ 3*checkpoint_segments + 1
- Sous-répertoire archive_status
  - Extension .ready : liste des journaux prêt à être archivés
  - Extension .done : liste des journaux archivés

Organisation des fichiers - 9

- global
  - Tables globales à l'instance (utilisateurs et tablespaces)
- pg_clog
  - Informations de statut des transactions (en cours, COMMITée, ROLLBACKée)
- pg_log
  - Journaux applicatifs de PostgreSQL si logging_collector activé

Organisation des fichiers - 10

- pg_tblspc
  - Liens symboliques pointant vers l'emplacement des tablespaces
- pg_stat_tmp
  - Fichier pgstat.stat (contenant les statistiques de l'activité)
- pg_multixact, pg_subtrans, pg_twophase

Processus

- PostgreSQL n'est pas multi-thread
- Il est fortement multi-processus
- Deux types de processus
  - Processus d'administration
  - Processus de communication client/serveur

Processus - postmaster

- Processus père de tous les autres
- Commence par réclamer la mémoire partagée
- Exécute les processus d'administration au démarrage
- Écoute les connections entrantes, via le socket et/ou le port TCP/IP

Processus - bgwriter

- Processus d'écriture en tâche de fond des fichiers de données
- S'occupe de l'écriture des blocs modifiés en mémoire cache (shared_buffers) dans les fichiers de données
- Deux paramètres indiquent la fréquence des écritures sur disque
  - checkpoint_segments, checkpoint_timeout
  - checkpoint_completion_target

Processus – wal writer

- Processus d'écriture en tâche de fond des journaux de transactions
- S'occupe de l'écriture des blocs modifiés en mémoire cache (wal_buffers) dans les journaux de transactions

Processus – stats collector

- Récupère les statistiques sur le nombre de lignes et le nombre de blocs affectés par les opérations style INSERT/UPDATE/DELETE
- Récupère les dates et heures des VACUUM/ANALYZE manuels et automatiques
- Activable via le paramètre track_activities
- Discute avec les autres processus via un port UDP
- Données enregistrées dans le fichier pgstat.stat

Processus – log collector

- Réalise la rotation des journaux applicatifs
- (Dés)Activable avec le paramètre logging_collector
- Réagit à deux signaux
  - Signal SIGHUP pour lui demander de relire la configuration
  - Signal SIGUSR1 pour demander une rotation du journal applicatif

Processus – autovacuum launcher

- Exécuté en permanence
- Lance à intervalle régulier un processus autovacuum worker
- Désactivable via le paramètre autovacuum

Processus – autovacuum worker

- Se connecte à une base
- Lance des VACUUM et des ANALYZE
- Leur nombre dépend de autovacuum_max_workers
- Chacun peut utiliser maintenance_work_mem en cas d'exécution d'un VACUUM
- Ils peuvent être exécutés sur la même base en même temps
  - Utilisation de mémoire partagée pour communiquer

Processus - pgarch

- Archive les journaux de transactions
- Activable avec le paramètre archive_mode

Processus - postgres

- Chargé de la communication entre un client et le serveur
- Nombre maximum de processus dépendant du paramètre max_connections
- Peut allouer work_mem par tri et hachage pour l'exécution d'une requête
- Peut allouer maintenance_work_mem pour certaines opérations
  - VACUUM, CREATE INDEX, etc.

Gestion de la mémoire

- Deux types d'utilisation de la mémoire
  - Mémoire partagée par tous les processus
  - Mémoire spécifique à chaque processus

Mémoire - Partagée

- Cache disque
  - fichiers de données : shared_buffers
  - Journaux de transactions : wal_buffers
- Mais aussi
  - Verrous
  - Tables de correspondance (buffers, verrous)
  - Bgwriter
  - Autovacuum
  - Syncscan
  - etc.

Mémoire spécifique

- Tri, hachage
  - work_mem
- VACUUM, création d'index, clé étrangère
  - maintenance_work_mem
- Objets temporaires
  - temp_buffers
- Pile d'exécution
  - max_stack_depth

Interne

- Quelques concepts importants
- ACID
- MVCC
- Journaux de transactions
- Sauvegarde/Restauration

ACID

- Principe théorique des transactions
  - Atomic, Consistent, Isolation, Duration
- Implémentation complète dans PostgreSQL
  - Valable pour les modifications de données
  - Comme les modifications de la structure de la base
  - Seul bémol, les séquences
  - Méthode MVCC

MVCC

- Acronyme
  - Multi
  - Version
  - Concurrency
  - Control

- Autrement dit ?
  - Contrôle des accès simultanés par gestion de différentes versions d'une même ligne

MVCC - Principe

- Une lecture ne doit pas bloquer une autre lecture
- Une écriture ne doit pas bloquer une lecture
- Une lecture ne doit pas bloquer une écriture
- Une écriture ne doit pas bloquer une écriture
  - Sauf si l'écriture concerne la même ligne

MVCC - But

- Fluidifier l'accès aux données grâce à des verrous légers
- Faciliter la vie des développeurs
  - pas de gestion manuelle des verrous

MVCC - Contenu réel d'une table

- En plus des colonnes utilisateurs, il existe aussi des colonnes systèmes
  - xmin, identifiant de transaction de création de la ligne
  - xmax, identifiant de transaction de suppression de la ligne
  - ctid, paire indiquant l'emplacement dans le fichier
  - et quelques autres...

MVCC – Exécution réelle

- Pour un DELETE, PostgreSQL n'enregistre que le moment de la suppression des lignes
- Pour un UPDATE, PostgreSQL fait comme si la ligne avait été supprimée et enregistre la nouvelle version sur une autre ligne
- Il en résulte une fragmentation de la table

MVCC – VACUUM

- Pas de récupération automatique de l'espace mort
- Seul moyen (non pénible) d'éviter la fragmentation
  - VACUUM
- Lit séquentiellement la table
- Vérifie pour chaque ligne morte si la ligne est toujours visible par certaines transactions
- Les espaces libres sont stockés dans les fichiers d'extention _fsm

MVCC – Défragmentation

- Le VACUUM ne permet pas de gagner de l'espace disque
- Moyens de défragmentation réelle
  - VACUUM FULL, CLUSTER
    - Verrou exclusif
  - SELECT * INTO nouvelle_table FROM ancienne_table_fragmentée
  - dump/restore
    - L'activité pour ces deux derniers ne sera pas pris en compte

MVCC – la fragmentation n'est pas un mal

- Ou plutôt « pas forcément »
- Un peu de fragmentation peut être utile
- Suffisamment pour qu'il soit possible de la réclamer
  - Clause FILL FACTOR
  - À partir de la 8.2
- Par défaut:
  - 100% pour les tables
  - 90% pour les index

MVCC – améliorations techniques

- Autovacuum
- Refonte de la FSM
- VACUUM partiel

Journaux de transactions

- Contient toute l'activité de l'instance
  - Donc l'activité des différentes bases de cette instance s'y trouve
- Seul bémol : les index hachés
- Un journal de transaction == un fichier de 16 Mo
- Au maximum, 3*checkpoint_segments + 1 fichiers

Journaux de transactions – contenu d'un fichier

- Liste d'enregistrements
  - Création d'une base
  - Suppression d'une base
  - Modification de tel fichier, à tel offset, pour y ajouter la liste d'octets qui suit
  - Etc.
- L'enregistrement ne contient pas la requête SQL
  - Et il est impossible de la déduire

Journaux de transactions – gestion d'un fichier

- PostgreSQL remplit un journal de transaction
- Puis passe au suivant
- L'ancien est archivé si nécessaire
- Puis renommé

Sauvegarde / Restauration

- Trois moyens de sauvegarder
  - Sauvegarde des fichiers
  - pg_dump/pg_dumpall
  - Sauvegarde au fil de l'eau
- Toutes permettent des sauvegardes cohérentes
- Seule la sauvegarde des fichiers implique un arrêt de production
  - Sauf cas particulier : snapshot (géré par le FS, le SAN, le NetApp)

Sauvegarde - pg_dump

- Sauvegarde des objets d'une seule base
- Différents formats : SQL, tar, tar compressé
- Options pour spécifier les objets à sauvegarder
  - -t / -T : inclure / exclure une table
  - -n / -N : inclure / exclure un schéma
- Est un client PostgreSQL « normal »
  - Se connecte à la base, initie une transaction
- Ne bloque pas l'activité normale de la base
  - … mais ralentit les autres utilisateurs

Sauvegarde - pg_dumpall

- Sauvegarde toutes les bases et les objets globaux (utilisateurs et tablespaces)
- Un seul format : SQL
- Options pour ne sauvegarder que les objets globaux (-g / -r / -t)
- Là-aussi, client PostgreSQL « normal »
- Ne bloque pas plus l'activité normale de la base

Restauration – psql / pg_restore

- Format SQL : psql
- Format tar et tar compressé : pg_restore
- Pour pg_restore
  - Utilisation d'un sommaire permettant de restaurer uniquement ce que l'on souhaite
  - La restauration peut être parallélisée (threads sur Windows, processus sur Unix)

Sauvegarde - fichiers

- Moteur arrêté
  - Sauf cas spécifique des snapshots
- Sauvegarde de $PGDATA
  - Attention à bien inclure aussi les tablespaces
  - Et les journaux de transactions
- Impossible de sauvegarder une base ou une table spécifique
  - Tout devra être sauvegardé, puis restauré

Sauvegarde – au fil de l'eau

- Mise en place
  - Archivage des journaux de transactions
    - Modification du postgresql.conf
    - Rechargement de la config pour PostgreSQL
  - Sauvegarde des fichiers de l'instance
    - pg_start_backup('label')
    - Tar / cpio / rsync / etc.
    - pg_stop_backup()

Restauration – au fil de l'eau

- Restauration des fichiers de l'instance
- Création d'un fichier recovery.conf
  - Inverse de l'archivage
- Démarrage de PostgreSQL
  - Qui sera indisponible pendant toute la restauration

Pour terminer, de la lecture

- Manuel officiel
  - http://docs.postgresqlfr.org/current/
  - http://www.postgresql.org/docs/books/
- Articles dans GNU/Linux Magazine France
  - Numéro 107 : Gestion mémoire avec PostgreSQL
  - Numéro 112 : Les processus de PostgreSQL
  - Numéro 109 : Opérations de maintenance sous PostgreSQL
  - Disponible sur le site http://dalibo.org/articles

Conclusion

Merci...

- d'être venu
- et de m'avoir écouté :)

- Vos questions sont les bienvenues !