PostgreSQL

Enterprise applications

NetApp
January 02, 2026

Índice

PostgreSQL

Visão geral

PostgreSQL vem com variantes que incluem PostgreSQL, PostgreSQL Plus e EDB Postgres Advanced Server (EPAS). O PostgreSQL é normalmente implantado como banco de dados back-end para aplicativos de várias camadas. Ele é suportado por pacotes de middleware comuns (como PHP, Java, Python, Tcl/Tk, ODBC e JDBC) e tem sido historicamente uma escolha popular para sistemas de gerenciamento de banco de dados de código aberto. O ONTAP é uma excelente escolha para executar bancos de dados PostgreSQL de acordo com sua confiabilidade, alto desempenho e recursos eficientes de gerenciamento de dados.

Esta documentação sobre o ONTAP e o banco de dados PostgreSQL substitui o banco de dados *TR-4770: PostgreSQL publicado anteriormente sobre as melhores práticas do ONTAP.*

À medida que os dados crescem exponencialmente, o gerenciamento de dados se torna mais complexo para as empresas. Essa complexidade aumenta os custos de licenciamento, operação, suporte e manutenção. Para reduzir o TCO geral, considere mudar de bancos de dados comerciais para de código aberto com armazenamento de back-end confiável e de alto desempenho.

O ONTAP é uma plataforma ideal porque o ONTAP é literalmente projetado para bancos de dados. Vários recursos, como otimizações aleatórias de latência de e/S para qualidade avançada de serviço (QoS) e funcionalidade básica do FlexClone, foram criados especificamente para atender às necessidades dos workloads de banco de dados.

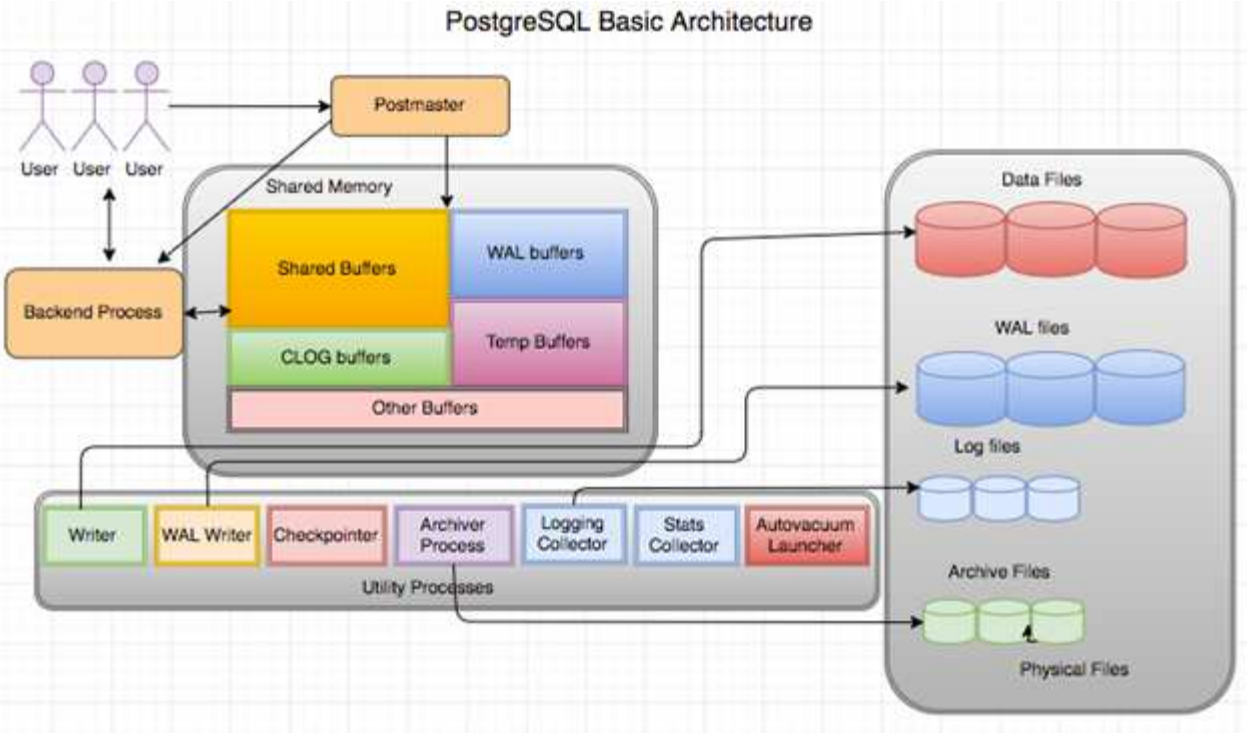
Recursos adicionais, como atualizações sem interrupções, (incluindo substituição de storage) garantem que seus bancos de dados essenciais permaneçam disponíveis. Você também pode ter recuperação instantânea de desastres para ambientes grandes por meio do MetroCluster ou selecionar bancos de dados usando o SnapMirror active Sync.

Mais importante ainda, o ONTAP oferece desempenho incomparável com a capacidade de dimensionar a solução para suas necessidades exclusivas. Nossos sistemas high-end podem fornecer mais de 1M IOPS com latências medidas em microssegundos. No entanto, se você só precisa de 100K IOPS, pode redimensionar sua solução de storage com uma controladora menor que ainda executa o mesmo sistema operacional de storage.

Configuração do banco de dados

Arquitetura

PostgreSQL é um RDBMS baseado na arquitetura cliente e servidor. Uma instância do PostgreSQL é conhecida como um cluster de banco de dados, que é uma coleção de bancos de dados em vez de uma coleção de servidores.



Há três elementos principais em um banco de dados PostgreSQL: O postmaster, o front-end (cliente) e o back-end O cliente envia solicitações para o postmaster com informações como protocolo IP e a que banco de dados se conetar. O postmaster autentica a conexão e a passa para o processo de back-end para comunicação adicional. O processo back-end executa a consulta e envia resultados diretamente para o front-end (cliente).

Uma instância do PostgreSQL é baseada em um modelo multiprocesso em vez de um modelo multithread. Ele gera vários processos para diferentes trabalhos, e cada processo tem sua própria funcionalidade. Os principais processos incluem o processo do cliente, o processo do escritor DE WAL, o processo do escritor de fundo, e o processo do checkpointer:

- Quando um processo de cliente (primeiro plano) envia solicitações de leitura ou gravação para a instância do PostgreSQL, ele não lê ou grava dados diretamente no disco. Primeiro, ele armazena os dados em buffers compartilhados e buffers de log write-ahead (WAL).
- Um processo de escritor WAL manipula o conteúdo dos buffers compartilhados e buffers WAL para escrever nos logs DO WAL. Os logs WAL são normalmente Registros de transações do PostgreSQL e são sequencialmente escritos. Portanto, para melhorar o tempo de resposta do banco de dados, o PostgreSQL primeiro grava nos logs de transação e reconhece o cliente.
- Para colocar o banco de dados em um estado consistente, o processo de gravação em segundo plano verifica o buffer compartilhado periodicamente para páginas sujas. Em seguida, ele limpa os dados para os arquivos de dados armazenados em volumes NetApp ou LUNs.
- O processo de checkpointer também é executado periodicamente (com menos frequência do que o processo em segundo plano) e impede qualquer modificação nos buffers. Ele sinaliza para o processo de gravação WAL para escrever e lavar o Registro do ponto de verificação para o final dos logs DO WAL que são armazenados no disco NetApp. Ele também sinaliza o processo de gravação em segundo plano para escrever e lavar todas as páginas sujas para o disco.

Parâmetros de inicialização

Você cria um novo cluster de banco de dados usando o `initdb` programa. `initdb`Um

script cria os arquivos de dados, tabelas do sistema e bancos de dados de modelos (template0 e template1) que definem o cluster.

A base de dados de modelos representa uma base de dados de stock. Ele contém definições para tabelas de sistema, exibições padrão, funções e tipos de dados. `pgdata` atua como um argumento para o `initdb` script que especifica a localização do cluster do banco de dados.

Todos os objetos de banco de dados no PostgreSQL são gerenciados internamente pelos respetivos OIDs. Tabelas e índices também são gerenciados por OIDs individuais. As relações entre objetos de banco de dados e seus respetivos OIDs são armazenadas em tabelas de catálogo de sistema apropriadas, dependendo do tipo de objeto. Por exemplo, OIDs de bancos de dados e tabelas de heap são armazenados em `pg_database` e 'pg_class, respetivamente. Você pode determinar os OIDs emitindo consultas no cliente PostgreSQL.

Cada banco de dados tem suas próprias tabelas individuais e arquivos de índice que são restritos a 1GB. Cada tabela tem dois arquivos associados, sufixos respetivamente com `_fsm` e `_vm`. Eles são referidos como o mapa de espaço livre e o mapa de visibilidade. Esses arquivos armazenam as informações sobre a capacidade de espaço livre e têm visibilidade em cada página no arquivo de tabela. Os índices têm apenas mapas de espaço livre individuais e não têm mapas de visibilidade.

O `pg_xlog/pg_wal` diretório contém os registos de escrita antecipada. Os logs write-ahead são usados para melhorar a confiabilidade e o desempenho do banco de dados. Sempre que você atualizar uma linha em uma tabela, o PostgreSQL primeiro grava a alteração no log write-ahead e, mais tarde, grava as modificações nas páginas de dados reais em um disco. O `pg_xlog` diretório geralmente contém vários arquivos, mas initdb cria apenas o primeiro. Arquivos extras são adicionados conforme necessário. Cada arquivo xlog tem 16MBcm de comprimento.

Definições

Existem várias configurações de ajuste do PostgreSQL que podem melhorar o desempenho.

Os parâmetros mais utilizados são os seguintes:

- `max_connections = <num>`: O número máximo de conexões de banco de dados a ter ao mesmo tempo. Use este parâmetro para restringir a troca para o disco e eliminar o desempenho. Dependendo do requisito do aplicativo, você também pode ajustar esse parâmetro para as configurações do pool de conexões.
- `shared_buffers = <num>`: O método mais simples para melhorar o desempenho do seu servidor de banco de dados. O padrão é baixo para a maioria dos hardwares modernos. Ele é definido durante a implantação para aproximadamente 25% da RAM disponível no sistema. Essa configuração de parâmetro varia dependendo de como ele funciona com instâncias de banco de dados específicas; você pode ter que aumentar e diminuir os valores por tentativa e erro. No entanto, a configuração alta pode degradar o desempenho.
- `effective_cache_size = <num>`: Este valor diz ao otimizador do PostgreSQL quanta memória o PostgreSQL tem disponível para armazenar dados em cache e ajuda a determinar se deve usar um índice. Um valor maior aumenta a probabilidade de usar um índice. Este parâmetro deve ser definido para a quantidade de memória alocada para `shared_buffers` mais a quantidade de cache de SO disponível. Muitas vezes, esse valor é mais de 50% da memória total do sistema.
- `work_mem = <num>`: Este parâmetro controla a quantidade de memória a ser usada em operações de classificação e tabelas de hash. Se você fizer uma classificação pesada em seu aplicativo, talvez seja necessário aumentar a quantidade de memória, mas tenha cuidado. Não é um parâmetro amplo do

sistema, mas um parâmetro por operação. Se uma consulta complexa tiver várias operações de ordenação nela, ela usará várias unidades de memória work_mem, e vários back-ends poderiam estar fazendo isso simultaneamente. Essa consulta pode muitas vezes levar o servidor de banco de dados a trocar se o valor for muito grande. Essa opção foi anteriormente chamada sort_mem em em versões mais antigas do PostgreSQL.

- `fsync = <boolean> (on or off)`: Este parâmetro determina se todas as suas páginas WAL devem ser sincronizadas com o disco usando fsync() antes que uma transação seja confirmada. Desligá-lo às vezes pode melhorar o desempenho de gravação e ativá-lo aumenta a proteção contra o risco de corrupção quando o sistema trava.
- `checkpoint_timeout`: O processo de checkpoint elimina dados comprometidos no disco. Isso envolve muitas operações de leitura/gravação no disco. O valor é definido em segundos e valores mais baixos diminuem o tempo de recuperação de falhas e o aumento de valores pode reduzir a carga nos recursos do sistema reduzindo as chamadas de ponto de verificação. Dependendo da criticidade do aplicativo, uso, disponibilidade do banco de dados, defina o valor de checkpoint_timeout.
- `commit_delay = <num>` E `commit_siblings = <num>`: essas opções são usadas em conjunto para ajudar a melhorar o desempenho, escrevendo várias transações que estão cometendo de uma só vez. Se houver vários objetos commit_siblings ativos no momento em que a transação está sendo feita, o servidor aguarda por commit_delay microssegundos para tentar cometer várias transações de uma só vez.
- `max_worker_processes / max_parallel_workers`: Configurar o número ideal de trabalhadores para processos. Max_Parallel_workers corresponde ao número de CPUs disponíveis. Dependendo do design do aplicativo, as consultas podem exigir um número menor de trabalhadores para operações paralelas. É melhor manter o valor para ambos os parâmetros o mesmo, mas ajustar o valor após o teste.
- `random_page_cost = <num>`: Este valor controla a forma como o PostgreSQL visualiza leituras de discos não sequenciais. Um valor maior significa que o PostgreSQL é mais provável de usar uma varredura sequencial em vez de uma varredura de índice, indicando que seu servidor tem discos rápidos Modificar esta configuração depois de avaliar outras opções como otimização baseada em plano, aspiração, indexação para alterar consultas ou esquema.
- `effective_io_concurrency = <num>`: Este parâmetro define o número de operações de e/S de disco simultâneas que o PostgreSQL tenta executar simultaneamente. Aumentar esse valor aumenta o número de operações de e/S que qualquer sessão individual do PostgreSQL tenta iniciar em paralelo. O intervalo permitido é de 1 a 1.000, ou zero para desativar a emissão de solicitações de e/S assíncronas. Atualmente, essa configuração afeta somente digitalizações de heap bitmap. As unidades de estado sólido (SSDs) e outro storage baseado em memória (NVMe) costumam processar muitas solicitações simultâneas. Assim, o melhor valor pode ser nas centenas.

Consulte a documentação do PostgreSQL para obter uma lista completa dos parâmetros de configuração do PostgreSQL.

BRINDE

TOAST representa a técnica de armazenamento de atributos oversized. PostgreSQL usa um tamanho de página fixo (geralmente 8KB) e não permite que tuplas abranjam várias páginas. Portanto, não é possível armazenar grandes valores de campo diretamente. Quando você tenta armazenar uma linha que excede esse tamanho, O TOAST divide os dados de colunas grandes em "pedaços" menores e os armazena em uma MESA DE BRINDE.

Os grandes valores dos atributos tostados são retirados (se selecionados) apenas no momento em que o conjunto de resultados é enviado ao cliente. A tabela em si é muito menor e pode caber mais linhas no cache de buffer compartilhado do que poderia sem qualquer armazenamento fora de linha (TOAST).

VÁCUO

Na operação normal do PostgreSQL, tuplas que são excluídas ou tornadas obsoletas por uma atualização não são removidas fisicamente de sua tabela; elas permanecem presentes até que O VÁCUO seja executado. Portanto, você deve executar O VÁCUO periodicamente, especialmente em tabelas atualizadas com frequência. O espaço que ocupa deve ser recuperado para reutilização por novas linhas, para evitar a interrupção do espaço em disco. No entanto, ele não retorna o espaço para o sistema operacional.

O espaço livre dentro de uma página não é fragmentado. VACUUM reescreve todo o bloco, empacotando eficientemente as linhas restantes e deixando um único bloco contíguo de espaço livre em uma página.

Em contraste, O VACUUM FULL compacta ativamente as tabelas escrevendo uma versão completamente nova do arquivo de tabela sem espaço morto. Esta ação minimiza o tamanho da mesa, mas pode levar muito tempo. Ele também requer espaço em disco extra para a nova cópia da tabela até que a operação seja concluída. O objetivo do VÁCUO de rotina é evitar a atividade TOTAL DO VÁCUO. Este processo não só mantém as tabelas em seu tamanho mínimo, mas também mantém o uso em estado estável do espaço em disco.

Tablespaces

Dois espaços de tablespaces são criados automaticamente quando o cluster de banco de dados é inicializado.

O `pg_global` espaço de tabela é usado para catálogos de sistemas compartilhados. O `pg_default` espaço de tabela é o espaço de tabela padrão dos bancos de dados template1 e template0. Se a partição ou o volume em que o cluster foi inicializado ficar sem espaço e não puder ser estendido, um espaço de tabela pode ser criado em uma partição diferente e usado até que o sistema possa ser reconfigurado.

Um índice que é muito usado pode ser colocado em um disco rápido e altamente disponível, como um dispositivo de estado sólido. Além disso, uma tabela que armazena dados arquivados que raramente são usados ou não críticos ao desempenho pode ser armazenada em um sistema de disco mais lento e menos caro, como unidades SAS ou SATA.

As tablespaces são uma parte do cluster do banco de dados e não podem ser tratadas como uma coleção autônoma de arquivos de dados. Eles dependem dos metadados contidos no diretório de dados principal e, portanto, não podem ser anexados a um cluster de banco de dados diferente ou fazer backup individualmente. Da mesma forma, se você perder um espaço de tabela (por meio da exclusão de arquivos, falha de disco, etc.), o cluster do banco de dados pode se tornar ilegível ou incapaz de iniciar. Colocar um espaço de tabela em um sistema de arquivos temporário como um disco RAM corre o risco de a confiabilidade de todo o cluster.

Depois que ele é criado, um espaço de tabela pode ser usado a partir de qualquer banco de dados se o usuário solicitante tiver Privileges suficiente. PostgreSQL usa links simbólicos para simplificar a implementação de espaços de tabela. O PostgreSQL adiciona uma linha à `pg_tablespace` tabela (uma tabela abrangente) e atribui um novo identificador de objeto (OID) a essa linha. Finalmente, o servidor usa o OID para criar um link simbólico entre o cluster e o diretório fornecido. O diretório `$PGDATA/pg_tblspc` contém links simbólicos que apontam para cada um dos espaços de tablespaces não incorporados definidos no cluster.

Configuração de armazenamento

NFS

Os bancos de dados PostgreSQL podem ser hospedados em sistemas de arquivos

NFSv3 ou NFSv4. A melhor opção depende de fatores fora do banco de dados.

Por exemplo, o comportamento de bloqueio NFSv4D pode ser preferível em determinados ambientes em cluster. ("aqui"Consulte para obter detalhes adicionais)

A funcionalidade do banco de dados deve estar próxima da mesma, incluindo o desempenho. O único requisito é o uso da `hard` opção de montagem. Isso é necessário para garantir que os tempos limite flexíveis não produzam erros de e/S irrecuperáveis.

Se NFSv4 for escolhido como um protocolo, a NetApp recomenda o uso de NFSv4,1. Há algumas melhorias funcionais no protocolo NFSv4 em NFSv4,1 que aumentam a resiliência em relação a NFSv4,0.

Use as seguintes opções de montagem para cargas de trabalho gerais de banco de dados:

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

Se for esperado uma e/S sequencial grande, os tamanhos de transferência NFS podem ser aumentados conforme descrito na seção a seguir.

Tamanhos de transferência NFS

Por padrão, o ONTAP limita os tamanhos de e/S de NFS a 64K.

A e/S aleatória com a maioria dos aplicativos e bancos de dados usa um tamanho de bloco muito menor que está bem abaixo do máximo 64K. A e/S de bloco grande geralmente é paralelizada, portanto o máximo de 64K GB também não é uma limitação para obter largura de banda máxima.

Existem algumas cargas de trabalho em que o máximo 64K cria uma limitação. Em particular, operações de um único processo, como operação de backup ou recuperação ou uma verificação de tabela completa de banco de dados, são executadas de forma mais rápida e eficiente se o banco de dados puder executar menos, mas maiores I/os. O tamanho ideal de manuseio de e/S para ONTAP é 256K.

O tamanho máximo de transferência para um determinado SVM do ONTAP pode ser alterado da seguinte forma:

```
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::*>
```

Nunca diminua o tamanho máximo de transferência permitido no ONTAP abaixo do valor de rsize/wsize dos sistemas de arquivos NFS atualmente instalados. Isso pode criar pendências ou até mesmo corrupção de dados com alguns sistemas operacionais. Por exemplo, se os clientes NFS estiverem atualmente definidos em um rsize/wsize de 65536, o tamanho máximo de transferência do ONTAP poderá ser ajustado entre 65536 e 1048576 sem efeito porque os próprios clientes são limitados. Reduzir o tamanho máximo de transferência abaixo de 65536 pode danificar a disponibilidade ou os dados.

Uma vez que o tamanho da transferência é aumentado no nível ONTAP, as seguintes opções de montagem seriam usadas:

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

NFSv3 Tabelas de slot TCP

Se NFSv3 for usado com Linux, é fundamental definir corretamente as tabelas de slot TCP.

As tabelas de slot TCP são equivalentes a NFSv3 mm de profundidade de fila do adaptador de barramento do host (HBA). Essas tabelas controlam o número de operações NFS que podem ficar pendentes de uma só vez. O valor padrão é geralmente 16, o que é muito baixo para um desempenho ideal. O problema oposto ocorre em kernels Linux mais recentes, que podem aumentar automaticamente o limite da tabela de slots TCP para um nível que satura o servidor NFS com solicitações.

Para um desempenho ideal e para evitar problemas de desempenho, ajuste os parâmetros do kernel que controlam as tabelas de slots TCP.

Executar o `sysctl -a | grep tcp.*.slot_table` comando e respeitar os seguintes parâmetros:

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

Todos os sistemas Linux devem incluir `sunrpc.tcp_slot_table_entries`, mas apenas alguns incluem `sunrpc.tcp_max_slot_table_entries`. Ambos devem ser definidos para 128.

> A falha em definir esses parâmetros pode ter efeitos significativos no desempenho. Em alguns casos, o desempenho é limitado porque o sistema operacional linux não está emitindo e/S suficiente Em outros casos, as latências de e/S aumentam à medida que o sistema operacional linux tenta emitir mais e/S do que pode ser reparado.

SAN

Os bancos de dados PostgreSQL com SAN geralmente são hospedados em sistemas de arquivos xfs, mas outros podem ser usados se suportados pelo fornecedor do sistema operacional

Embora um único LUN possa geralmente suportar até 100K IOPS, os bancos de dados com uso intenso de e/S geralmente exigem o uso de LVM com striping.

LVM Striping

Antes da era das unidades flash, a distribuição foi usada para ajudar a superar as limitações de desempenho das unidades giratórias. Por exemplo, se um sistema operacional precisar executar uma operação de leitura 1MB, ler que 1MB TB de dados de uma única unidade exigiria muita busca e leitura de cabeça de unidade, pois o 1MB é transferido lentamente. Se esse 1MB TB de dados fosse distribuído em 8 LUNs, o sistema operacional poderia emitir oito operações de leitura 128K em paralelo e reduzir o tempo necessário para concluir a transferência 1MB.

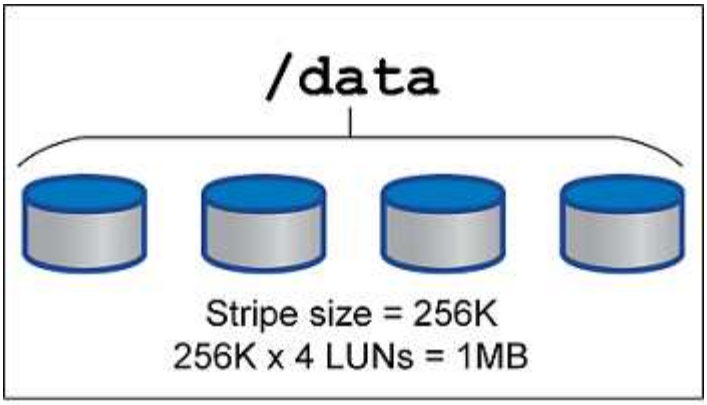
Riscar com unidades giratórias foi mais difícil porque o padrão de e/S tinha que ser conhecido com antecedência. Se o striping não foi ajustado corretamente para os padrões de e/S verdadeiros, as configurações listradas podem danificar o desempenho. Com os bancos de dados Oracle e, especialmente, com configurações all-flash, a distribuição é muito mais fácil de configurar e, comprovadamente, aumentou significativamente o desempenho.

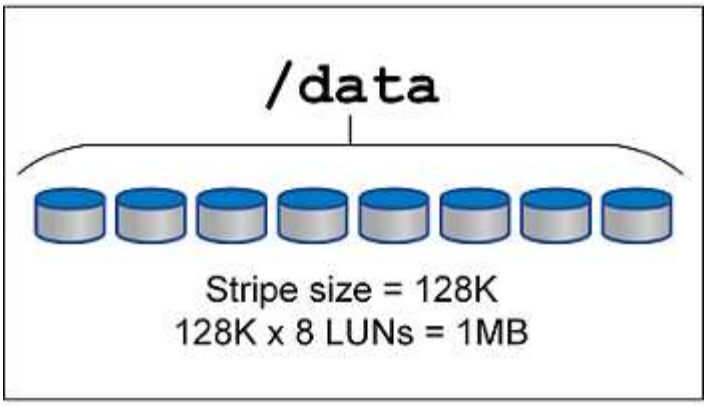
Gerenciadores de volume lógicos, como o Oracle ASM stripe por padrão, mas o LVM de SO nativo não. Alguns deles unem vários LUNs como um dispositivo concatenado, o que resulta em datafiles que existem em um e apenas um dispositivo LUN. Isso causa pontos quentes. Outras implementações de LVM são padrão para extensões distribuídas. Isso é semelhante ao striping, mas é mais grosseiro. Os LUNs no grupo de volumes são cortados em pedaços grandes, chamados de extensões e normalmente medidos em muitos megabytes, e os volumes lógicos são então distribuídos por essas extensões. O resultado é a e/S aleatória contra um arquivo deve ser bem distribuída entre LUNs, mas as operações de e/S sequenciais não são tão eficientes quanto poderiam ser.

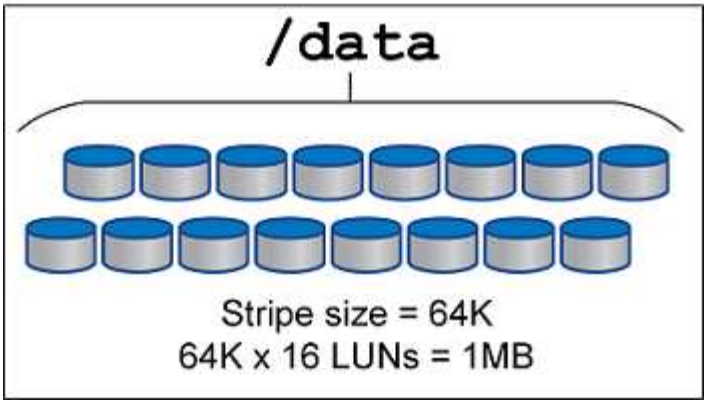
A e/S de aplicativos intensivos quase sempre é (a) em unidades do tamanho básico do bloco ou (b) um megabyte.

O principal objetivo de uma configuração de distribuição é garantir que a e/S de arquivo único possa ser executada como uma única unidade, e as e/S de vários blocos, que devem ter 1MB MB de tamanho, possam ser paralelizadas uniformemente em todos os LUNs no volume distribuído. Isso significa que o tamanho do stripe não deve ser menor que o tamanho do bloco do banco de dados e o tamanho do stripe multiplicado pelo número de LUNs deve ser 1MB.

A figura a seguir mostra três opções possíveis para o tamanho da faixa e ajuste de largura. O número de LUNs é selecionado para atender aos requisitos de performance conforme descrito acima, mas, em todos os casos, o total de dados em um único stripe é de 1MB.







Proteção de dados

Proteção Ddta nativa

Um dos principais aspetos do design de armazenamento é habilitar a proteção para volumes PostgreSQL. Os clientes podem proteger seus bancos de dados PostgreSQL usando a abordagem dump ou usando backups do sistema de arquivos. Esta seção explica as diferentes abordagens de backup de bancos de dados individuais ou de todo o cluster.

Existem três abordagens para fazer backup de dados PostgreSQL:

- Despejo do SQL Server
- Backup no nível do sistema de arquivo

- Arquivamento contínuo

A ideia por trás do método de despejo do SQL Server é gerar um arquivo com comandos do SQL Server que, quando retornados ao servidor, pode recriar o banco de dados como era no momento do despejo. PostgreSQL fornece os programas utilitários `pg_dump` e `pg_dump_all` para criar backup individual e em nível de cluster. Esses despejos são lógicos e não contêm informações suficientes para serem usados pelo replay WAL.

Uma estratégia alternativa de backup é usar o backup em nível de sistema de arquivos, no qual os administradores copiam diretamente os arquivos que o PostgreSQL usa para armazenar os dados no banco de dados. Este método é feito no modo offline: O banco de dados ou cluster deve ser desligado. Outra alternativa é usar `pg_basebackup` para executar o backup de streaming quente do banco de dados PostgreSQL.

Instantâneos

Os backups baseados em snapshot com PostgreSQL exigem a configuração de snapshots para datafiles, arquivos WAL e ARQUIVOS WAL arquivados para fornecer recuperação completa ou pontual.

Para bancos de dados PostgreSQL, o tempo médio de backup com snapshots está no intervalo de alguns segundos a alguns minutos. Essa velocidade de backup é 60 a 100 vezes mais rápida do que `pg_basebackup` outras abordagens de backup baseadas no sistema de arquivos.

Os snapshots no storage do NetApp podem ser consistentes com falhas e com aplicações. Um snapshot consistente com falhas é criado no storage sem silenciar o banco de dados, enquanto um snapshot consistente com aplicativos é criado enquanto o banco de dados está no modo de backup. O NetApp também garante que os snapshots subsequentes sejam backups incrementais para sempre, a fim de promover a economia de storage e a eficiência da rede.

Como os snapshots são rápidos e não afetam o desempenho do sistema, você pode agendar vários snapshots diariamente em vez de criar um único backup diário, como acontece com outras tecnologias de backup de streaming. Quando uma operação de restauração e recuperação é necessária, o tempo de inatividade do sistema é reduzido por dois recursos principais:

- A tecnologia de recuperação de dados NetApp SnapRestore significa que a operação de restauração é executada em segundos.
- Objetivos agressivos do ponto de restauração (RPOs) significam que menos logs do banco de dados devem ser aplicados e a recuperação avançada também é acelerada.

Para fazer backup do PostgreSQL, você deve garantir que os volumes de dados sejam protegidos simultaneamente com O WAL (Consistency-group) e os logs arquivados. Enquanto você estiver usando a tecnologia Snapshot para copiar ARQUIVOS WAL, certifique-se de executar `pg_stop` para liberar todas as entradas WAL que devem ser arquivadas. Se você lavar as ENTRADAS WAL durante a restauração, precisará apenas parar o banco de dados, desmontar ou excluir o diretório de dados existente e executar uma operação SnapRestore no armazenamento. Depois que a restauração for concluída, você pode montar o sistema e trazê-lo de volta ao seu estado atual. Para recuperação pontual, você também pode restaurar logs DO WAL e do arquivo; em seguida, o PostgreSQL decide o ponto mais consistente e recupera-o automaticamente.

Os grupos de consistência são um recurso no ONTAP e são recomendados quando há vários volumes montados em uma única instância ou em um banco de dados com vários espaços de tabela. Um snapshot de grupo de consistência garante que todos os volumes sejam agrupados e protegidos. Um grupo de

consistência pode ser gerenciado de forma eficiente a partir do Gerenciador de sistemas do ONTAP e até mesmo cloná-lo para criar uma cópia de instância de um banco de dados para fins de teste ou desenvolvimento.

Software de proteção de dados

O plugin NetApp SnapCenter para banco de dados PostgreSQL, combinado com as tecnologias Snapshot e NetApp FlexClone, oferece benefícios como:

- Backup e restauração rápidos.
- Clones com uso eficiente de espaço.
- A capacidade de construir um sistema de recuperação de desastres rápido e eficaz.

Você pode preferir escolher os parceiros premium de backup da NetApp, como o Veeam Software e a CommVault, nas seguintes circunstâncias:

- Gerenciamento de workloads em um ambiente heterogêneo
- Armazenamento de backups na nuvem ou em fita para retenção de longo prazo
- Suporte para uma ampla gama de versões e tipos de SO

O plugin SnapCenter para PostgreSQL é um plugin compatível com a comunidade e a configuração e documentação estão disponíveis na loja de automação NetApp. Com o SnapCenter, o usuário pode fazer backup do banco de dados, clonar e restaurar dados remotamente.

Informações sobre direitos autorais

Informações sobre marcas comerciais