PostgreSQL

Enterprise applications

NetApp
October 14, 2024

目录

PostgreSQL

基于ONTAP的PostgreSQL数据库

PostgreSQL附带的变体包括PostgreSQL、PostgreSQL Plus和EDB Postgrers Advanced Server (ePAS)。PostgreSQL通常部署为多层应用程序的后端数据库。它受常见中间件包(如PHP、Java、Python、Tcl/Tk、ODBC、 和JDBC)、并且一直以来都是开源数据库管理系统的常用选择。对于运行PostgreSQL数据库来说、ONTAP的可靠性、高性能和高效的数据管理功能是一个绝佳的选择。

有关ONTAP和PostgreSQL数据库的本文档将取代先前发布的_TR-4770：《基于ONTAP的PostgreSQL数据库最佳实践》。_

随着数据呈指数级增长、企业的数据管理变得更加复杂。这种复杂性会增加许可、运营、支持和维护成本。要降低总体TCO、请考虑使用可靠的高性能后端存储从商用数据库切换到开源数据库。

ONTAP是理想的平台、因为ONTAP专为数据库而设计。随机IO延迟优化、高级服务质量(Quality of Service、QoS)和基本FlexClone功能等众多功能专为满足数据库工作负载的需求而创建。

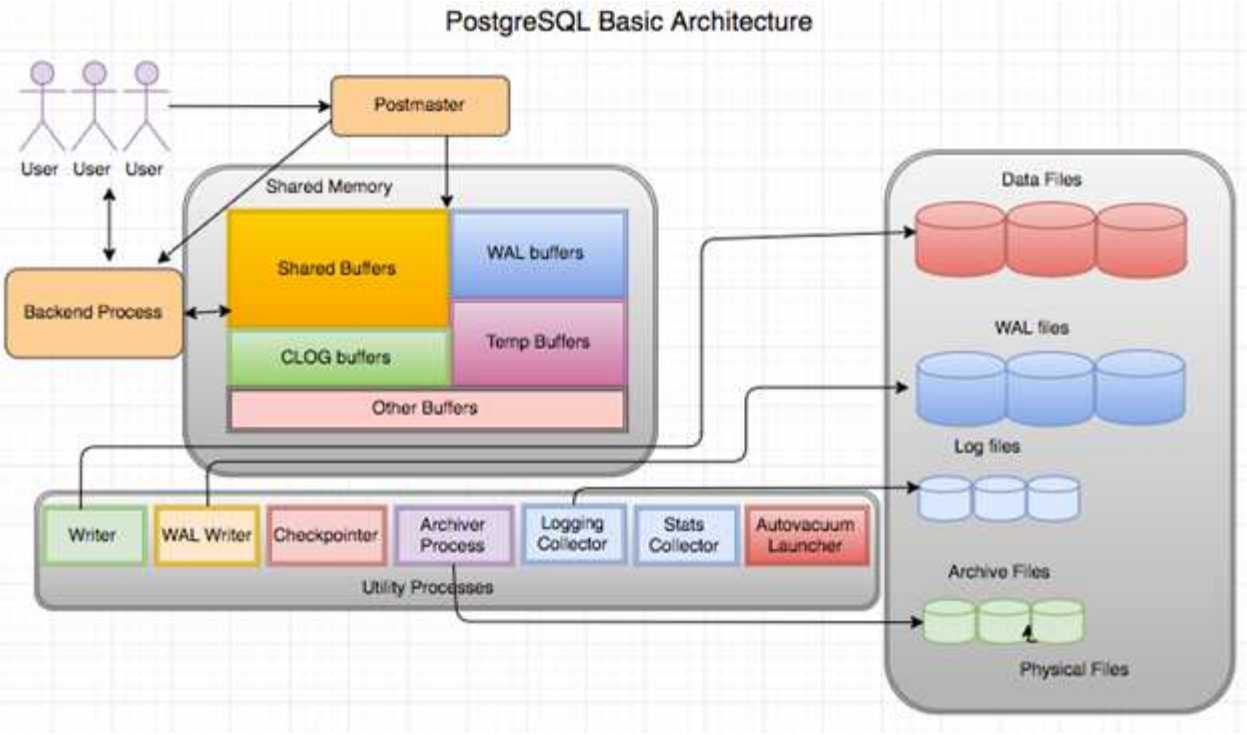
无中断升级(包括更换存储)等其他功能可确保关键数据库始终可用。您还可以通过MetroCluster为大型环境实现即时灾难恢复、或者使用SnapMirror主动同步功能选择数据库。

最重要的是、ONTAP提供无与伦比的性能、能够根据您的独特需求调整解决方案的大小。我们的高端系统可以提供超过100万次IOPS、延迟以微秒为单位。但是、如果您只需要10万次IOPS、则可以使用仍运行完全相同存储操作系统的小型控制器来调整存储解决方案的大小。

数据库配置

PostgreSQL架构

PostgreSQL是基于客户端和服务器架构的RDBMS。PostgreSQL实例称为数据库集群、它是一组数据库、而不是一组服务器。



PostgreSQL数据库中有三个主要元素：PostMAIN、前端(客户端)和后端客户端向邮件服务器发送请求、并提供IP协议和要连接到的数据库等信息。邮件服务器对连接进行身份验证，并将其传递到后端进程以进行进一步的通信。后端进程执行查询并将结果直接发送到前端(客户端)。

PostgreSQL实例基于多进程模型、而不是多线程模型。它会为不同的作业生成多个进程、每个进程都有自己的功能。主要进程包括客户端进程、WAL写入程序进程、后台写入程序进程和检查指针进程：

- 当客户端(前台)进程向PostgreSQL实例发送读取或写入请求时、它不会直接向磁盘读取或写入数据。它首先将数据缓冲在共享缓冲区和预写日志记录(Write-Ahead Logging、WAL)缓冲区中。
- WAL写入程序进程操纵共享缓冲区和WAL缓冲区的内容以写入WAL日志。WAL日志通常是PostgreSQL的事务日志、并按顺序写入。因此、为了缩短数据库的响应时间、PostgreSQL首先写入事务日志并确认客户端。
- 为了使数据库处于一致状态、后台写入程序进程会定期检查共享缓冲区中是否存在脏页。然后、它会将数据转储到存储在NetApp卷或LUN上的数据文件中。
- 检查指针进程也会定期运行(比后台进程频率低)、并防止对缓冲区进行任何修改。它会向WAL写入程序进程发出信号、将检查点记录写入并刷新到NetApp磁盘上存储的WAL日志的末尾。它还会向后台写入程序进程发出信号、指示将所有脏页写入磁盘并将其刷新。

PostgreSQL初始化参数

您可以使用创建新的数据库集群 `initdb` 计划。一个 `initdb` 脚本将创建用于定义集群的数据文件、系统表和模板数据库(template0和template1)。

模板数据库表示一个常用数据库。它包含系统表、标准视图、函数和数据类型的定义。 `pgdata` 用作的参数 `initdb` 指定数据库集群位置的脚本。

PostgreSQL中的所有数据库对象都由各自的OID在内部管理。表和索引也由各个OID管理。数据库对象及其各自的OID之间的关系存储在相应的系统目录表中、具体取决于对象类型。例如、数据库和堆表的OID存储在中 `pg_database` 和`pg_class。您可以通过在PostgreSQL客户端上发出查询来确定这些OID。

每个数据库都有自己的表和索引文件、这些表和索引文件的大小限制为1 GB。每个表都有两个关联文件、后缀分别为 `_fsm` 和 `_vm`。它们称为可用空间映射和可见性映射。这些文件用于存储有关可用空间容量的信息、并可查看表文件中的每个页面。索引只具有单个可用空间映射、而不具有可见性映射。

。 `pg_xlog/pg_wal` 目录包含预写日志。预写日志用于提高数据库可靠性和性能。每当您更新表中的某一行时、PostgreSQL都会首先将更改写入预写日志、然后将修改写入实际数据页面到磁盘。。 `pg_xlog` 目录通常包含多个文件、但initdb仅创建第一个文件。根据需要添加其他文件。每个xlog文件的长度为16 MB。

使用**ONTAP**的**PostgreSQL**数据库配置

可以通过多种PostgreSQL调整配置来提高性能。

最常用的参数如下：

- `max_connections = <num>`：一次可建立的最大数据库连接数。使用此参数可限制磁盘交换并导致性能下降。根据您的应用程序要求、您还可以针对连接池设置调整此参数。
- `shared_buffers = <num>`：提高数据库服务器性能的最简单方法。对于大多数现代硬件、默认值为low。在部署期间、它会设置为系统上可用RAM的大约25%。此参数设置因其在特定数据库实例中的工作方式而异；您可能需要按试用和错误增加和减小值。但是、将其设置为高可能会降低性能。
- `effective_cache_size = <num>`:该值告诉PostgreSQL的优化器PostgreSQL有多少内存可用于缓存数据，并帮助确定是否使用索引。值越大、使用索引的可能性就越大。此参数应设置为分配给的内存量 `shared_buffers` 加上可用的操作系统缓存容量。此值通常超过系统总内存的50%。
- `work_mem = <num>`:该参数控制在排序操作和哈希表中使用的内存量。如果您在应用程序中进行大量排序，则可能需要增加内存量，但要小心。它不是系统范围的参数、而是每个操作的参数。如果复杂查询包含多个排序操作、则它会使用多个work _mEM单元内存、多个后端可以同时执行此操作。如果值过大、此查询通常会导致数据库服务器进行交换。此选项以前在早期版本的PostgreSQL中称为Sort_mm。
- `fsync = <boolean> (on or off)`:此参数用于确定在提交事务之前是否应使用fsync()将所有WAL页面同步到磁盘。关闭它有时可以提高写入性能、而打开它可以增强保护、防止在系统崩溃时发生损坏。
- `checkpoint_timeout`：检查点进程将已提交的数据转至磁盘。这涉及到磁盘上的大量读/写操作。该值以秒为单位进行设置、较低的值可缩短崩溃恢复时间、增加该值可通过减少检查点调用来减少系统资源的负载。根据应用程序的严重程度、使用情况和数据库可用性、设置checkpoint_timeout的值。
- `commit_delay = <num>` 和 `commit_siblings = <num>`：这些选项结合使用，可以同时写出多个提交的事务，从而帮助提高性能。如果在提交事务时有多个commit兄弟姐妹对象处于活动状态、则服务器将等待commit delay微秒尝试一次提交多个事务。
- `max_worker_processes / max_parallel_workers`：为流程配置最佳数量的员工。max_paral_workers对应于可用的CPU数量。根据应用程序设计、查询可能需要较少的工作人员来执行并行操作。最好保持两个参数的值相同、但在测试后调整该值。
- `random_page_cost = <num>`：该值控制PostgreSQL查看非顺序磁盘读取的方式。值越高、意味着PostgreSQL更有可能使用顺序扫描而不是索引扫描、这表示您的服务器具有快速磁盘在评估基于计划的优化、清理、索引以更改查询或架构等其他选项后修改此设置。
- `effective_io_concurrency = <num>`：此参数用于设置PostgreSQL尝试同时执行的并发磁盘I/O操作的数量。提高此值会增加任何单个PostgreSQL会话尝试并行启动的I/O操作的数量。允许的范围为1到1、000或零、用于禁止发出异步I/O请求。目前、此设置仅影响位图堆扫描。固态驱动器(SSD)和其他基于内存的存储(NVMe)通常可以处理多个并发请求、因此、数百个请求的最大价值可能是。

有关PostgreSQL配置参数的完整列表、请参见PostgreSQL文档。

吐司

TOAST表示超大属性存储技术。PostgreSQL使用固定的页面大小(通常为8 KB)、并且不允许元组跨越多个页面。因此、无法直接存储大字段值。当您尝试存储超过此大小的行时、TOAST会将大型列的数据拆分成较小的"部分"、并将其存储在TOAST表中。

只有在将结果集发送到客户端时、才会提取已分配属性的大值(如果已选中)。表本身要小得多、并且可以在共享缓冲区缓存中容纳更多行、而不是在没有任何线外存储(TOAST)的情况下。

真空

在正常的PostgreSQL操作中，被更新删除或废弃的元组不会从其表中物理删除；它们会一直存在，直到运行真空为止。因此、您必须定期运行真空、尤其是在频繁更新的表上。然后、必须回收占用的空间、供新行重复使用、以避免磁盘空间中断。但是、它不会将空间归还给操作系统。

页面中的可用空间不会碎片化。真空会重新写入整个数据块、从而高效地填充其余行、并在一页中保留一个连续的可用空间块。

相比之下、真空全满会通过写入不含死空间的全新表文件来主动压缩表。此操作可最大程度地减小表大小、但可能需要很长时间。此外、还需要额外的磁盘空间来创建新的表副本、直到操作完成为止。常规真空的目标是避免真空完全发挥作用。此过程不仅可以将表保持在其最小大小、还可以保持磁盘空间的稳定使用状态。

PostgreSQL表空间

初始化数据库集群时、系统会自动创建两个表空间。

。 `pg_global` 表空间用于共享系统目录。。 `pg_default` 表空间是template1和template0数据库的默认表空间。如果用于初始化集群的分区或卷用尽空间且无法扩展、则可以在其他分区上创建并使用表空间、直到可以重新配置系统为止。

使用率较高的索引可以放置在快速、高可用性的磁盘上、就像固态设备一样。此外、存储很少使用或不影响性能的归档数据的表可以存储在成本较低、速度较慢的磁盘系统上、例如SAS或SATA驱动器。

表空间是数据库集群的一部分、不能视为数据文件的自主集合。它们依赖于主数据目录中包含的元数据、因此无法连接到其他数据库集群或单独备份。同样、如果丢失表空间(由于文件删除、磁盘故障等原因)、数据库集群可能会变得不可读或无法启动。将表空间放置在临时文件系统(如RAM磁盘)上会危及整个集群的可靠性。

创建表空间后、如果请求用户具有足够的权限、则可以使用任何数据库中的表空间。PostgreSQL使用符号链接来简化表空间的实现。PostgreSQL在中添加一行 `pg_tablespace` 表(一个全组范围的表)、并为该行分配一个新的对象标识符(OID)。最后、服务器使用OID在集群和给定目录之间创建符号链接。目录 `$PGDATA/pg_tblspc` 包含指向集群中定义的每个非内置表空间的符号链接。

存储配置

使用NFS文件系统的PostgreSQL数据库

PostgreSQL数据库可以托管在NFSv3或NFSv4文件系统上。最佳选择取决于数据库以外的因素。

例如、在某些集群环境中、NFSv4锁定行为可能更好。(请参见 "此处" 了解更多详细信息)

否则、数据库功能(包括性能)应接近相同。唯一的要求是使用 `hard` 挂载选项。要确保软超时不会产生不可恢复的IO错误、必须执行此操作。

如果选择NFSv4作为协议、NetApp建议使用NFSv4.1。在NFSv4.1中、NFSv4协议有一些功能增强功能、可提高NFSv4.0的故障恢复能力。

对常规数据库工作负载使用以下挂载选项：

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

如果预期顺序IO较多、则可以按下一节所述增加NFS传输大小。

NFS传输大小

默认情况下、ONTAP会将NFS I/O大小限制为64K。

大多数应用程序和数据库的随机I/O使用的块大小要小得多、远远低于64K的最大值。大型块I/O通常会并行处理、因此最大64K也不会限制获得最大带宽。

在某些工作负载中、最大64K会产生限制。特别是、如果数据库执行的I/O数量较少但规模较大、则单线程操作(例如备份或恢复操作或数据库完整表扫描)运行速度会更快、效率也会更高。ONTAP的最佳I/O处理大小为256K。

给定ONTAP SVM的最大传输大小可按如下方式进行更改：

```
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::*>
```

小心
请勿将ONTAP上允许的最大传输大小减小到低于当前挂载的NFS文件系统的rsize/wsize值。在某些操作系统中、这可能会导致挂起甚至数据损坏。例如、如果NFS客户端当前设置为rsize/wsize 65536,则ONTAP最大传输大小可以在65536- 1048576之间进行调整,但不会产生任何影响,因为客户端本身是有限的。将最大传输大小减小至65536,可能会损坏可用性或数据。

在ONTAP级别增加传输大小后、将使用以下挂载选项：

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

NFSv3 TCP插槽表

如果在Linux中使用NFSv3、则正确设置TCP插槽表至关重要。

TCP插槽表相当于主机总线适配器(Host Bus Adapter、HBA)队列深度的NFSv3。这些表可控制任何时候都可以处理的NFS操作的数量。默认值通常为16、该值太低、无法实现最佳性能。在较新的Linux内核上会出现相反的问题、这会自动将TCP插槽表限制增加到使NFS服务器充满请求的级别。

为了获得最佳性能并防止出现性能问题、请调整控制TCP插槽表的内核参数。

运行 `sysctl -a | grep tcp.*.slot_table` 命令、并观察以下参数：

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

所有Linux系统都应包括 `sunrpc.tcp_slot_table_entries`，但只有部分包括 `sunrpc.tcp_max_slot_table_entries`。它们都应设置为128。

小心
如果未设置这些参数、可能会对性能产生显著影响。在某些情况下、性能会受到限制、因为Linux操作系统发出的I/O不足在其他情况下、随着Linux操作系统尝试问题描述的I/O数超过可处理的I/O数、I/O时间会增加。

使用**SAN**文件系统的**PostgreSQL**

采用SAN的PostgreSQL数据库通常托管在xfs文件系统上、但如果操作系统供应商支持、则可以使用其他数据库

虽然一个LUN通常可支持高达10万次IOPS、但IO密集型数据库通常需要使用带区化LVM。

LVM拼花

在闪存驱动器时代之前、条带化用于帮助克服旋转驱动器的性能限制。例如、如果操作系统需要执行1 MB的读取操作、则从单个驱动器读取1 MB的数据将需要大量的驱动器磁头查找和读取、因为1 MB的传输速度较慢。如果在8个LUN上对1 MB的数据进行条带化、则操作系统可以问题描述并行执行8个128 K读取操作、从而减少完成1 MB传输所需的时间。

使用旋转驱动器进行条带化更为困难、因为必须事先知道I/O模式。如果条带化未正确调整为真正的I/O模式、则条带化配置可能会损害性能。使用Oracle数据库、尤其是使用全闪存配置时、条带化更易于配置、并且经验证可显著提高性能。

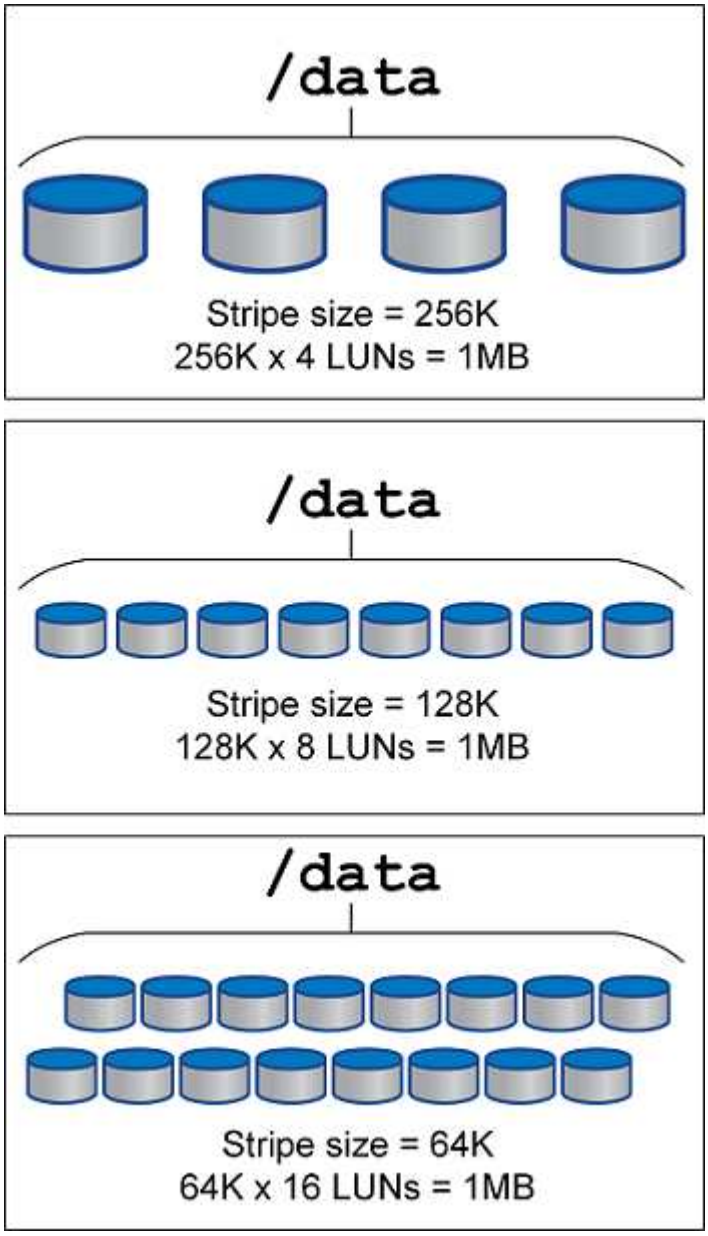
默认情况下、逻辑卷管理器(例如Oracle ASM)会进行条带化、但本机操作系统LVM则不会进行条带化。其中一些会将多个LUN绑定在一起、形成一个串联设备、从而导致数据文件只存在于一个LUN设备上。这会导致热点。其他LVM实施默认使用分布式块区。这与条带化类似、但更粗。卷组中的LUN会被划分为多个大块、称为块区、通常以MB为单位进行测量、然后逻辑卷会分布在这些块区中。结果是、文件的随机I/O应在各个LUN之间分布良好、但顺序I/O操作的效率不如所能达到的高。

性能密集型应用程序I/O几乎始终为(a)基本块大小单位或(b) 1兆字节。

条带化配置的主要目标是确保单文件I/O可作为一个单元执行、多块I/O (大小应为1 MB)可在条带化卷中的所有LUN之间均匀并行。这意味着条带大小不能小于数据库块大小、条带大小乘以LUN数量应为1 MB。

下图显示了三个可能的条带大小和宽度调整选项。选择LUN数量是为了满足上述性能要求、但在所有情况下、单

个条带内的总数据均为1 MB。



数据保护

PostgreSQL数据保护

存储设计的一个主要方面是为PostgreSQL卷启用保护。客户可以使用转储方法或文件系统备份来保护其PostgreSQL数据库。本节介绍备份单个数据库或整个集群的不同方法。

备份PostgreSQL数据有三种方法：

- SQL Server转储
- 文件系统级备份
- 持续归档

SQL Server转储方法的理念是、使用SQL Server命令生成一个文件、当返回到服务器时、该文件可以像转储时那样重新创建数据库。PostgreSQL提供实用程序 `pg_dump` 和 `pg_dump_all` 用于创建单个和集群级别的备份。这些转储是逻辑转储、不包含可供WAL重放使用的足够信息。

另一种备份策略是使用文件系统级备份、即管理员直接复制PostgreSQL用于将数据存储在数据库中的文件。此方法在脱机模式下完成：必须关闭数据库或集群。另一种替代方法是使用 `pg_basebackup` 对PostgreSQL数据库运行热流备份。

PostgreSQL数据库和存储快照

使用PostgreSQL进行基于Snapshot的备份时、需要为数据文件、WAL文件和归档的WAL文件配置快照、以提供完整或时间点恢复。

对于PostgreSQL数据库、快照的平均备份时间介于几秒到几分钟之间。此备份速度比快60到100倍 `pg_basebackup` 以及其他基于文件系统的备份方法。

NetApp存储上的快照可以是崩溃状态一致的快照、也可以是应用程序一致的快照。系统会在存储上创建崩溃状态一致的快照、而不会使数据库处于静音状态、而是在数据库处于备份模式时创建应用程序一致的快照。NetApp还可确保后续快照是永久增量备份、以节省存储空间并提高网络效率。

由于快照速度快且不会影响系统性能、因此您可以计划每天创建多个快照、而不是像使用其他流式备份技术那样创建一个每日备份。在需要执行还原和恢复操作时、系统停机时间可通过以下两个主要功能来减少：

- NetApp SnapRestore数据恢复技术意味着只需几秒钟即可执行还原操作。
- 主动恢复点目标(Recovery Point目标、RPO)意味着、必须应用的数据库日志减少、并且前向恢复也会加快。

要备份PostgreSQL、必须确保数据卷同时受到(一致性组) WAL和归档日志的保护。在使用Snapshot技术复制WAL文件时、请确保运行 `pg_stop` 刷新必须归档的所有WAL条目。如果在还原期间刷新WAL条目、则只需停止数据库、卸载或删除现有数据目录、并对存储执行SnapRestore操作即可。还原完成后、您可以挂载系统并将其恢复到当前状态。对于时间点恢复、您还可以恢复WAL和归档日志；然后PostgreSQL确定最一致的点并自动恢复它。

一致性组是ONTAP中的一项功能、如果在一个实例或包含多个表空间的数据库中挂载了多个卷、则建议使用一致性组。一致性组快照可确保所有卷都分组在一起并受到保护。您可以从ONTAP系统管理器高效管理一致性组、甚至可以克隆一致性组以创建数据库的实例副本、用于测试或开发目的。

PostgreSQL数据保护软件

适用于PostgreSQL数据库的NetApp SnapCenter插件与Snapshot和NetApp FlexClone技术相结合、可为您提供以下优势：

- 快速备份和恢复。
- 节省空间的克隆。
- 构建快速有效的灾难恢复系统的能力。

在以下情况下、您可能更愿意选择NetApp的高级备份合作伙伴、例如Veeam Software和Commvell：

- 管理异构环境中的工作负载
- 将备份存储到云或磁带以供长期保留
- 支持多种操作系统版本和类型

适用于PostgreSQL的SnapCenter插件是社区支持的插件、安装和文档可从NetApp自动化商店获得。通过SnapCenter、用户可以远程备份数据库、克隆和还原数据。