4

Architecture des processus

PostgreSQL est un serveur reposant sur le concept du multiprocessus. À son démarrage, il exécute différents processus, suivant la configuration. Certains sont activés par défaut, d'autres non. Certains ne sont pas désactivables. La plupart sont configurables via différents paramètres. Ils ont tous leur utilité et assurent une cohérence dans le fonctionnement d'une instance. De ce fait, si un processus s'arrête suite à un arrêt brutal, le serveur PostgreSQL peut se retrouver dans un état instable et forcera dans ce cas un redémarrage automatique ainsi qu'une récupération des données depuis le dernier checkpoint.

Voici ce que donne la commande `ps` tout de suite après avoir démarré un serveur PostgreSQL :

```
$ ps fxo pid,cmd | grep [p]ostgres
652662  \_ /opt/postgresql/14/bin/postgres
652663      \_ postgres: logger
652665      \_ postgres: checkpointer
652666      \_ postgres: background writer
652667      \_ postgres: walwriter
652668      \_ postgres: autovacuum launcher
652669      \_ postgres: archiver last was 000000010000000000000000D
652670      \_ postgres: stats collector
652671      \_ postgres: logical replication launcher
```

NOMS DES PROCESSUS

Sous certains systèmes d'exploitation, postgres est le nom de tous les processus du serveur PostgreSQL. Autrement dit, il est impossible de les différencier. Ceci est gênant pour l'administrateur, pas pour le serveur en lui-même.

Néanmoins, il peut être intéressant de vérifier la valeur du paramètre `update_process_title`. Ce dernier permet d'activer et de désactiver la mise à jour du nom du processus. Il est activé par défaut, sauf sous Windows à partir de la version 10 de PostgreSQL en raison de la surcharge généralement importante que cause l'activation de ce paramètre sous ce système d'exploitation.

D'autre part, il est possible de donner un nom à l'instance. Pour cela, il faut configurer le paramètre `cluster_name` en utilisant uniquement des caractères ASCII (tout autre

caractère est remplacé par un point d'interrogation). Ce nom est affiché après le texte `postgres:` , comme ici avec `prod` comme nom d'instance :

```
 652745  \_ /opt/postgresql/14/bin/postgres
652746      \_ postgres: prod: logger
652748      \_ postgres: prod: checkpointer
652749      \_ postgres: prod: background writer
652750      \_ postgres: prod: walwriter
652751      \_ postgres: prod: autovacuum launcher
652752      \_ postgres: prod: archiver
652753      \_ postgres: prod: stats collector
652754      \_ postgres: prod: logical replication launcher
```

Il existe deux types de processus :

- les processus de gestion interne (par exemple, ceux chargés des écritures dans les fichiers de données, celui chargé des statistiques d'activité, etc.) ;
- les processus de communication avec les clients (plus simplement, ceux chargés de l'exécution des requêtes SQL et des flux de réplication).

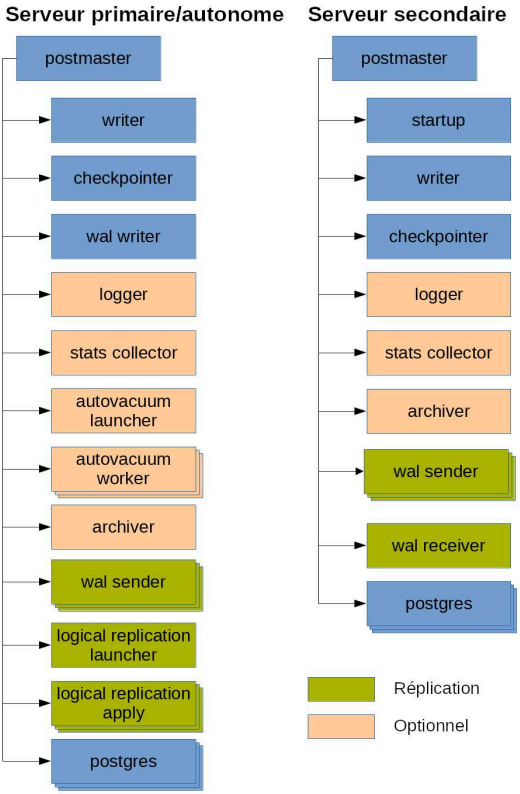
Les processus de gestion sont peu nombreux et disponibles en un seul exemplaire sur le système pour une même instance (à l'exception de l'`autovacuum worker`, du `wal sender` et du `logical replication worker`) :

- démarrage et gestion des sous-processus : `postmaster` et `startup` ;
- écriture dans les fichiers de données : `writer` et `checkpointer` ;
- écriture dans les journaux de transactions : `wal writer` ;
- gestion des statistiques d'activité : `stats collector` ;
- gestion des fichiers de traces : `logger` ;
- gestion automatique des `VACUUM` et `ANALYZE` : `autovacuum launcher` et `autovacuum worker` ;
- gestion de l'archivage des journaux de transactions : `archiver` ;
- gestion de la réplication par streaming : `wal receiver`, `logical replication launcher`, `logical replication worker`.

Les processus de communication client/serveur sont de deux types. Le premier gère les connexions standards (`postgres`). Le deuxième gère les connexions de réplication (`wal sender`).

Certains processus sont disponibles sur un serveur primaire ou sur un serveur autonome, alors que d'autres sont spécifiques aux serveurs secondaires. Certains sont optionnels, d'autres non. La Figure 4.1 en fournit une vue d'ensemble.

Figure 4.1 : *Processus serveurs*

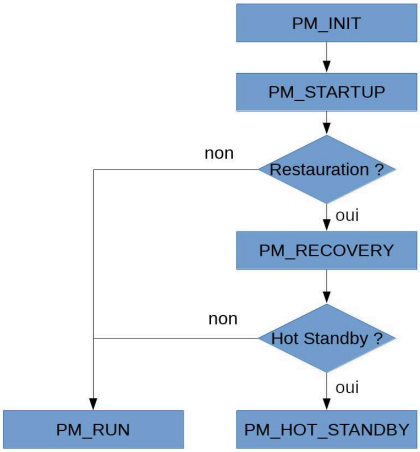


4.1. Démarrage et gestion des sous-processus

Toute la phase d'initialisation est réalisée par deux processus : `postmaster` lui-même et `startup`. Le premier commence en réalisant quelques initialisations, puis lance le processus `startup`. Ces deux processus vont s'exécuter en parallèle. Le processus `startup` s'occupe de rejouer les changements contenus dans les journaux de transactions qui n'auraient pas eu le temps d'atteindre les fichiers de données. Cela peut survenir en cas d'arrêt brutal du serveur. Il avertit le processus `postmaster` quand il entame le rejeu, il

l'avertit de nouveau quand il est arrivé au point de cohérence, ce qui permet au processus `postmaster` de changer l'état de l'instance à chaque avancée. La Figure 4.2 montre les différentes étapes de l'initialisation du point de vue de la machine à états.

Figure 4.2 : *États de la phase d'initialisation*



À la fin de l'initialisation, le processus `postmaster` est soit dans l'état `PM_RUN` (auquel cas le processus `startup` s'est arrêté), soit dans l'état `PM_HOT_STANDBY` (auquel cas le processus `startup` restera présent, et l'instance sera placée en lecture seule).

La phase d'exécution ne demande pas beaucoup de travail au processus `postmaster`. Il se contente de réagir aux signaux envoyés par les autres processus et aux demandes de connexions.

La phase d'arrêt est plus complexe, étant donné qu'il existe trois types d'arrêts :

- Le type *smart* oblige le processus `postmaster` à attendre l'arrêt volontaire des autres processus (et de ce fait, que les clients se déconnectent de leur propre volonté). Aucune connexion n'est possible en attendant la fin de l'arrêt. L'arrêt est propre, toutes les données modifiées en mémoire sont enregistrées sur disque.
- Le type *fast* demande au processus `postmaster` d'arrêter toutes les connexions de façon unilatérale (les clients ne sont pas informés de l'arrêt). L'arrêt est propre et rapide.
- Le type *immediate* correspond à un arrêt catastrophique. Tous les processus sont arrêtés rapidement, les données modifiées en mémoire sont perdues car le proces-