



Configuración de la base de datos

Enterprise applications

NetApp
February 10, 2026

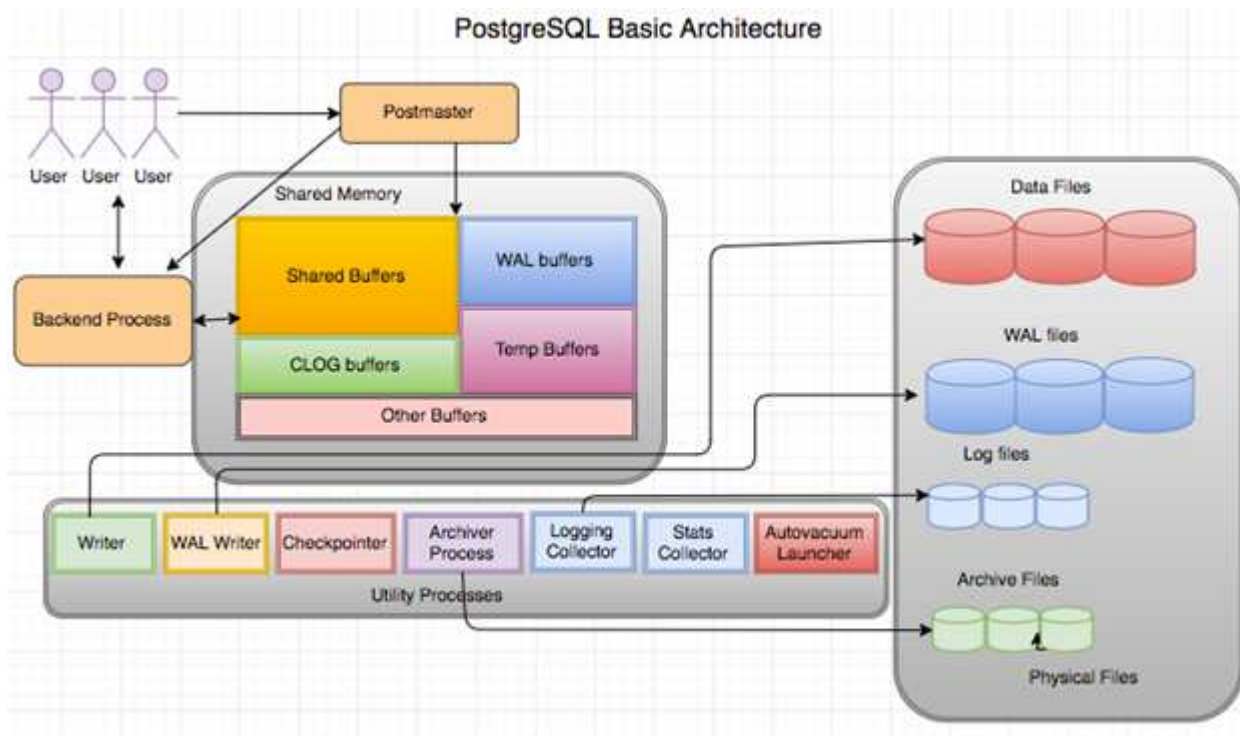
Tabla de contenidos

- Configuración de la base de datos 1
 - Arquitectura 1
 - Parámetros de inicialización 2
 - Configuración 2
 - BRINDIS 4
 - VACÍO 4
 - Espacios de tabla 4

Configuración de la base de datos

Arquitectura

PostgreSQL es un RDBMS basado en la arquitectura de cliente y servidor. Una instancia PostgreSQL se conoce como un cluster de base de datos, que es una colección de bases de datos en lugar de una colección de servidores.



Hay tres elementos principales en una base de datos PostgreSQL: El postmaster, el front-end (cliente) y el back-end. El cliente envía solicitudes al postmaster con información como el protocolo IP y a qué base de datos conectarse. El postmaster autentica la conexión y la pasa al proceso back-end para continuar la comunicación. El proceso back-end ejecuta la consulta y envía los resultados directamente al front-end (cliente).

Una instancia PostgreSQL se basa en un modelo multiproceso en lugar de un modelo multithread. Genera múltiples procesos para diferentes trabajos, y cada proceso tiene su propia funcionalidad. Los procesos principales incluyen el proceso del cliente, el proceso del escritor de WAL, el proceso del escritor en segundo plano y el proceso del puntero de control:

- Cuando un proceso de cliente (primer plano) envía solicitudes de lectura o escritura a la instancia de PostgreSQL, no lee ni escribe datos directamente en el disco. Primero almacena los datos en buffers compartidos y buffers de registro de escritura anticipada (WAL).
- Un proceso de escritor WAL manipula el contenido de los buffers compartidos y los buffers WAL para escribir en los logs WAL. Los registros WAL son normalmente registros de transacciones de PostgreSQL y se escriben secuencialmente. Por lo tanto, para mejorar el tiempo de respuesta de la base de datos, PostgreSQL primero escribe en los registros de transacciones y reconoce al cliente.
- Para poner la base de datos en un estado coherente, el proceso de escritor en segundo plano comprueba periódicamente el buffer compartido para ver si hay páginas sucias. A continuación, vacía los datos en los archivos de datos que se almacenan en volúmenes o LUN de NetApp.

- El proceso de puntero de control también se ejecuta periódicamente (con menos frecuencia que el proceso en segundo plano) e impide cualquier modificación en los buffers. Indica al proceso de escritor WAL que escriba y vacíe el registro de punto de control al final de los registros WAL que están almacenados en el disco NetApp. También indica al proceso de escritura en segundo plano que escriba y vacíe todas las páginas sucias en el disco.

Parámetros de inicialización

Cree un nuevo cluster de base de datos mediante `initdb` programa. An `initdb` script crea los archivos de datos, las tablas del sistema y las bases de datos de plantilla (`template0` y `template1`) que definen el cluster.

La base de datos de plantillas representa una base de datos de stock. Contiene definiciones para tablas del sistema, vistas estándar, funciones y tipos de dato. `pgdata` actúa como argumento para el `initdb` script que especifica la ubicación del cluster de base de datos.

Todos los objetos de base de datos en PostgreSQL son administrados internamente por los OIDs respectivos. Las tablas y los índices también se gestionan mediante OID individuales. Las relaciones entre los objetos de base de datos y sus respectivos OID se almacenan en las tablas de catálogo del sistema adecuadas, según el tipo de objeto. Por ejemplo, los OIDs de las bases de datos y las tablas de pila se almacenan en `pg_database` y `pg_class`, respectivamente. Puede determinar los OID emitiendo consultas en el cliente PostgreSQL.

Cada base de datos tiene sus propias tablas individuales y archivos de índice que están restringidos a 1GB. Cada tabla tiene dos archivos asociados, sufijos respectivamente con `_fsm` y `_vm`. Se les conoce como el mapa de espacio libre y el mapa de visibilidad. Estos archivos almacenan la información sobre la capacidad de espacio libre y tienen visibilidad en cada página del archivo de tabla. Los índices solo tienen mapas de espacio libre individuales y no tienen mapas de visibilidad.

La `pg_xlog/pg_wal` el directorio contiene los logs de escritura anticipada. Los registros de escritura anticipada se utilizan para mejorar el rendimiento y la fiabilidad de las bases de datos. Cada vez que actualiza una fila en una tabla, PostgreSQL escribe primero el cambio en el registro de escritura anticipada y, más tarde, escribe las modificaciones en las páginas de datos reales en un disco. La `pg_xlog` el directorio normalmente contiene varios archivos, pero `initdb` crea solo el primero. Se añaden archivos adicionales según sea necesario. Cada archivo xlog tiene 16MB cm de longitud.

Configuración

Existen varias configuraciones de ajuste PostgreSQL que pueden mejorar el rendimiento.

Los parámetros más utilizados son los siguientes:

- `max_connections` = <num>: El número máximo de conexiones de base de datos que se deben tener al mismo tiempo. Use este parámetro para restringir el intercambio en disco y eliminar el rendimiento. En función de los requisitos de la aplicación, también puede ajustar este parámetro para la configuración del pool de conexiones.
- `shared_buffers` = <num>: El método más simple para mejorar el rendimiento de su servidor de base de datos. El valor por defecto es bajo para la mayoría del hardware moderno. Se establece durante la implementación en aproximadamente el 25% de la RAM disponible en el sistema. Esta configuración de parámetro varía según cómo funciona con instancias de base de datos concretas; es posible que tenga que aumentar y disminuir los valores por prueba y error. Sin embargo, es probable que si lo establece alto, el rendimiento se vea afectado.

- `effective_cache_size = <num>`: Este valor indica al optimizador de PostgreSQL cuánta memoria PostgreSQL tiene disponible para almacenar datos en caché y ayuda a determinar si se debe usar un índice. Un valor mayor aumenta la probabilidad de usar un índice. Este parámetro se debe definir en la cantidad de memoria asignada a `shared_buffers` Más la cantidad de caché del sistema operativo disponible. A menudo, este valor supera el 50% de la memoria total del sistema.
- `work_mem = <num>`: Este parámetro controla la cantidad de memoria que se utilizará en las operaciones de ordenación y las tablas hash. Si realiza una clasificación intensiva en su aplicación, es posible que necesite aumentar la cantidad de memoria, pero tenga cuidado. No es un parámetro de todo el sistema, sino uno por operación. Si una consulta compleja tiene varias operaciones de ordenación en ella, utiliza varias unidades de memoria `work_mem`, y varios back-ends podrían estar haciendo esto simultáneamente. Esta consulta a menudo puede hacer que el servidor de base de datos cambie si el valor es demasiado grande. Esta opción se llamaba anteriormente `sort_mem` en versiones anteriores de PostgreSQL.
- `fsync = <boolean> (on or off)`: Este parámetro determina si todas sus páginas WAL deben sincronizarse con el disco mediante el uso de `fsync()` antes de que se confirme una transacción. Desactivarlo puede mejorar el rendimiento de escritura y activarlo aumenta la protección frente al riesgo de daño cuando el sistema se bloquea.
- `checkpoint_timeout`: El proceso de punto de control vacía los datos confirmados en el disco. Esto implica una gran cantidad de operaciones de lectura/escritura en disco. El valor se establece en segundos y los valores más bajos disminuyen el tiempo de recuperación de fallos y el aumento de los valores puede reducir la carga en los recursos del sistema reduciendo las llamadas de punto de control. En función de la criticidad de la aplicación, el uso y la disponibilidad de la base de datos, defina el valor de `checkpoint_timeout`.
- `commit_delay = <num>` y `commit_siblings = <num>`: Estas opciones se utilizan juntas para ayudar a mejorar el rendimiento mediante la escritura de múltiples transacciones que se comprometen a la vez. Si hay varios objetos COMMIT_SIBLINGS activos en el momento en que la transacción se está confirmando, el servidor espera a COMMIT_DELAY microsegundos para intentar confirmar varias transacciones a la vez.
- `max_worker_processes / max_parallel_workers`: Configure el número óptimo de trabajadores para los procesos. `Max_parallel_workers` corresponde al Núm. De CPU disponibles. Dependiendo del diseño de la aplicación, las consultas pueden requerir un número menor de trabajadores para las operaciones en paralelo. Es mejor mantener el valor de ambos parámetros igual, pero ajustar el valor después de la prueba.
- `random_page_cost = <num>`: Este valor controla la forma en que PostgreSQL visualiza las lecturas de disco no secuenciales. Un valor más alto significa que PostgreSQL es más probable que use una exploración secuencial en lugar de una exploración de índice, lo que indica que su servidor tiene discos rápidos Modificar esta configuración después de evaluar otras opciones como optimización basada en planes, aspirar, indexar para alterar consultas o esquemas.
- `effective_io_concurrency = <num>`: Este parámetro establece el número de operaciones de E/S de disco simultáneas que PostgreSQL intenta ejecutar simultáneamente. Al aumentar este valor, aumenta el número de operaciones de I/O que cualquier sesión de PostgreSQL individual intenta iniciar en paralelo. El rango permitido es de 1 a 1.000, o cero para deshabilitar la emisión de solicitudes de E/S asíncronas. Actualmente, esta configuración sólo afecta a las exploraciones de pila de bitmap. Las unidades de estado sólido (SSD) y otro almacenamiento basado en memoria (NVMe) pueden procesar muchas solicitudes concurrentes, con lo que el mejor valor puede ser entre cientos.

Consulte la documentación de PostgreSQL para obtener una lista completa de los parámetros de configuración de PostgreSQL.

BRINDIS

TOAST es la sigla en inglés de la Técnica de Almacenamiento de Atributos Sobredimensionados. PostgreSQL utiliza un tamaño de página fijo (comúnmente 8KB) y no permite que las tuplas se abarquen varias páginas. Por lo tanto, no es posible almacenar valores de campo grandes directamente. Cuando intenta almacenar una fila que excede este tamaño, TOAST divide los datos de las columnas grandes en “pedazos” más pequeños y los almacena en una tabla de TOSTADAS.

Los grandes valores de atributos tostados se extraen (si se selecciona) solo en el momento en que se envía el conjunto de resultados al cliente. La tabla en sí es mucho más pequeña y puede caber más filas en la caché de buffers compartida de lo que podría sin ningún almacenamiento fuera de línea (TOSTADO).

VACÍO

En el funcionamiento normal de PostgreSQL, las tuplas que se eliminan o quedan obsoletas por una actualización no se eliminan físicamente de su tabla; permanecen presentes hasta que se ejecuta EL VACÍO. Por lo tanto, debe ejecutar EL VACÍO periódicamente, especialmente en tablas actualizadas con frecuencia. A continuación, se debe reclamar el espacio que ocupa para que las nuevas filas lo reutilicen, a fin de evitar la interrupción del espacio en disco. Sin embargo, no devuelve el espacio al sistema operativo.

El espacio libre dentro de una página no está fragmentado. EL VACÍO reescribe todo el bloque, empaquetando eficientemente las filas restantes y dejando un único bloque contiguo de espacio libre en una página.

Por el contrario, EL VACÍO COMPLETO compacta activamente las tablas escribiendo una versión completamente nueva del archivo de tabla sin espacio muerto. Esta acción minimiza el tamaño de la tabla, pero puede tardar mucho tiempo. También requiere espacio adicional en disco para la nueva copia de la tabla hasta que finalice la operación. El objetivo del VACÍO DE rutina es evitar la actividad COMPLETA DEL VACÍO. Este proceso no solo mantiene las tablas en su tamaño mínimo, sino que también mantiene el uso constante del espacio en disco.

Espacios de tabla

Al inicializar el cluster de la base de datos, se crean automáticamente dos tablespaces.

La `pg_global` el tablespace se utiliza para catálogos de sistemas compartidos. La `pg_default` tablespace es el tablespace por defecto de las bases de datos `template1` y `template0`. Si la partición o el volumen en el que se inicializó el cluster se queda sin espacio y no se puede ampliar, se puede crear un tablespace en una partición diferente y utilizarlo hasta que se pueda volver a configurar el sistema.

Un índice muy utilizado se puede colocar en un disco rápido y de alta disponibilidad, como un dispositivo de estado sólido. Además, se puede almacenar una tabla que almacene datos archivados que no se utilizan con poca frecuencia o que no son críticos para el rendimiento en un sistema de disco menos costoso y más lento como las unidades SAS o SATA.

Los tablespaces forman parte del cluster de base de datos y no se pueden tratar como una recopilación autónoma de archivos de datos. Dependen de los metadatos contenidos en el directorio de datos principal y, por lo tanto, no se pueden asociar a otro clúster de base de datos ni realizar copias de seguridad individuales. Del mismo modo, si pierde un tablespace (mediante la supresión de archivos, fallos de disco, etc.), el cluster de base de datos puede volverse ilegible o no se puede iniciar. Colocar un tablespace en un sistema de archivos temporal como un disco RAM pone en riesgo la fiabilidad de todo el cluster.

Una vez creado, se puede utilizar un tablespace desde cualquier base de datos si el usuario solicitante tiene suficientes privilegios. PostgreSQL utiliza enlaces simbólicos para simplificar la implementación de

tablespaces. PostgreSQL añade una fila al `pg_tablespace` Tabla (una tabla en todo el clúster) y asigna un nuevo identificador de objeto (OID) a esa fila. Por último, el servidor utiliza el OID para crear un enlace simbólico entre el cluster y el directorio dado. El directorio `$PGDATA/pg_tblspc` contiene enlaces simbólicos que apuntan a cada uno de los tablespaces no incorporados definidos en el cluster.

Información de copyright

Copyright © 2026 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en <http://www.netapp.com/TM> son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.