PostgreSQL

Enterprise applications

NetApp
January 02, 2026

Sommario

PostgreSQL

Panoramica

PostgreSQL viene fornito con varianti che includono PostgreSQL, PostgreSQL Plus ed EDB Postgres Advanced Server (ECAS). PostgreSQL viene in genere distribuito come database back-end per applicazioni multi-Tier. È supportato da pacchetti middleware comuni (come PHP, Java, Python, Tcl/TK, ODBC, E JDBC) ed è stata storicamente una scelta popolare per i sistemi di gestione di database open-source. ONTAP è una scelta eccellente per l'esecuzione di database PostgreSQL per la sua affidabilità, prestazioni elevate ed efficienza di gestione dei dati.

Questa documentazione su ONTAP e il database PostgreSQL sostituisce il database *TR-4770: PostgreSQL precedentemente pubblicato sulle Best practice di ONTAP.*

Con la crescita esponenziale dei dati, la gestione dei dati diventa più complessa per le aziende. Questa complessità aumenta i costi di licenza, operativi, di supporto e di manutenzione. Per ridurre il TCO complessivo, considerare il passaggio da database commerciali a open-source con storage back-end affidabile e dalle performance elevate.

ONTAP è una piattaforma ideale, perché ONTAP è letteralmente progettato per i database. Sono state create numerose funzionalità come le ottimizzazioni della latenza io random per la qualità del servizio avanzata fino alle funzionalità FlexClone di base per rispondere specificamente alle esigenze dei carichi di lavoro dei database.

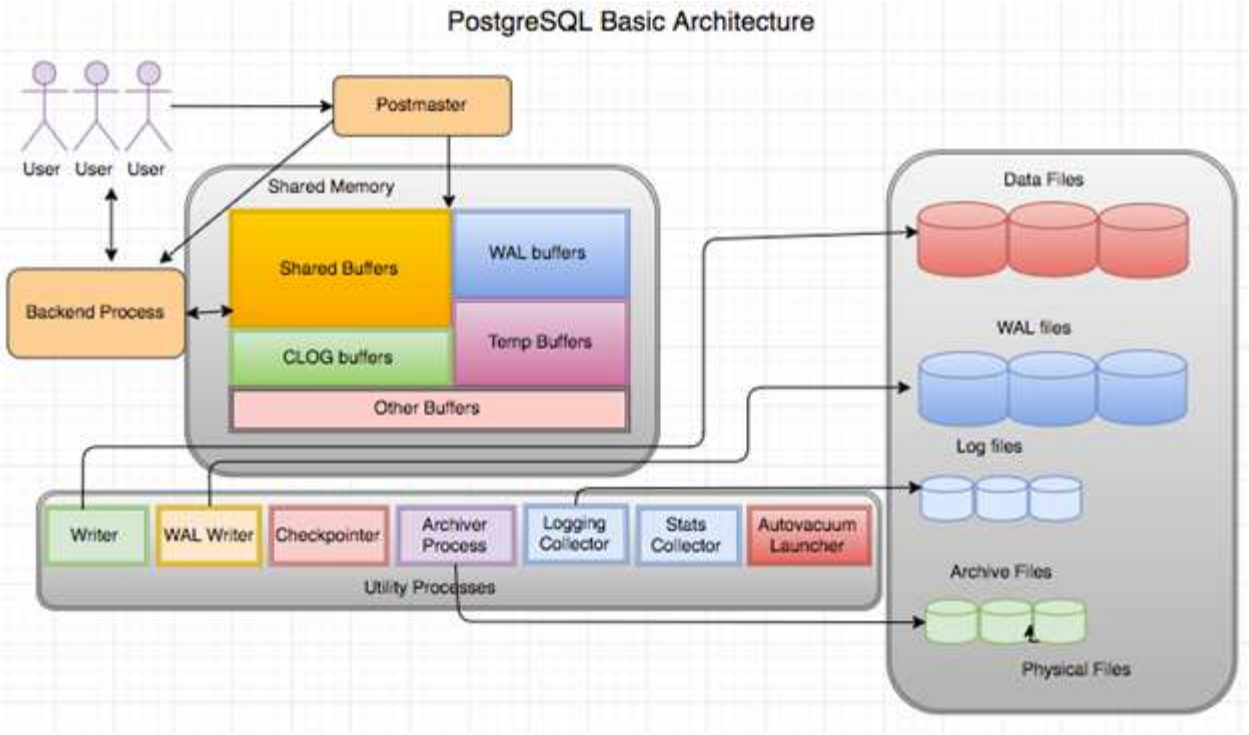
Funzioni aggiuntive come gli aggiornamenti senza interruzioni, (inclusa la sostituzione dello storage) garantiscono la disponibilità dei database critici. Puoi anche disporre di un disaster recovery istantaneo per ambienti di grandi dimensioni tramite MetroCluster o selezionare database tramite la sincronizzazione attiva di SnapMirror.

Soprattutto, ONTAP offre prestazioni senza pari con la possibilità di dimensionare la soluzione in base alle proprie esigenze specifiche. I nostri sistemi high-end possono offrire oltre 1M IOPS con latenze misurate in microsecondi, ma se ti servono solo 100K IOPS, puoi dimensionare al meglio la tua soluzione storage con un controller più piccolo che esegue ancora lo stesso sistema operativo per lo storage.

Configurazione del database

Architettura

PostgreSQL è un RDBMS basato su architettura client e server. Un'istanza di PostgreSQL è nota come cluster di database, ovvero una raccolta di database anziché una raccolta di server.



Un database PostgreSQL contiene tre elementi principali: Il postmaster, il front-end (client) e il back-end Il client invia richieste al postmaster con informazioni quali il protocollo IP e il database a cui connettersi. Il postmaster autentica la connessione e la passa al processo back-end per ulteriori comunicazioni. Il processo back-end esegue la query e invia i risultati direttamente al front-end (client).

Un'istanza PostgreSQL si basa su un modello multiprocesso anziché su un modello multithread. Genera più processi per diversi processi e ogni processo ha una propria funzionalità. I processi principali includono il processo client, il processo di scrittura WAL, il processo di scrittura in background e il processo di checkpointer:

- Quando un processo client (in primo piano) invia richieste di lettura o scrittura all'istanza PostgreSQL, non legge o scrive dati direttamente sul disco. Innanzitutto, memorizza i dati nei buffer condivisi e nei buffer WAL (Write-ahead logging).
- Un processo di scrittura WAL manipola il contenuto dei buffer condivisi e dei buffer WAL da scrivere nei registri WAL. I registri WAL sono in genere registri di transazioni di PostgreSQL e vengono scritti in sequenza. Pertanto, per migliorare i tempi di risposta dal database, PostgreSQL scrive prima nei registri delle transazioni e riconosce il client.
- Per impostare il database in uno stato coerente, il processo di scrittura in background verifica periodicamente la presenza di pagine sporche nel buffer condiviso. Quindi, scarica i dati sui file di dati che sono memorizzati su volumi NetApp o LUN.
- Anche il processo checkpointer viene eseguito periodicamente (meno frequentemente del processo in background) e impedisce qualsiasi modifica ai buffer. Segnala al processo di scrittura WAL di scrivere e svuotare il record del punto di verifica alla fine dei registri WAL memorizzati sul disco NetApp. Segnala inoltre al processo di scrittura in background di scrivere e scaricare tutte le pagine sporche sul disco.

Parametri di inizializzazione

È possibile creare un nuovo cluster di database utilizzando `initdb` programma. An `initdb` script crea i file di dati, le tabelle di sistema e i database dei modelli (template0 e template1) che definiscono il cluster.

Il database dei modelli rappresenta un database di stock. Contiene le definizioni per le tabelle di sistema, le viste standard, le funzioni e i tipi di dati. `pgdata` funge da argomento per il `initdb` script che specifica la posizione del cluster di database.

Tutti gli oggetti di database in PostgreSQL sono gestiti internamente dai rispettivi OID. Le tabelle e gli indici sono inoltre gestiti da singoli OID. Le relazioni tra gli oggetti del database e i rispettivi OID vengono memorizzate in tabelle di cataloghi di sistema appropriate, a seconda del tipo di oggetto. Ad esempio, gli OID dei database e delle tabelle heap vengono memorizzati in `pg_database` e `pg_class, rispettivamente. È possibile determinare gli OID eseguendo query sul client PostgreSQL.

Ogni database ha le proprie tabelle e i file di indice che sono limitati a 1GB. Ogni tabella ha due file associati, rispettivamente con il suffisso `_fsm` e. `_vm`. Sono indicate come mappa dello spazio libero e mappa di visibilità. Questi file memorizzano le informazioni sulla capacità di spazio libero e hanno visibilità su ogni pagina del file di tabella. Gli indici hanno solo mappe di spazio libero individuali e non hanno mappe di visibilità.

Il `pg_xlog/pg_wal` la directory contiene i registri write-ahead. I registri write-ahead sono utilizzati per migliorare l'affidabilità e le performance del database. Ogni volta che si aggiorna una riga in una tabella, PostgreSQL scrive prima la modifica nel registro write-ahead e successivamente scrive le modifiche alle pagine di dati effettive su un disco. Il `pg_xlog` la directory di solito contiene diversi file, ma initdb crea solo il primo. I file aggiuntivi vengono aggiunti in base alle necessità. Ciascun file xlog è lungo 16MB MB.

Impostazioni

Esistono diverse configurazioni di ottimizzazione PostgreSQL che possono migliorare le prestazioni.

I parametri più comunemente utilizzati sono i seguenti:

- `max_connections = <num>`: Il numero massimo di connessioni al database da avere contemporaneamente. Utilizzare questo parametro per limitare lo scambio sul disco e l'interruzione delle prestazioni. A seconda delle esigenze dell'applicazione, è anche possibile regolare questo parametro per le impostazioni del pool di connessione.
- `shared_buffers = <num>`: Il metodo più semplice per migliorare le prestazioni del server di database. Il valore predefinito è basso per la maggior parte dei componenti hardware moderni. Durante l'implementazione viene impostato su circa il 25% della RAM disponibile sul sistema. Questa impostazione di parametro varia in base al funzionamento con determinate istanze di database; potrebbe essere necessario aumentare e diminuire i valori per prova ed errore. Tuttavia, l'impostazione di un livello elevato potrebbe degradare le prestazioni.
- `effective_cache_size = <num>`: Questo valore indica all'ottimizzatore di PostgreSQL la quantità di memoria disponibile per la memorizzazione nella cache dei dati e aiuta a determinare se utilizzare un indice. Un valore maggiore aumenta la probabilità di utilizzare un indice. Questo parametro deve essere impostato sulla quantità di memoria allocata a. `shared_buffers` più la quantità di cache del sistema operativo disponibile. Spesso questo valore corrisponde a più del 50% della memoria di sistema totale.
- `work_mem = <num>`: Questo parametro controlla la quantità di memoria da utilizzare nelle operazioni di ordinamento e nelle tabelle hash. Se si esegue un ordinamento pesante nell'applicazione, potrebbe essere necessario aumentare la quantità di memoria, ma prestare attenzione. Non si tratta di un parametro a livello di sistema, ma di un parametro per operazione. Se una query complessa contiene diverse operazioni di ordinamento, utilizza più unità di memoria work_mem e più backend potrebbero farlo contemporaneamente. Questa query può spesso indurre il server di database a effettuare lo swap se il valore è troppo grande. Questa opzione era precedentemente chiamata sort_mem nelle versioni precedenti di PostgreSQL.

- `fsync = <boolean> (on or off)`: Questo parametro determina se tutte le pagine WAL devono essere sincronizzate su disco utilizzando fsync() prima che venga eseguito il commit di una transazione. Disattivandolo a volte si possono migliorare le prestazioni di scrittura e attivandolo si aumenta la protezione dal rischio di danneggiamento quando il sistema si blocca.
- `checkpoint_timeout`: Il processo del punto di verifica elimina i dati sottoposti a commit sul disco. Ciò comporta numerose operazioni di lettura/scrittura su disco. Il valore è impostato in secondi e valori inferiori riducono il tempo di recupero da crash e valori crescenti possono ridurre il carico sulle risorse di sistema riducendo le chiamate al punto di verifica. In base alla criticità dell'applicazione, all'utilizzo, alla disponibilità del database, impostare il valore di checkpoint_timeout.
- `commit_delay = <num>` e. `commit_siblings = <num>`: Queste opzioni vengono utilizzate insieme per migliorare le prestazioni scrivendo più transazioni che vengono effettuate contemporaneamente. Se ci sono diversi oggetti commit_siblings attivi nel momento in cui la transazione è in fase di commit, il server attende Commit_delay microsecondi per tentare di eseguire più transazioni contemporaneamente.
- `max_worker_processes / max_parallel_workers`: Configurare il numero ottimale di lavoratori per i processi. Max_Parallel_Workers corrisponde al numero di CPU disponibili. A seconda della progettazione dell'applicazione, le query potrebbero richiedere un numero minore di lavoratori per le operazioni parallele. È meglio mantenere lo stesso valore per entrambi i parametri, ma regolare il valore dopo la verifica.
- `random_page_cost = <num>`: Questo valore controlla il modo in cui PostgreSQL visualizza le letture del disco non sequenziali. Un valore più elevato indica che PostgreSQL è più probabile che utilizzi una scansione sequenziale invece di una scansione di indice, indicando che il server dispone di dischi veloci modificare questa impostazione dopo aver valutato altre opzioni come l'ottimizzazione basata su piano, l'aspirazione, l'indicizzazione per modificare query o schemi.
- `effective_io_concurrency = <num>`: Questo parametro imposta il numero di operazioni di i/o su disco simultanee che PostgreSQL tenta di eseguire contemporaneamente. L'aumento di questo valore aumenta il numero di operazioni di i/o che una singola sessione PostgreSQL tenta di avviare in parallelo. L'intervallo consentito è compreso tra 1 e 1.000 o zero per disattivare l'emissione di richieste i/o asincrone. Attualmente, questa impostazione influisce solo sulle scansioni bitmap heap. I dischi a stato solido (SSD) e altro storage basato su memoria (NVMe) possono spesso elaborare molte richieste simultanee, cosicché il valore migliore può essere centinaia.

Consultare la documentazione di PostgreSQL per un elenco completo dei parametri di configurazione di PostgreSQL.

TOAST

TOAST è l'acronimo di OVERSIZED-Attribute Storage Technique. PostgreSQL utilizza una dimensione di pagina fissa (in genere 8KB) e non consente alle tuple di occupare più pagine. Pertanto, non è possibile memorizzare direttamente valori di campo grandi. Quando si tenta di memorizzare una riga che supera queste dimensioni, TOAST suddivide i dati delle colonne di grandi dimensioni in "pezzi" più piccoli e li memorizza in una tabella TOAST.

I valori elevati degli attributi tostati vengono estratti (se selezionati) solo quando il set di risultati viene inviato al client. La tabella stessa è molto più piccola e può contenere più righe nella cache buffer condivisa di quanto non possa fare senza alcuna archiviazione out-of-line (TOAST).

VUOTO

Nelle normali operazioni PostgreSQL, le tuple eliminate o rese obsolete da un aggiornamento non vengono fisicamente rimosse dalla tabella; rimangono presenti fino all'esecuzione di VACUUM. Pertanto, è necessario eseguire il VUOTO periodicamente, soprattutto nelle tabelle aggiornate di frequente. Lo spazio occupato deve quindi essere recuperato per essere riutilizzato da nuove righe, per evitare di esaurire lo spazio su disco. Tuttavia, non restituisce lo spazio al sistema operativo.

Lo spazio libero all'interno di una pagina non è frammentato. L'ASPIRAPOLVERE riscrive l'intero blocco, comprimendo in modo efficiente le righe rimanenti e lasciando un singolo blocco contiguo di spazio libero in una pagina.

Al contrario, VACUUM FULL comprime attivamente le tabelle scrivendo una versione completamente nuova del file di tabella senza spazio morto. Questa azione riduce al minimo le dimensioni della tabella, ma può richiedere molto tempo. Richiede inoltre ulteriore spazio su disco per la nuova copia della tabella fino al completamento dell'operazione. L'obiettivo del VUOTO DI routine è di evitare l'attività di VUOTO PIENO. Questo processo non solo mantiene le tabelle alla loro dimensione minima, ma mantiene anche l'utilizzo costante dello spazio su disco.

Tablespace

Due tablespace vengono create automaticamente al momento dell'inizializzazione del cluster di database.

Il `pg_global` tablespace viene utilizzato per i cataloghi di sistema condivisi. Il `pg_default` tablespace è la tablespace predefinita dei database template1 e template0. Se la partizione o il volume su cui il cluster è stato inizializzato esaurisce lo spazio e non può essere esteso, è possibile creare uno spazio di tabella in un'altra partizione ed utilizzarlo fino a quando il sistema non può essere riconfigurato.

Un indice molto utilizzato può essere collocato su un disco veloce e altamente disponibile, come un dispositivo a stato solido. Inoltre, una tabella che memorizza i dati archiviati utilizzati raramente o non critici per le prestazioni può essere archiviata su un sistema su disco meno costoso e più lento, come le unità SAS o SATA.

Gli spazi di tabella fanno parte del cluster di database e non possono essere trattati come una raccolta autonoma di file di dati. Dipendono dai metadati contenuti nella directory dei dati principale e pertanto non possono essere collegati a un cluster di database diverso o sottoposti a backup individuale. Analogamente, se si perde uno spazio di tabella (a causa dell'eliminazione dei file, del guasto del disco e così via), il cluster del database potrebbe diventare illeggibile o non avviarsi. Posizionando una tablespace su un file system temporaneo come un disco RAM si rischia l'affidabilità dell'intero cluster.

Una volta creato, è possibile utilizzare un tablespace da qualsiasi database se l'utente richiedente dispone di privilegi sufficienti. PostgreSQL utilizza collegamenti simbolici per semplificare l'implementazione di tablespace. PostgreSQL aggiunge una riga al `pg_tablespace` Tabella (una tavola a livello di cluster) e assegna un nuovo identificatore di oggetto (OID) a quella riga. Infine, il server utilizza l'OID per creare un collegamento simbolico tra il cluster e la directory specificata. La directory `$PGDATA/pg_tblspc` contiene collegamenti simbolici che puntano a ciascuno degli spazi di tabella non incorporati definiti nel cluster.

Configurazione dello storage

NFS

I database PostgreSQL possono essere ospitati su filesystem NFSv3 o NFSv4. L'opzione migliore dipende da fattori esterni al database.

Per esempio, il comportamento di bloccaggio di NFSv4 può essere preferibile in certi ambienti raggruppati. (Vedere "qui" per ulteriori informazioni)

In caso contrario, la funzionalità del database dovrebbe essere quasi identica, incluse le prestazioni. L'unico requisito è l'uso di `hard` opzione di montaggio. Questo è necessario per garantire che i timeout software non producano errori io irreversibili.

Se si sceglie NFSv4 come protocollo, NetApp consiglia di utilizzare NFSv4,1. Nel NFSv4,1 sono stati apportati alcuni miglioramenti funzionali al protocollo NFSv4 che migliorano la resilienza rispetto al NFSv4,0.

Utilizzare le seguenti opzioni di montaggio per i carichi di lavoro generali del database:

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

Se si prevede un io sequenziale pesante, le dimensioni del trasferimento NFS possono essere aumentate come descritto nella sezione seguente.

Dimensioni trasferimento NFS

Per impostazione predefinita, ONTAP limita le dimensioni i/o NFS a 64K.

L'i/o casuale con la maggior parte delle applicazioni e dei database utilizza blocchi di dimensioni molto inferiori, ben al di sotto del limite massimo di 64K KB. L'i/o a blocchi di grandi dimensioni è solitamente a parallelismo, pertanto anche il massimo di 64K Gbps non costituisce un limite all'ottenimento della massima larghezza di banda.

Ci sono alcuni carichi di lavoro in cui il massimo di 64K crea un limite. In particolare, le operazioni single-threaded, come l'operazione di backup o ripristino o la scansione di un database completa della tabella, vengono eseguite più velocemente e in modo più efficiente se il database è in grado di eseguire un numero di i/o inferiore ma maggiore. Le dimensioni ottimali per la gestione i/o per ONTAP sono 256K KB.

Le dimensioni massime di trasferimento per una SVM ONTAP possono essere modificate come segue:

```
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::*>
```

Non diminuire mai la dimensione massima di trasferimento consentita su ONTAP al di sotto del valore rsize/wsize dei file system NFS attualmente montati. In alcuni sistemi operativi, ciò può causare blocchi o addirittura danni ai dati. Ad esempio, se i client NFS sono attualmente impostati su un valore rsize/wsize di 65536, la dimensione massima di trasferimento ONTAP potrebbe essere regolata tra 65536 e 1048576 senza alcun effetto perché i client stessi sono limitati. La riduzione della dimensione massima di trasferimento inferiore a 65536 GB può danneggiare la disponibilità o i dati.

Una volta aumentata la dimensione di trasferimento a livello ONTAP, si utilizzeranno le seguenti opzioni di montaggio:

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

NFSv3 tabelle slot TCP

Se NFSv3 viene usato con Linux, è fondamentale impostare correttamente le tabelle degli slot TCP.

Le tabelle degli slot TCP sono l'equivalente di NFSv3 della profondità della coda degli HBA (host Bus Adapter). Queste tabelle controllano il numero di operazioni NFS che possono essere in sospeso in qualsiasi momento. Il valore predefinito è di solito 16, che è troppo basso per ottenere prestazioni ottimali. Il problema opposto si verifica sui kernel Linux più recenti, che possono aumentare automaticamente il limite della tabella degli slot TCP a un livello che satura il server NFS con le richieste.

Per prestazioni ottimali e per evitare problemi di prestazioni, regolare i parametri del kernel che controllano le tabelle degli slot TCP.

Eseguire `sysctl -a | grep tcp.*.slot_table` e osservare i seguenti parametri:

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

Tutti i sistemi Linux dovrebbero includere `sunrpc.tcp_slot_table_entries`, ma solo alcuni includono `sunrpc.tcp_max_slot_table_entries`. Entrambi devono essere impostati su 128.

> La mancata impostazione di questi parametri può avere effetti significativi sulle prestazioni. In alcuni casi, le prestazioni sono limitate poiché il sistema operativo linux non fornisce i/o sufficienti In altri casi, le latenze i/o aumentano quando il sistema operativo linux tenta di emettere più i/o di quanto possa essere gestito.

SAN

I database PostgreSQL con SAN sono generalmente ospitati su filesystem xfs, ma altri possono essere utilizzati se supportati dal fornitore del sistema operativo

Mentre un singolo LUN può generalmente supportare fino a 100K IOPS, i database io-intensive richiedono generalmente l'utilizzo di LVM con lo striping.

Striping LVM

Prima dell'era dei dischi flash, era stato utilizzato lo striping per superare i limiti di performance dei dischi rotanti. Ad esempio, se un sistema operativo deve eseguire un'operazione di lettura a 1MB bit, la lettura di 1MB GB di dati da un'unica unità richiederebbe un'ampia ricerca e lettura della testina dell'unità poiché il sistema 1MB viene trasferito lentamente. Se quei 1MB TB di dati sono stati suddivisi in 8 LUN, il sistema operativo potrebbe emettere otto operazioni di lettura 128K in parallelo, riducendo il tempo necessario per completare il trasferimento da 1MB GB.

Lo striping con dischi rotanti era più difficile perché lo schema di i/o doveva essere noto in anticipo. Se lo striping non è stato regolato correttamente per i modelli i/o reali, le configurazioni con striping potrebbero danneggiare le prestazioni. Con i database Oracle, e in particolare con le configurazioni all-flash, lo striping è molto più semplice da configurare ed è stato dimostrato che le performance risultano notevolmente migliorate.

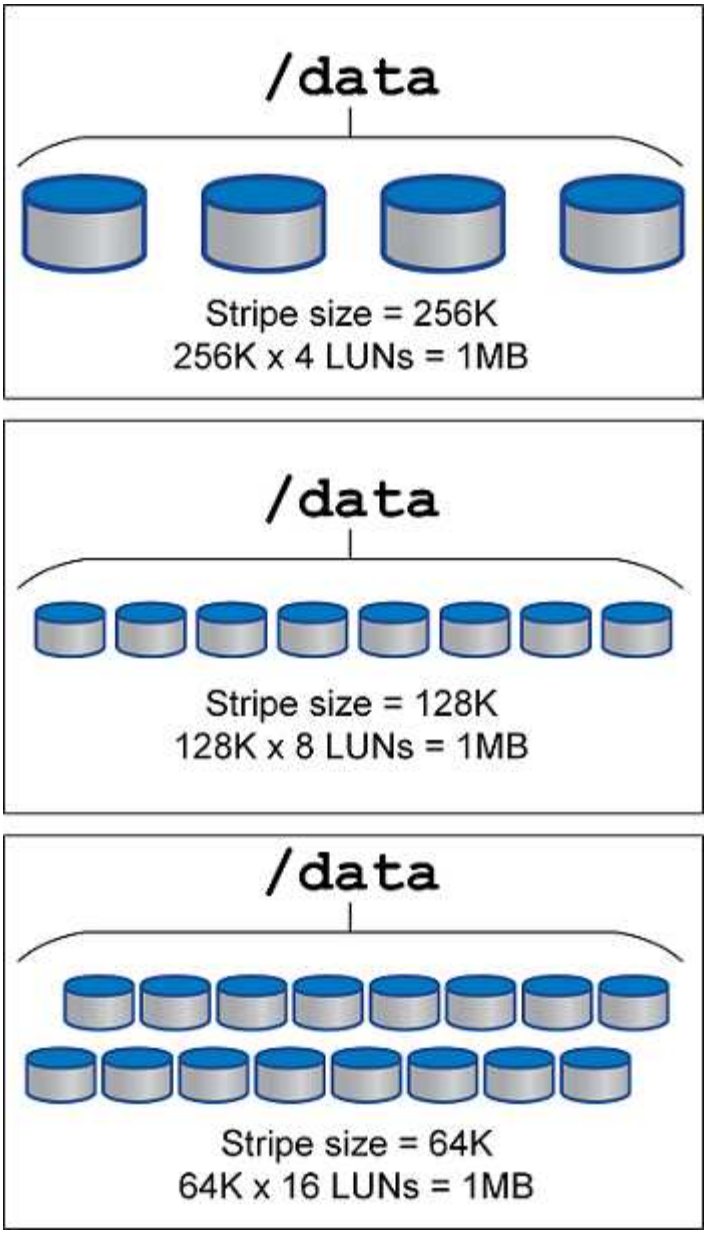
Per impostazione predefinita, i gestori di volume logici, come lo stripe di Oracle ASM, ma il sistema operativo LVM nativo non lo fanno. Alcune di esse collegano più LUN insieme come un dispositivo concatenato, il che

comporta file di dati che esistono su un solo dispositivo LUN. Ciò causa punti caldi. Le altre implementazioni LVM sono impostate per impostazione predefinita su estensioni distribuite. Questo è simile allo striping, ma è più grossolano. I LUN nel gruppo di volumi vengono suddivisi in porzioni di grandi dimensioni, chiamate estensioni e generalmente misurati in molti megabyte, e i volumi logici vengono quindi distribuiti tra tali estensioni. Il risultato è un i/o casuale per un file dovrebbe essere ben distribuito tra i LUN, ma le operazioni i/o sequenziali non sono così efficienti come potrebbero essere.

L'i/o delle applicazioni che richiedono elevate performance è quasi sempre (a) in unità delle dimensioni dei blocchi di base o (b) un megabyte.

L'obiettivo principale di una configurazione con striping è quello di garantire che l'i/o a file singolo possa essere eseguito come una singola unità, mentre l'i/o a blocchi multipli, di dimensioni pari a 1MB GB, può essere parallelizzato in modo uniforme tra tutti i LUN del volume con striping. Ciò significa che la dimensione dello stripe non deve essere inferiore alla dimensione del blocco del database e che la dimensione dello stripe moltiplicata per il numero di LUN deve essere 1MB.

La figura seguente mostra tre possibili opzioni per la regolazione delle dimensioni dello stripe e della larghezza. Il numero di LUN viene selezionato per soddisfare i requisiti di prestazioni come descritto sopra, ma in tutti i casi i dati totali all'interno di uno stripe singolo sono 1MB.



Protezione dei dati

Protezione DDTA nativa

Uno degli aspetti principali della progettazione dello storage è la protezione dei volumi PostgreSQL. I clienti possono proteggere i database PostgreSQL utilizzando l'approccio dump o i backup del file system. In questa sezione vengono illustrati i diversi approcci per il backup di singoli database o dell'intero cluster.

Sono disponibili tre approcci per il backup dei dati PostgreSQL:

- Dump di SQL Server
- Backup a livello di file system
- Archiviazione continua

L'idea alla base del metodo dump di SQL Server è generare un file con comandi di SQL Server che, quando viene restituito al server, può ricreare il database così come era al momento del dump. PostgreSQL fornisce i programmi di utilità `pg_dump` e. `pg_dump_all` per la creazione di backup singolo e a livello di cluster. Questi dump sono logici e non contengono informazioni sufficienti per essere utilizzati da WAL Replay.

Una strategia di backup alternativa consiste nell'utilizzare il backup a livello di file system, in cui gli amministratori copiano direttamente i file utilizzati da PostgreSQL per memorizzare i dati nel database. Questo metodo viene eseguito in modalità non in linea: Il database o il cluster devono essere chiusi. Un'altra alternativa è quella di utilizzare `pg_basebackup` Per eseguire il backup hot streaming del database PostgreSQL.

Snapshot

I backup basati su snapshot con PostgreSQL richiedono la configurazione di snapshot per file di dati, file WAL e file WAL archiviati per garantire un ripristino completo o point-in-time.

Per i database PostgreSQL, il tempo medio di backup con gli snapshot è compreso tra pochi secondi e pochi minuti. Questa velocità di backup è da 60 a 100 volte più veloce di `pg_basebackup` e altri approcci di backup basati sul file system.

Le snapshot sullo storage NetApp possono essere coerenti con il crash e con l'applicazione. Viene creato uno snapshot coerente con i crash sullo storage senza chiudere il database, mentre uno snapshot coerente con l'applicazione viene creato mentre il database è in modalità backup. NetApp garantisce inoltre che le snapshot successive siano backup incrementali perenni, per promuovere il risparmio dello storage e l'efficienza della rete.

Poiché le snapshot sono rapide e non influiscono sulle prestazioni del sistema, è possibile pianificare snapshot multiple ogni giorno invece di creare un unico backup giornaliero come avviene con l'altra tecnologia di backup in streaming. Quando è necessaria un'operazione di ripristino e ripristino, il downtime del sistema viene ridotto da due caratteristiche principali:

- La tecnologia di recovery di dati NetApp SnapRestore consente di eseguire l'operazione di ripristino in pochi secondi.
- Obiettivi di recovery point (RPO) aggressivi richiedono l'applicazione di un numero inferiore di log dei database e un'accelerazione del recovery in avanti.

Per eseguire il backup di PostgreSQL, è necessario assicurarsi che i volumi di dati siano protetti contemporaneamente con WAL (gruppo di coerenza) e i registri archiviati. Mentre si utilizza la tecnologia Snapshot per copiare i file WAL, assicurarsi di eseguire `pg_stop` Per svuotare tutte le voci WAL che devono essere archiviate. Se si svuotano le voci WAL durante il ripristino, sarà sufficiente arrestare il database, smontare o eliminare la directory dei dati esistente ed eseguire un'operazione SnapRestore sull'archiviazione. Al termine del ripristino, è possibile montare il sistema e riportarlo allo stato corrente. Per il ripristino point-in-time, è anche possibile ripristinare i registri WAL e di archivio; quindi PostgreSQL decide il punto più coerente e lo recupera automaticamente.

I gruppi di coerenza sono una funzionalità di ONTAP e sono consigliati quando ci sono più volumi montati su una singola istanza o su un database con tablespace multiple. Uno snapshot del gruppo di coerenza garantisce che tutti i volumi siano raggruppati e protetti. È possibile gestire in modo efficiente un gruppo di coerenza da ONTAP System Manager, clonandolo per creare una copia dell'istanza di un database a scopo di test o sviluppo.

Software per la data Protection

Il plug-in NetApp SnapCenter per i database PostgreSQL, combinato con le tecnologie Snapshot e NetApp FlexClone, offre diversi vantaggi, tra cui:

- Backup e ripristino rapidi.
- Cloni efficienti in termini di spazio.
- La capacità di creare un sistema di disaster recovery rapido ed efficace.

> Potresti preferire scegliere i partner di backup premium di NetApp come Veeam Software e CommVault nelle seguenti circostanze:
>
> - Gestire i carichi di lavoro in un ambiente eterogeneo
> - Memorizzazione dei backup su cloud o nastro per una conservazione a lungo termine
> - Supporto per un'ampia gamma di versioni e tipi di sistema operativo

Il plug-in SnapCenter per PostgreSQL è un plugin supportato dalla comunità e la configurazione e la documentazione sono disponibili nell'archivio automazione di NetApp. Tramite SnapCenter, l'utente può eseguire il backup di database, clonare e ripristinare i dati in remoto.