

Performances PostgreSQL au niveau du système

Introduction

- L'optimisation n'est pas un processus unique
 - Processus itératif
- Va de pair avec la supervision

Niveaux d'optimisation

- Système hôte, niveau matériel
- Configuration de PostgreSQL
- Répartition des données
- Application, modélisation et requêtes

Matériel

- Processeur
- Mémoire
- Disques
- Système disque

CPU

- Principaux critères
 - Nombre de cœurs
 - Fréquence
 - Taille du cache
- 32/64 bits

Choisir le CPU

- PostgreSQL n'est pas multithreadé
 - Connaître le type d'application
 - Privilégier
 - le nombre de cœurs (type appli web)
 - La fréquence (type DW)

RAM

- Essentiel pour les performances
- De moins en moins cher
- « Plus il y en a, mieux c'est »

Disques

- 3 technologies : SATA, SAS et SSD
- Temps d'accès / débit
- Tester les disques

RAID

- Tolérance à la panne, performances
- Éviter le RAID 5
- Privilégier RAID 1 / RAID 10
- Attention à la qualité de la carte

RAID - cache

- Toujours activer le cache en lecture
- Cache en écriture
 - Protégé par une batterie
 - Vérifier l'état de la batterie

Virtualisation

- Non recommandé
 - Mutualisation des disques
 - Masque les ressources
 - Overcommit
 - Gestion des vCPU multiples

Systeme d'exploitation

- Quel systeme choisir ?
- Quelle configuration ?

Choix du système

- PostgreSQL fonctionne sur de nombreux OS
- Principalement développé et testé sous linux
- Windows intéressant pour les développeurs

Noyau

- Choisir le noyau le plus récent
 - Plus stable
 - Plus de compatibilité
 - Plus de fonctionnalité
 - Meilleures performances
- Ne pas compiler son noyau

Configuration du noyau

- Mémoire partagée
- Overcommit
- Swap
- Affinité cœurs / espaces mémoire
- Configuration dans `/etc/sysctl.conf`
- Activer avec `sysctl -p`

Mémoire partagée

- SHMMAX
- SHMALL
- Valeur basse par défaut pour Debian

Cache disque

- Gestion des « dirty pages »
 - `vm.dirty_ratio`
 - `vm.dirty_background_ratio`
 - `vm.dirty_bytes`
 - `vm.dirty_background_bytes`
- Configurer pour avoir qq centaines de Mo à synchroniser

Swap

- Contre performant pour un SGBD
 - Pas plus de 2 Go de swap
 - Paramètre swappiness
 - 60 par défaut
 - Descendre à 5 ou 10

Overcommit

- Pas d'overcommit sur PostgreSQL
 - `vm.overcommit_memory`
 - =2 pour le désactiver
 - `vm.overcommit_ratio`
 - Spécifie l'espace mémoire allouable:
 - «overcommit_ratio» pourcent de la RAM + SWAP
 - Pour pouvoir allouer uniquement la RAM disponible:
 - **$\text{overcommit_ratio} = (\text{RAM} - \text{SWAP}) * 100 / \text{RAM}$**

Afinité processeur / mémoire

- Architecture NUMA (multi-sockets)
- Contre productif avec PostgreSQL
- `vm.zone_reclaim_mode = 0` pour le désactiver

Système de fichier

- Windows
 - NTFS
- Linux
 - ext3, ext4, xfs ...
- Solaris
 - ZFS

Configuration FS

- noatime
- nodirctime
- dir_index

Configuration PostgreSQL

- Mémoire
- Planificateur
- WAL
- Statistiques
- Fichiers

Configuration - mémoire

- shared_buffers
- wal_buffers
- work_mem / maintenance_work_mem

Configuration - planificateur

- `random_page_cost`
 - 4 par défaut, bonne valeur pour disques lents
 - RAID 5
 - Disques SATA
 - 2,5 à 3 pour du RAID 10
 - 1 pour du SSD ou RAM \geq taille base
- `effective_cache_size`

Configuration - WAL

- fsync: assure la sécurité
- checkpoint_segments
- checkpoint_timeout
- checkpoint_completion_target

Configuration – statistiques

- track_activities
- track_count
- autovacuum

Configuration – fichiers WAL

- Placer les journaux sur un autre disque
- Option -X de l'outil initdb
- Lien symbolique

Configuration - fichiers

- Séparer les objets suivant leur utilisation
- Tablespaces
 - Séparer tables et index
 - Séparer archives et données vivantes
 - Paramétrage au tablespace
 - ALTER TABLESPACE nom SET (random_page_cost TO X);

Questions ?

- N'hésitez pas, c'est le moment