5

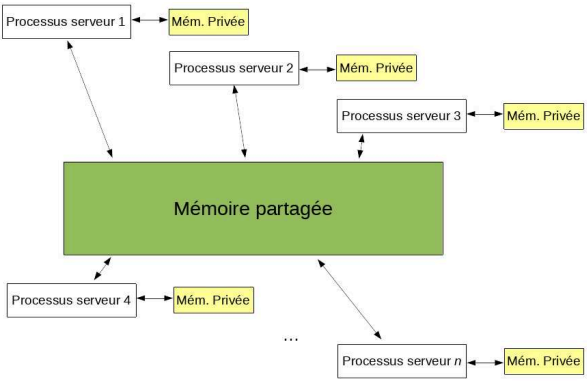
Architecture mémoire

Chaque processus a besoin d'allouer, d'utiliser, puis de désallouer de la mémoire pour réaliser les tâches qui sont à sa charge. Sur les systèmes d'exploitation modernes, les processus n'ont accès qu'aux portions de mémoire qu'ils ont allouées et ne peuvent pas accéder à des espaces mémoires appartenant à d'autres processus. Cela permet d'empêcher un processus de corrompre la mémoire d'autres processus suite à des erreurs de programmation. Or, pour fonctionner de façon efficace, ces différents processus doivent pouvoir partager certaines informations. Par exemple, lorsqu'un processus `postgres` lit une table particulière, il va la placer en mémoire. Pour que les autres processus qui doivent accéder à cette même table n'aient pas eux aussi à lire la table sur disque, il est intéressant qu'ils puissent accéder au bout de mémoire où le premier processus a placé la table. Il est aussi intéressant que les processus s'échangent des informations via un bout de mémoire, par exemple pour que le processus `autovacuum launcher` puisse indiquer à un processus `autovacuum worker` sur quelle base il doit travailler. La mémoire allouée de façon standard, avec `malloc`, ne peut pas être partagée entre différents processus. Les systèmes d'exploitation proposent depuis longtemps des appels systèmes permettant de partager de la mémoire entre différents processus, la seule contrainte étant que ces processus doivent avoir le même processus père. Le SGBD PostgreSQL utilise ce type de mémoire pour partager des informations entre ses différents processus.

Chaque processus utilise aussi une mémoire privée, nécessaire à l'exécution de certaines opérations, principalement pour exécuter des requêtes.

Ce chapitre va détailler les deux types de mémoire et présenter les paramètres de configuration permettant de régler les tailles mémoires.

Figure 5.1 : Différents types de mémoire



5.1. Mémoire partagée

PostgreSQL utilise de la mémoire partagée dans deux cas :

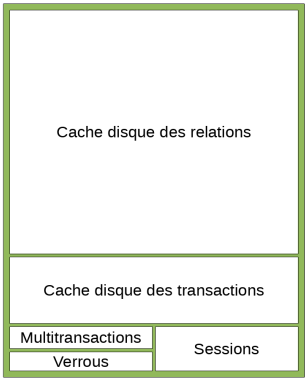
- diminuer les lectures et écritures disques via un système de mémoire cache (pour les relations et pour les fichiers de transactions) ;
- partager des informations entre processus.

Vue générale

La mémoire partagée est allouée dès le démarrage de PostgreSQL par le processus `postmaster`. Il est possible de catégoriser les différentes parties de la mémoire partagée ainsi :

- le cache disque des fichiers des relations (tables, index, vues matérialisées) ainsi que des fichiers de métadonnées (FSM et VM) de ces relations ;
- le cache disque des transactions ;
- le cache disque des multitransactions ;
- les verrous ;
- les données des sessions ;
- et des données diverses.

Figure 5.2 : Catégories de mémoire partagée



Note *› Le concept des multitransactions est expliqué dans la section Accès concurrents et verrous.*

La Figure 5.2 montre les différentes catégories au sein de la mémoire partagée. Le ratio de taille entre les différentes catégories n'est pas représenté exactement. Cependant, l'idée de base est là : le cache disque des relations est prédominant, ensuite vient le cache sur les transactions, puis les données des sessions.

À partir de la version 13, il est possible d'avoir une liste des différentes allocations de mémoire partagée avec la vue système `pg_shmem_allocations`. Voici par exemple les dix allocations les plus volumineuses :

```
SELECT * FROM pg_shmem_allocations
ORDER BY allocated_size DESC
LIMIT 10;

         name         |    off    |   size    | allocated_size
----------------------+-----------+-----------+----------------
 Buffer Blocks        |   6843392 | 134221824 |      134221824
 <anonymous>          |           |   4730240 |        4730240
 XLOG Ctl             |     54016 |   4208200 |        4208256
                      | 147614720 |   1922048 |        1922048
 Buffer Descriptors   |   5794816 |   1048576 |        1048576
 CommitTs             |   4792064 |    533920 |         534016
 Xact                 |   4262912 |    529152 |         529152
 Checkpointer Data    | 146858880 |    393280 |         393344
 Checkpoint BufferIds | 141327360 |    327680 |         327680
 Shared Memory Stats  | 147336832 |    277880 |         277888
(10 rows)
```

Depuis la version 15, il est aussi possible de connaître la quantité totale allouée pour la mémoire partagée avec le paramètre en lecture seule `shared_memory_size`.

Note › *Cette vue et ce paramètre ne concernent que les allocations pour le segment de mémoire partagé principal. Les allocations en mémoire partagée dynamique, notamment utilisée pour les requêtes parallèles, n'apparaissent pas dans cette vue.*

Cache disque des relations

PostgreSQL maintient son propre cache disque des relations (tables, index et vues matérialisées). Les métadonnées des tables (`Free Space Map` et `Visibility Map`) sont aussi présentes dans ce cache. Chaque lecture et chaque écriture sur ces objets passent par ce cache. Les processus `postgres` et `autovacuum worker` y placent des données suite aux lectures qu'ils font. Les processus `postgres`, `writer` et `checkpointer` écrivent les données modifiées en cache sur disque.

Comme PostgreSQL travaille sur les données par blocs de 8 ko, le cache est segmenté en un tableau de blocs de 8 ko. Chaque élément de ce tableau permet de monter en mémoire un bloc d'une relation ou d'un fichier de métadonnées afin de pouvoir le manipuler. Ce tableau est stocké dans un segment de mémoire partagée appelé `Buffer Blocks` dont la taille dépend du nombre d'éléments à conserver multiplié par la taille d'un élément. La taille d'un élément correspond à la taille d'un bloc (donc 8 ko par défaut). Le paramètre `shared_buffers` permet d'indiquer la taille du segment de mémoire partagée. Cette taille divisée par la taille d'un bloc permet de déterminer le nombre de blocs que l'on peut monter en cache.

Le fait de pouvoir accéder en mémoire à chacun des blocs des différentes relations à travers un tableau nécessite de pouvoir faire le lien entre le bloc stocké à tel emplacement dans le tableau et le bloc dans le fichier et les fichiers eux-mêmes. Un deuxième segment est donc créé pour contenir des enregistrements de description des différents blocs du cache. Ce segment est appelé `Buffer Descriptors`. Un descripteur contient plusieurs informations nécessaires au bon fonctionnement du cache. Notamment, il dispose du champ `usage_count` indiquant le nombre de fois où le bloc a été utilisé (un bloc peu utilisé sera rapidement remplacé par un autre bloc, à la différence d'un bloc très utilisé qui est plus intéressant à maintenir en cache). Il dispose aussi de plusieurs champs d'informations booléennes : bloc modifié (la documentation officielle parle de bloc *dirty*), bloc valide, lecture/écriture en cours, table journalisée, etc. La taille de ce segment de mémoire correspond à la multiplication du nombre de blocs conservés dans le cache disque (donc la valeur de `shared_buffers` divisée par la valeur de `block_size`) et de la taille d'un enregistrement de type descripteur de bloc.

Toujours dans l'idée de gérer efficacement le tableau de blocs, une table de hachage est créée dans un segment de mémoire partagée afin de permettre une recherche rapide. Ce segment s'appelle `Shared Buffer Lookup Table`. Afin d'éviter des conten-