PostgreSQL

Enterprise applications

NetApp
May 09, 2024

Tabla de contenidos

PostgreSQL

Bases de datos PostgreSQL en ONTAP

PostgreSQL viene con variantes que incluyen PostgreSQL, PostgreSQL Plus y EDB Postgres Advanced Server (EPAS). PostgreSQL suele ponerse en marcha como base de datos de back-end para aplicaciones de varios niveles. Es compatible con paquetes de middleware comunes (como PHP, Java, Python, Tcl/Tk, ODBC, etc.). JDBC) y, desde siempre, ha sido una opción popular para los sistemas de gestión de bases de datos de código abierto. ONTAP es una opción excelente para ejecutar bases de datos PostgreSQL en cuanto a su fiabilidad, alto rendimiento y eficacia.

Esta documentación sobre ONTAP y la base de datos PostgreSQL reemplaza a la base de datos *TR-4770: PostgreSQL sobre las mejores prácticas de ONTAP.*

A medida que los datos crecen exponencialmente, la gestión de datos se vuelve más compleja para las empresas. Esta complejidad aumenta los costes de licencias, operaciones, soporte y mantenimiento. Para reducir el coste total de propiedad, considere la posibilidad de cambiar de bases de datos comerciales a bases de datos de código abierto con un almacenamiento de back-end fiable y de alto rendimiento.

ONTAP es una plataforma ideal, ya que ONTAP está literalmente diseñada para bases de datos. Numerosas funciones, como las optimizaciones de latencia de I/O aleatorias, pasando por una calidad de servicio avanzada o una funcionalidad FlexClone básica, se crearon específicamente para cubrir las necesidades de cargas de trabajo de bases de datos.

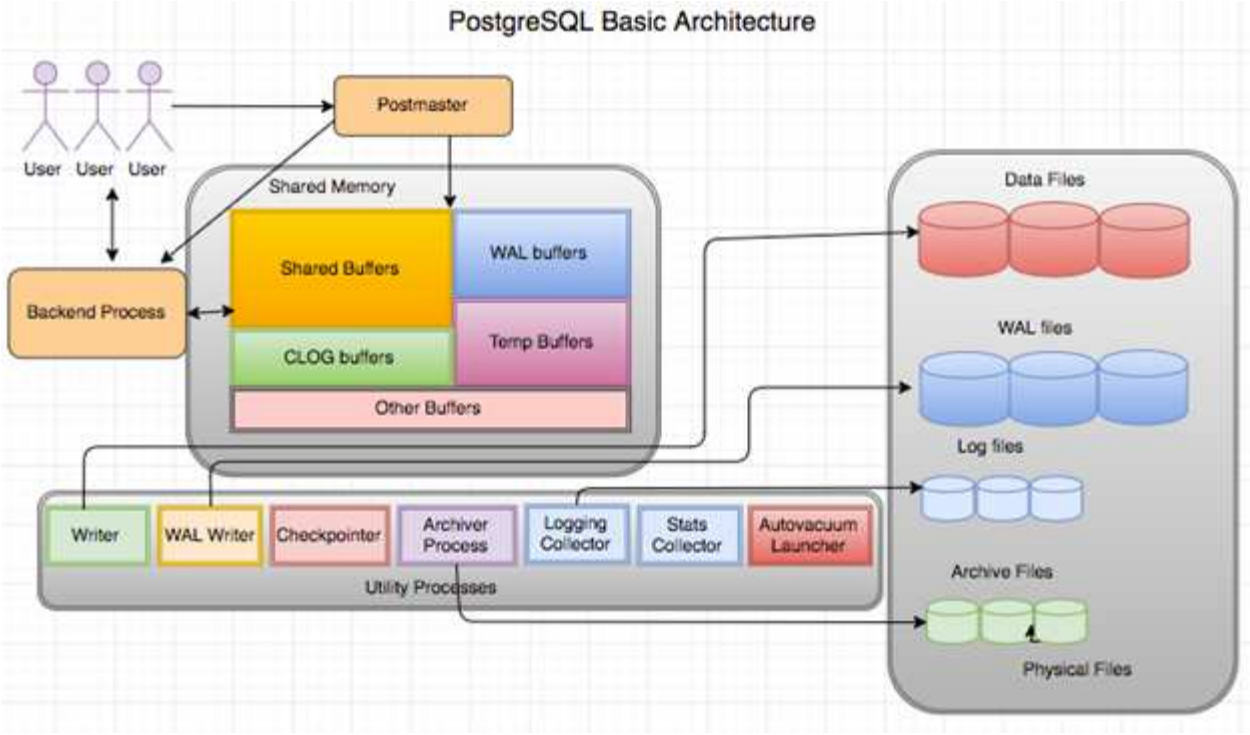
Otras funciones como las actualizaciones no disruptivas (entre ellas la sustitución de almacenamiento) garantizan que sus bases de datos cruciales seguirán estando disponibles. También se puede disponer de recuperación ante desastres instantánea para entornos grandes mediante MetroCluster o seleccionar bases de datos usando SnapMirror active sync.

Y lo que es más importante, ONTAP ofrece un rendimiento sin igual con la capacidad de dimensionar la solución en función de sus necesidades únicas. Nuestros sistemas de gama alta pueden ofrecer más de 1M 000 IOPS con latencias de microsegundos, pero si solo necesita 100K 000 IOPS, puede ajustar el tamaño de su solución de almacenamiento con una controladora más pequeña aún ejecutando exactamente el mismo sistema operativo de almacenamiento.

Configuración de la base de datos

Arquitectura PostgreSQL

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 posmaster 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 de PostgreSQL

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 de base de datos PostgreSQL con ONTAP

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_SIDBINGS 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.

Tablespaces PostgreSQL

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.

Configuración del almacenamiento

Bases de datos PostgreSQL con sistemas de archivos NFS

Las bases de datos PostgreSQL se pueden alojar en sistemas de archivos NFSv3 o NFSv4. La mejor opción depende de factores fuera de la base de datos.

Por ejemplo, el comportamiento de bloqueo NFSv4 puede ser preferible en ciertos entornos agrupados en clúster. (Consulte "aquí" para obtener más información).

De lo contrario, la funcionalidad de la base de datos debería ser casi idéntica, incluido el rendimiento. El único requisito es el uso del `hard` opción de montaje. Esto es necesario para garantizar que los tiempos de espera de software no produzcan errores de E/S irrecuperables.

Si se elige NFSv4 como protocolo, NetApp recomienda usar NFSv4,1. Existen algunas mejoras funcionales en el protocolo NFSv4 en NFSv4,1 que mejoran la resiliencia con respecto a la versión NFSv4,0.

Utilice las siguientes opciones de montaje para cargas de trabajo generales de bases de datos:

```
rw,hard,nointr,bg,vers=[3|4],proto=tcp,rsize=65536,wsize=65536
```

Si se esperan operaciones de I/O secuenciales pesadas, los tamaños de transferencia NFS pueden aumentar, tal como se describe en la siguiente sección.

Tamaños de transferencia de NFS

De forma predeterminada, ONTAP limita el tamaño de I/O de NFS a 64K.

La I/O aleatoria con la mayoría de aplicaciones y bases de datos utiliza un tamaño de bloque mucho más pequeño, que es muy inferior al máximo de 64K KB. Las operaciones de I/O de grandes bloques suelen estar en paralelo, por lo que el máximo de 64K KB tampoco se limita a obtener el ancho de banda máximo.

Hay algunas cargas de trabajo en las que el máximo de 64K crea una limitación. En particular, las operaciones de subproceso único como la operación de copia de seguridad o recuperación o una exploración de tabla completa de la base de datos se ejecutan de forma más rápida y eficiente si la base de datos puede realizar menos E/S pero más grandes. El tamaño óptimo de gestión de I/O para ONTAP es de 256K KB.

El tamaño de transferencia máximo para una SVM de ONTAP determinada se puede cambiar de la siguiente manera:

```
Cluster01::> set advanced
Warning: These advanced commands are potentially dangerous; use them only
when directed to do so by NetApp personnel.
Do you want to continue? {y|n}: y
Cluster01::*> nfs server modify -vserver vserver1 -tcp-max-xfer-size
262144
Cluster01::*>
```

Precaución
No reduzca nunca el tamaño máximo permitido de transferencia en ONTAP por debajo del valor de rsize/wsize de los sistemas de archivos NFS montados actualmente. Esto puede crear bloqueos o incluso corrupción de datos con algunos sistemas operativos. Por ejemplo, si los clientes NFS se establecen actualmente con un valor de rsize/wsize de 65536 000, el tamaño de transferencia máximo de ONTAP se podría ajustar entre 65536 000 y 1048576 000 sin que ello afecte a porque los propios clientes están limitados. Reducir el tamaño máximo de transferencia por debajo de 65536 puede dañar la disponibilidad o los datos.

Una vez que se aumenta el tamaño de transferencia en el nivel de ONTAP, se utilizarán las siguientes opciones de montaje:

```
rw,hard,nointr,bg,vers=[3|4],proto=tcp,rsize=262144,wsize=262144
```

NFSv3 Tablas de ranuras TCP

Si NFSv3 se utiliza con Linux, es fundamental configurar correctamente las tablas de ranuras TCP.

Las tablas de ranuras TCP son equivalentes a NFSv3 a la profundidad de la cola del adaptador de bus de host (HBA). En estas tablas se controla el número de operaciones de NFS que pueden extraordinarias a la vez. El valor predeterminado suele ser 16, que es demasiado bajo para un rendimiento óptimo. El problema opuesto ocurre en los kernels más nuevos de Linux, que pueden aumentar automáticamente el límite de la tabla de ranuras TCP a un nivel que sature el servidor NFS con solicitudes.

Para obtener un rendimiento óptimo y evitar problemas de rendimiento, ajuste los parámetros del núcleo que controlan las tablas de ranuras TCP.

Ejecute el `sysctl -a | grep tcp.*.slot_table` command, y observe los siguientes parámetros:

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

Todos los sistemas Linux deben incluir `sunrpc.tcp_slot_table_entries`, pero solo algunos incluyen `sunrpc.tcp_max_slot_table_entries`. Ambos deben establecerse en 128.

Precaución
Si no se establecen estos parámetros, puede tener efectos significativos en el rendimiento. En algunos casos, el rendimiento es limitado porque el sistema operativo linux no está emitiendo suficiente I/O. En otros casos, las latencias de I/O aumentan cuando el sistema operativo linux intenta emitir más operaciones de I/O de las que se pueden mantener.

PostgreSQL con sistemas de archivos SAN

Las bases de datos PostgreSQL con SAN generalmente se alojan en sistemas de archivos xfs, pero otras se pueden usar si es compatible con el proveedor del sistema operativo

Mientras que una única LUN puede admitir por lo general hasta 100K 000 IOPS, las bases de datos con un gran volumen de I/O normalmente requieren el uso de LVM con segmentación.

Segmentación de LVM

Antes de la era de las unidades flash, se utilizaba la segmentación para ayudar a superar las limitaciones de rendimiento de las unidades giratorias. Por ejemplo, si un sistema operativo necesita realizar una operación de lectura de 1MB KB, para leer que 1MB TB de datos de una sola unidad se requeriría buscar y leer muchos cabezales de unidad ya que 1MB se transfiere lentamente. Si esos 1MB TB de datos se segmentaron en 8

LUN, el sistema operativo podría emitir ocho operaciones de lectura de 128K KB en paralelo y reducir el tiempo necesario para realizar la transferencia de 1MB GB.

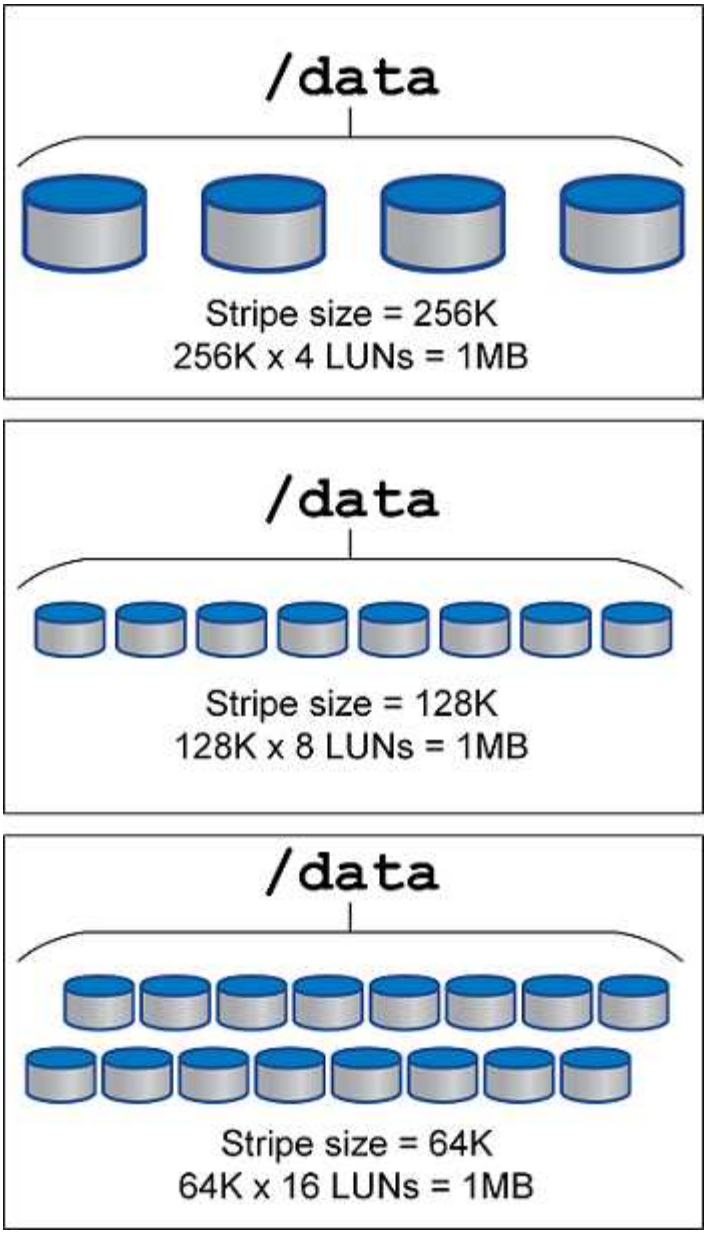
La segmentación con unidades giratorias era más difícil porque se tenía que conocer el patrón de I/O con anterioridad. Si la segmentación no se ajustó correctamente para los patrones de I/O reales, las configuraciones seccionadas podrían dañar el rendimiento. Con las bases de datos de Oracle y, especialmente con las configuraciones all-flash, la segmentación es mucho más fácil de configurar y se ha demostrado que mejora drásticamente el rendimiento.

Los gestores de volúmenes lógicos como Oracle ASM segmentan por defecto, pero el LVM del sistema operativo nativo no lo hacen. Algunos de ellos unen varias LUN como un dispositivo concatenado, lo que da como resultado archivos de datos que existen en un único dispositivo LUN. Esto provoca puntos calientes. Otras implementaciones de LVM toman por defecto extensiones distribuidas. Esto es similar a la segmentación, pero es más grueso. Las LUN del grupo de volúmenes se dividen en partes grandes, denominadas extensiones y normalmente se miden en muchos megabytes, y los volúmenes lógicos se distribuyen por esas extensiones. El resultado es que las operaciones de I/O aleatorias en un archivo se deben distribuir bien entre las LUN, pero las operaciones de I/O secuenciales no son tan eficientes como podrían.

La I/O de aplicaciones con rendimiento intensivo casi siempre es una (a) en unidades del tamaño de bloque básico o (b) un megabyte.

El principal objetivo de una configuración seccionada es garantizar que la I/O de archivo único se pueda realizar como una unidad única y que las I/O de varios bloques, que deben tener un tamaño de 1MB TB, se puedan paralelizar de manera uniforme entre todas las LUN del volumen seccionado. Esto significa que el tamaño de franja no debe ser menor que el tamaño del bloque de la base de datos y el tamaño de franja multiplicado por el número de LUN debe ser 1MB.

En la siguiente figura, se muestran tres opciones posibles para el ajuste del tamaño de la franja y el ancho. Se selecciona el número de LUN para satisfacer los requisitos de rendimiento tal como se han descrito anteriormente, pero en todos los casos los datos totales de una sola franja es 1MB.



Protección de datos

Protección de datos PostgreSQL

Uno de los principales aspectos del diseño del almacenamiento es permitir la protección para volúmenes PostgreSQL. Los clientes pueden proteger sus bases de datos PostgreSQL mediante el método de volcado o mediante copias de seguridad del sistema de archivos. Esta sección explica los diferentes enfoques para realizar una copia de seguridad de bases de datos individuales o de todo el cluster.

Existen tres enfoques para respaldar los datos de PostgreSQL:

- Volcado de SQL Server
- Backup de nivel de sistema de archivos

- Archivado continuo

La idea detrás del método de volcado de SQL Server es generar un archivo con comandos de SQL Server que, cuando se devuelve al servidor, pueda volver a crear la base de datos como estaba en el momento del volcado. PostgreSQL proporciona los programas de utilidad `pg_dump` y.. `pg_dump_all` para crear backup individual y a nivel de clúster. Estos volcados son lógicos y no contienen suficiente información para ser utilizada por WAL Replay.

Una estrategia de backup alternativa consiste en utilizar copias de seguridad a nivel de sistema de archivos, en las que los administradores copian directamente los archivos que PostgreSQL utiliza para almacenar los datos en la base de datos. Este método se realiza en modo offline: La base de datos o el cluster deben cerrarse. Otra alternativa es usar `pg_basebackup` Para ejecutar la copia de seguridad de transmisión en caliente de la base de datos PostgreSQL.

Bases de datos PostgreSQL e instantáneas de almacenamiento

Las copias de seguridad basadas en instantáneas con PostgreSQL requieren la configuración de instantáneas para archivos de datos, archivos WAL y archivos WAL archivados para proporcionar una recuperación completa o puntual.

Para las bases de datos PostgreSQL, el tiempo promedio de backup con snapshots es de unos pocos segundos a unos pocos minutos. Esta velocidad de backup es entre 60 y 100 veces más rápida que `pg_basebackup` y otros enfoques de backup basados en sistemas de archivos.

Las copias Snapshot en el almacenamiento de NetApp pueden ser coherentes con los fallos y con las aplicaciones. Se crea una copia Snapshot coherente con los fallos en el almacenamiento sin desactivar la base de datos, mientras que se crea una copia Snapshot coherente con la aplicación mientras la base de datos está en modo de backup. NetApp también garantiza que las copias Snapshot posteriores sean backups permanentes para ahorrar en almacenamiento y mejorar la eficiencia de la red.

Como las copias Snapshot son rápidas y no afectan al rendimiento del sistema, puede programar varias copias Snapshot diariamente en lugar de crear un único backup diario, como ocurre con otra tecnología de backup en streaming. Cuando es necesaria una operación de restauración y recuperación, el tiempo de inactividad del sistema se reduce gracias a dos funciones clave:

- La tecnología de recuperación de datos de NetApp SnapRestore significa que la operación de restauración se ejecuta en segundos.
- Los objetivos de punto de recuperación agresivos (RPO) significan que es necesario aplicar menos registros de base de datos y que también se acelera la nueva recuperación.

Para realizar el backup de PostgreSQL, debe asegurarse de que los volúmenes de datos estén protegidos simultáneamente con WAL (grupo de consistencia) y los registros archivados. Mientras utiliza la tecnología Snapshot para copiar archivos WAL, asegúrese de ejecutar `pg_stop` Para vaciar todas las entradas de WAL que se deben archivar. Si vacíe las entradas DE WAL durante la restauración, solo tendrá que detener la base de datos, desmontar o eliminar el directorio de datos existente, y realizar una operación de SnapRestore en el almacenamiento. Una vez finalizada la restauración, puede montar el sistema y devolverlo a su estado actual. Para la recuperación point-in-time, también puede restaurar WAL y archive logs; luego PostgreSQL decide el punto más consistente y lo recupera automáticamente.

Los grupos de coherencia son una función en ONTAP y se recomienda cuando hay varios volúmenes montados en una sola instancia o en una base de datos con varios espacios de tabla. Una snapshot de grupo de coherencia garantiza que todos los volúmenes estén agrupados y protegidos. Un grupo de consistencia puede gestionarse de manera eficiente desde ONTAP System Manager, e incluso puede clonarlo para crear

una copia de instancia de una base de datos con fines de prueba o desarrollo.

Para obtener más información sobre los grupos de consistencia, consulte "Información general de los grupos de consistencia NetApp".

Software de protección de datos PostgreSQL

El complemento de NetApp SnapCenter para base de datos de PostgreSQL, combinado con las tecnologías de Snapshot y FlexClone de NetApp, le ofrece ventajas como:

- Backup y restauración rápidos.
- Clones con gestión eficiente del espacio.
- La capacidad de crear un sistema de recuperación ante desastres rápido y eficaz.

> Puede que prefiera elegir los partners de backup premium de NetApp, como Veeam Software y Commvault, bajo las siguientes circunstancias:
>
> - Gestionar cargas de trabajo en un entorno heterogéneo
> - Almacenar backups en el cloud o en cinta para su retención a largo plazo
> - Soporte para una amplia gama de versiones y tipos de SO

El plugin de SnapCenter para PostgreSQL es un plugin de soporte comunitario y la configuración y documentación está disponible en la tienda de automatización de NetApp. Con SnapCenter, el usuario puede realizar backups de la base de datos, clonar y restaurar los datos remotamente.