max_connections

max_connections aplica la cantidad máxima de conexiones de clientes. esto es muy importante para algunos de los parámetros siguientes (particularmente work_mem) porque hay recursos de memoria que pueden ser ubicados por cliente, entonces el número máximo de clientes puede sugerir el máximo de memoria utilizada posible.

shared_buffers

La configuración del parámetro shared_buffers determina cuanta memoria está dedicada a PostgreSQL para datos en caché. Si tiene un sistema con 1GB o más de RAM, un valor inicial razonable es un 1/4 de dicha memoria. Si tiene menos deberá calcular cuidadosamente este valor de acuerdo al sistema operativo, cercano al 15% en los casos más comunes.

Note que en Windows y versiones viejas de Postgresql (anteriores a 8.1), altos valores de shared_buffers no son efectivos, teniendo buenos resultados manteniéndolo relativamente bajo (alrededor de 50.000, quizás menos) y utilizando mejor el caché del sistema operativo.

Es parecido a que elevando la cantidad de memoria de su sistema operativo le permitirá establecer el valor de shared_buffers alto. Si ud. lo establece más allá de lo soportado, obtendrá un mensaje parecido a este:

Cambiar este valor requerirá reiniciar la base de datos. Además, se trata de una difícil asignación de memoria; quedará todo fuera de la memoria virtual cuando se inicie la base de datos.

effective_cache_size

Este debe ser establecido en un monto estimado de cuanta memoria está disponible para memoria intermedia en el disco para el sistema operativo, luego de entrar a una cuenta usada por el Sistema operativo, dedicado a la memoria de postgresql y otras aplicaciones. Esta es una guía para cómo se espera que esté disponible la memoria en el búfer de caché de sistema operativo, no una asignación. Este valor es solo utilizado por el planeador de consultas para tener en cuenta los planes que puedan o no caber en memoria. Si es establecido demasiado bajo, los índices no se utilizarían del modo que ud. esperaría.

Estableciendo effective_cache_size a la mitad del total de la memoria, debería ser la opción más conservadora y 3/4 para una opción más agresiva pero que sigue siendo razonable. Sería conveniente que elija este valor de acuerdo a las estadísticas del sistema operativo. En sistemas Unix/linux, sume free + cached arrojados por los comandos free o top para tener una estimación. En Windows vea el tamaño del "System Cache" de la pestaña del Administrador de Tareas. Cambiar este valor no requiere reiniciar el servidor (un HUP sería suficiente o un RELOAD).

work_mem maintainance_work_mem

Si hace muchas ordenaciones complejas, y tiene bastante memoria, incrementando esta variable le permitirá a PostgreSQL a realizar ordenamientos más distendidos en memoria, obviamente incrementando la performance en comparación a las basadas en disco.

Este tamaño está aplicada a cada uno de los ordenamientos para cada usuario, y consultas complejas pueden utilizar múltiples buffers de memoria dedicados a estos. Establecerlo en 50MB y teniendo 30 usuarios ejecutando consultas y estaría utilizando 1.5GB de memoria real. Más allá, si una consulta implica hacer ordenamientos con juntas de 8 tablas, requeriría 8 veces work_mem. Debería considerar lo que tiene establecido en max_connections para establecer el work_mem apropiadamente. Este es un valor donde los almacenes de datos, donde los usuarios ejecutan consultas extensas, podrían llegar a utilizar gigas en memoria.

maintenance_work_mem es utilizada para operaciones de vacuum (limpieza). Usar valores muy altos no ayudaría mucho, y porque debería reservar esa memoria cuando vacuum entre en escenario, para cuando realmente estaría con mejores propósitos. En esos casos 256mb es anecdóticamente razonable para los valores altos.

checkpoint_segments checkpoint_completion_target

PostgreSQL escribe las nuevas transacciones a la Base de Datos en un archivo llamado segmentos del WAL que son de 16MB de tamaño. Todo el tiempo el valor de checkpoint_segments escrito, por defecto 3, ocurre un 'checkpoint' o punto de chequeo. Estos pueden ser un recurso intensivo, y en un sistema moderno hacer uno cada 48 MB puede ocasionar cuellos de botella. Estableciendo este valor un poco más alto mejoraría este inconveniente. Si esta corriendo en una configuración pequeña, debería establecer esta al menos en 10, permitiendo alcanzar los objetivos.

Para sistemas de escritura masiva, valores desde 32 (punto de chequeo cada 512MB) a 256 (cada 128GB) son vias populares. Sistemas muy grandes utilizan muchísimo más disco de lo que la recuperación llevaría más tiempo, por lo que debería elegir en qué rango se encuentra comfortable. Normalmente los valores altos (>64/1GB) son utilizadas para cargas de gran aumento de volumen. de cualquier manera que elija los segmentos, necesitará un punto de chequeo por lo menos cada 5 minutos a menos que aumente el checkpoint_timeout (lo que no es necesario en muchos sistemas).

Comenzando con PostgreSQL 8.3, las escrituras del punto de chequeo se extiende un poco mientras comienza a trabajar en el próximo punto. Puedes difundir las nuevas escrituras, la reducción de escribir encima de la media, incrementando el parámetro checkpoint_completion_target a su máximo de 0.9 (con el objetivo de terminar en el 90 % del tiempo antes del próximo checkpoint) en vez del valor por defecto de 0.5 (terminando cuando el próximo está en un 50%). Establecerlo a 0 daría algo similar a lo que era en las

versiones más tempranas. La razón principal de que 0.9 no es el valor por defecto es que se necesita un valor alto de checkpoint_segments para la difusión funcione bien.