Conceptos PosgreSQL

José David Mamani Vilca

I. *shared buffers*

Los *shared buffers* reprentan la proporción de memoria cedida a PostgreSQL para datos en caché. Por defecto tiene un valor bajo (de aproximadamente 32 mb) puesto que en algunos casos, con versiones antiguas de Solaris y SGI, se requieren de acciones invasivas como la recompilación de Kernel. El mínimo valor que se le puede colocar a este parámetro es de 128 mb pero se recomienda descartar esta opción puesto que valores más grandes revelan un mejor rendimiento. Si por ejemplo, se tiene un sistema con 1GB de RAM, un valor inicial razonable es un 25 % de dicha memoria. Si en caso, el sistema dispone de menor memoria se recomienda reducir dicho valor al 15 %.

Algunas observaciones extras recaen en versiones viejas de Windows y PostgreSQL (previas a la 8.1). En estos casos, valores altos del *shared buffers* no son efectivos, por lo que es aconsejable utilizar el caché del sistema operativo. Para versiones más modernas de Windows, el rango útil de memoria recae entre los 64mb a los 512mb. Finalmente, si el valor del *shared buffer* es muy grande, se recomienda incrementar el valor de *checkpoint segment* para que el proceso de escritura con bloques de datos my grandes se haga con mayor eficiencia.

II. *effective cache size*

Indica cuanta memoria tiene PosgreSQL para los datos de almacenamiento en caché y ayuda a determinar si es viable el uso de un índice (Por defecto tiene un valor de 4GB). Por lo general su asignación va en relación a la cantidad de memoria dispuesta en los *shared buffers* más la cantidad de memoria caché disponble para el sistem operativo. Debido a esto su asignación va por encima del 50 % de la cantidad total de memoria en el sistema.

Para la configuración de este parámetro se debe también considerar el número máximo de *queries simultáneas* en diferente tablas dado que estas tendrán que compartir dicho espacio en memoria (*effective cache size* $= 0{,}75*$ Memoria RAM disponible).

III. *Checkpoint Segment*

Es un mecanismo que actúa como seguro en caso la basee de datos déje de funcionar. Su uso evita que la base de datos tenga que enviar y validar sus datos constatemente. Básicamente, ante de realizar un cambio, este primero tiene que ser registrado en un archivo de manera tal que si un error sucede, se deshacen todos los cambios que no se han completado.

Actualmente se conocen como *WAL checkpoints* y están compuesto por una serie de parámetros que controlan cuando se debe crear un nuevo punto de verificación. Un *checkpoint* es por ende un punto en el tiempo en donde se garantiza que los cambios descritos han sido concretados, y todos los registros previos a este no deben ser tomados en cuenta en caso un error suceda.