テクニカル レポート

ONTAP上のPostgreSQLデータベース

ベストプラクティス

NetApp
Anup Bharti / Ebin Kadavy / Manohar Kulkarni / Jeffrey Steiner
2023年4月| TR-4770

概要

PostgreSQLには、PostgreSQL、PostgreSQL Plus、EDB Postgres Advanced Server（EPAS）などのバリアントが付属しています。PostgreSQLは通常、多層アプリケーションのバックエンドデータベースとして導入されます。一般的なミドルウェアパッケージ(PHP、Java、Python、Tcl/Tk、ODBCなど)でサポートされています。とJDBC）は、オープンソースのデータベース管理システムでは、歴史的に人気のある選択肢でした。本ドキュメントでは、NetApp® ONTAP® データ管理ソフトウェアにPostgreSQLを導入するためのチューニングおよびストレージ構成の要件とガイダンスについて説明します。

目次

図一覧

はじめに

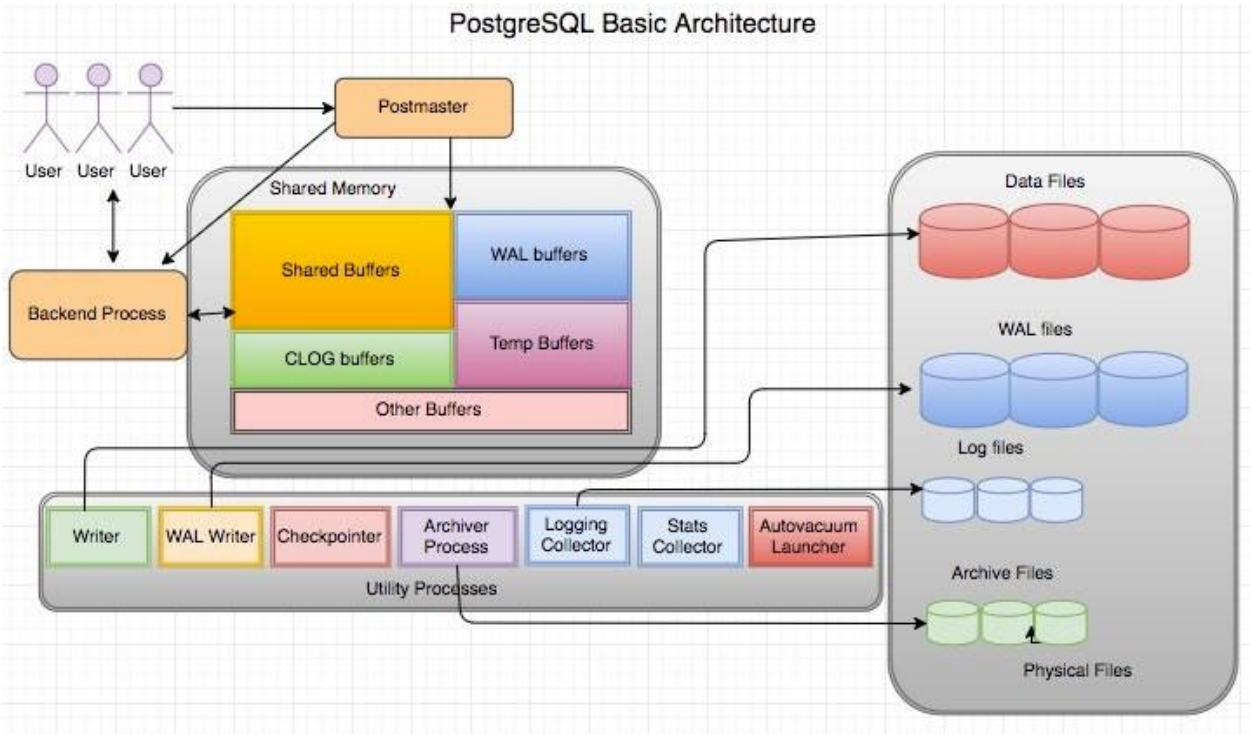
データが急激に増加するにつれて、企業のデータ管理はますます複雑になります。この複雑さにより、ライセンス、運用、サポート、メンテナンスのコストが増大します。全体的なTCOを削減するには、信頼性とパフォーマンスに優れたバックエンドストレージを使用して、商用データベースからオープンソースデータベースに切り替えることを検討してください。PostgreSQLは、学術、商業、大企業で広く使用されている高度なオープンソースデータベースです。他のリレーショナルデータベース管理システム(RDBMS)のギャップに対処する新しい機能セットが付属しています。NetApp FASやNetApp AFF、NetApp ONTAP Select、Amazon Web Services（AWS）FSx on ONTAP、NetApp Cloud Volumes ONTAPなどのストレージディスクアレイでは、ONTAP独自のソフトウェアが使用されています。ONTAP 9のリリースにより、フラッシュ、ディスク、クラウドにわたってデータ管理を統合できます。この統合により、ストレージ環境が簡易化され、データをどこにでも簡単に移動できるデータファブリックの基盤が構築されます。

注: このドキュメントは、物理環境と仮想環境の両方でPostgreSQLデータベースを使用するお客様を対象としています。ONTAP 9.xでPostgreSQLデータベースを正常に展開および管理するためのベストプラクティスについて説明します。このレポートの推奨事項は一般的なものであるため、どの構成にも固有のものではありません。ビジネスニーズによっては、一部の提案で変更が必要になる場合があります。使用する環境は、PostgreSQL、ハイパーバイザー、OS、ONTAPストレージに関する公式ドキュメントに照らして評価する必要があります。

PostgreSQLアーキテクチャ

PostgreSQLは、クライアントとサーバのアーキテクチャに基づいたRDBMSです。PostgreSQLインスタンスは データベースクラスタと呼ばれ、サーバの集合ではなくデータベースの集合です。図1 に、PostgreSQLアーキテクチャの概要を示します。

図1）PostgreSQLの基本アーキテクチャ



PostgreSQLデータベースには、postmaster、フロントエンド(クライアント)、バックエンドの3つの要素があります。クライアントは、IPプロトコルや接続先データベースなどの情報を含む要求をポストマスターに送信します。postmasterは接続を認証し、さらに通信するためにバックエンドプロセスに渡します。バックエンドプロセスはクエリを実行し、結果を直接フロントエンド（クライアント）に送信します。

PostgreSQLインスタンスは、マルチスレッドモデルではなく、マルチプロセスモデルに基づいています。ジョブごとに複数のプロセスが生成され、各プロセスには独自の機能があります。主なプロセスには、クライアントプロセス、WALライタプロセス、バックグラウンドライタプロセス、およびcheckpointerプロセスが含まれます。

- クライアント（フォアグラウンド）プロセスがPostgreSQLインスタンスに読み取りまたは書き込み要求を送信しても、データを直接ディスクに読み取りまたは書き込みすることはありません。最初に、共有バッファとWAL(Write-Ahead Logging)バッファにデータをバッファします。
- WALライタプロセスは、共有バッファとWALバッファの内容を操作してWALログに書き込みます。WALログは通常PostgreSQLのトランザクションログであり、シーケンシャルに書き込まれます。したがって、データベースからの応答時間を短縮するために、PostgreSQLはまずトランザクションログに書き込み、クライアントに確認応答します。
- データベースを整合性のある状態にするために、バックグラウンドライタープロセスは共有バッファにダーティページがないか定期的にチェックします。次に、NetAppボリュームまたはLUNに格納されているデータファイルにデータをフラッシュします。
- checkpointerプロセスも定期的に（バックグラウンドプロセスよりも少ない頻度で）実行され、バッファへの変更を防ぎます。WALライタプロセスに、NetAppディスクに保存されているWALログの末尾にチェックポイントレコードを書き込み、フラッシュするように指示します。また、すべてのダーティページをディスクに書き込み、フラッシュするようにバックグラウンドライタープロセスに通知します。

PostgreSQLのストレージ設計に関する考慮事項

このセクションでは、Storage Virtual Machine（SVM）、論理インターフェイス（LIF）の設計、ボリュームレイアウト、Storage Efficiency、PostgreSQLデータベースを導入するためのデータ保護など、NetApp ONTAPストレージ設計の主要なトピックについて説明します。ONTAPの導入と設定の詳細については、『ONTAPアドミニストレーションガイド』を参照してください。

Storage Virtual Machine

SVMには、複数のデータボリュームと、クライアントへのデータの提供に使用するLIFが1つ以上含まれます。データをやり取りするために、クラスタでは少なくとも1つのSVMが必要です。SVMは共有されている仮想データ ストレージおよびネットワークを安全に分離し、クライアントにはそれぞれのSVMが単一の専用サーバとして認識されます。

マルチテナントのPostgreSQL環境では、各テナントのデータ（データベースインスタンス）にONTAPストレージ上の専用SVMを割り当てることができます。NetAppでは、クラスタノードあたりのONTAP SVM数の上限を約125個にすることを推奨していますが、通常はこの上限に達する前にLIF数の上限に達します。

また、同じSVMに専用ボリュームを配置し、ネットワークセグメント（LIF、ポート）ごとに分離することで、ある程度マルチテナンシーを実現することもできます。

論理インターフェイス

PostgreSQLデータベース用のLIFを作成する場合は、次の点を考慮してください。

- **パフォーマンス**：ネットワーク帯域幅は十分か。
- **耐障害性**：設計に単一点障害（Single Point of Failure）はありますか。
- **管理性**：ネットワークを無停止で拡張できますか。

NFS LIFの設計

NFSプロトコルではデータへの複数のパスを定義する機能に制限がありますが、データベースがNFSプロトコルよりも優れたパフォーマンスと耐障害性を実現するのに役立つガイドラインがいくつかあります。

物理ポートにLIFをバインドすると、ネットワーク構成をきめ細かく制御できるようになります。これは、ONTAPシステム上の特定のIPアドレスは、一度に1つのネットワークポートにしか関連付けられないためです。フェイルオーバー グループとフェイルオーバー ポリシーを設定すれば、耐障害性も実現できます。大規模な環境で実行されるNFSデータベースでは、Link Aggregation Control Protocol（LACP）とIPロードバランシングを有効にする方法が最も一般的です。ただし、NFSクライアントの数が少ない場合は、ONTAPインターフェイスグループでLACPおよびIPロードバランシングを有効にするのは効率的ではないようです。

LIFフェイルオーバー

ネットワーク停止時のLIFの動作を制御するのが、フェイルオーバー ポリシーとフェイルオーバー グループです。ONTAPでは、ブロードキャストドメインに基づいてLIFのフェイルオーバーを管理できます。そのため、特定のサブネットにアクセスできるポートをすべて定義し、ONTAPが適切なフェイルオーバーLIFを選択できるようにすることができます。高速データベースストレージネットワーク環境では予測性がないため、このアプローチには限界があります。

たとえば、ファイルシステムへの日常的なアクセスに使用する1Gbポートと、データファイルI/Oに使用する10Gbポートの両方を環境に含めることができます。両方のタイプのポートが同じブロードキャストドメインにあると、LIFのフェイルオーバーによってデータファイルI/Oが10Gbポートから1Gbポートに移動することがあります。そのため、NetAppではLIFのフェイルオーバーに個々のポートを選択することを推奨します。

LIFフェイルオーバーに関する主な推奨事項

- ユーザ定義のフェイルオーバーグループを設定します。
- フェイルオーバーグループに、ストレージフェイルオーバー（SFO）パートナーコントローラのポートを入力します。LIFはストレージフェイルオーバー時にアグリゲートに従って作成されるため、間接トラフィックの生成を回避できます。
- パフォーマンス特性が元のLIFと同じフェイルオーバーポートを使用してください。例えば、10Gbの1つの物理ポート上のLIFには、10Gbポート1つだけで構成されたフェイルオーバー グループを含め、4ポートLACP LIFは、別の4ポートLACP LIFにフェイルオーバーする必要があります。これらのポートが、ブロードキャスト ドメインに定義されたポートのサブセットになります。

フェイルオーバーポリシーとフェイルオーバーグループの詳細については、『ONTAPネットワーク管理ガイド』を参照してください。

SAN LIFの設計

最新のSAN実装では、クライアントはマルチパスを使用して複数のネットワークパス経由でデータにアクセスし、アクセスに最適なパスを選択できます。そのため、SANクライアントは最適なパス間でI/Oの負荷を自動的に分散するため、LIFの設計は簡単です。あるパスが使用不可能になると、クライアントによって別のパスが自動で選択されます。その結果、設計がシンプルになるため、SAN LIFの管理性が向上します。

SAN環境では、ボリュームを再配置する際にLIFを移行する必要はありません。ボリュームの移動が完了すると、ONTAPはパスの変更をSANに通知し、SANクライアントは自動的に再最適化します。

主な推奨事項

- 必要以上の数のパスを作成しないでください。パスが多すぎると管理全体が複雑になり、一部のホストでパスのフェイルオーバーに原因の問題が発生する可能性があります。
- マルチパス環境では、高可用性を確保するために、異なるスイッチからゾーニングされたホスト用のNetApp LUNを用意することを推奨します。

- Data ONTAP 8.3以降のONTAPバージョンでは、選択的なLUNマッピングデータがデフォルトです。SLMを使用すると、新しいLUNはすべて、基盤となるアグリゲートを所有するノードとノードのハイアベイラビリティ（HA）パートナーから自動的に通知されます。これにより、ポートのアクセス性を制限するためにポートセットを作成したりゾーニングを設定したりする必要がなくなります。
- I/Oが大量に発生するデータベース環境では間接トラフィックは避けた方がよいでしょう。レイテンシはマイクロ秒単位で重要ですが、一般的なワークロードでは、パフォーマンスへの明らかな影響はごくわずかです。
- FCゾーンに複数のイニシエータを含めることはできません。このような配置は最初は機能しているように見えるかもしれませんが、イニシエータ間のクロストークは最終的にパフォーマンスと安定性の妨げになります。
- LUN移動イベント（クラスタ内のHAペア間でのLUNの移動に伴うボリュームやLUNの移動など）には、LUNが提供されていることを確認する手順が含まれます。この手順は、移動イベントが開始される前に、デスティネーションストレージコントローラを使用して実行できます。
- 移動する前に、選択的LUNマップのレポートノードリストを変更してください。

ボリュームのレイアウト

ボリュームレイアウトが適切に設計されているため、データベースのパフォーマンスが向上し、データベースの管理が容易になります。データベースのボリュームレイアウトを定義するときは、データベースのサイズ、データベースの増加率、バックアップ頻度など、複数の要素を考慮する必要があります。

専用ストレージレイアウト

- ミッションクリティカルな環境では、NetAppでは、SANとNFSの両方のレイアウトでパフォーマンスを向上させるために、データファイルとWALログを専用のボリュームまたはLUNに保持することを推奨しています。
- pg_basebackのpg_dumpを使用してデータベースをバックアップする場合は、管理しやすいように個別のLUNを用意する必要があります。また、アーカイブログを格納する専用のLUNも必要です。
- データファイルとWALログボリュームを分離することで、データベースのリカバリを高速化することもできます。
- PostgreSQLデータベースの複数のテーブルスペースを専用ボリュームに分散して、個々のデータベースのパフォーマンスを確保することもできます。
- ONTAPクラスタ内の他のアプリケーションの処理でアグリゲートがビジー状態でないノードの専用アグリゲートにPostgreSQLデータベースを分離する必要があります（図2）。
- 大規模なデータベースの場合、図3に示すように、独立した表領域を作成することで、データファイルを複数のボリュームに分割できます。

図2）PostgreSQLデータベース専用のストレージレイアウト

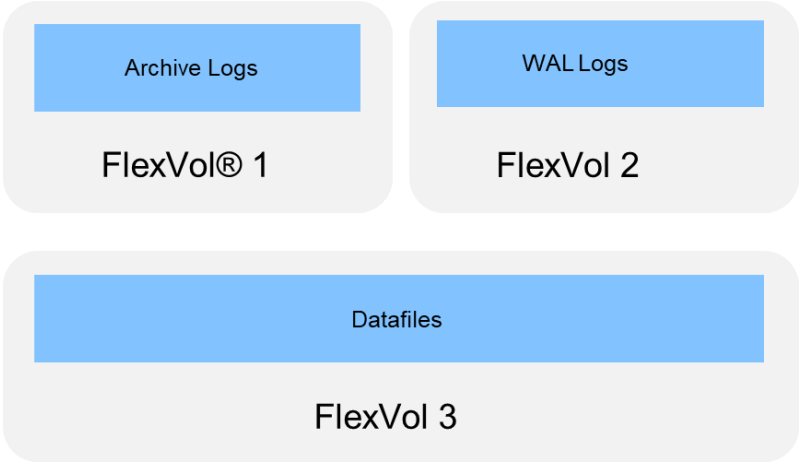
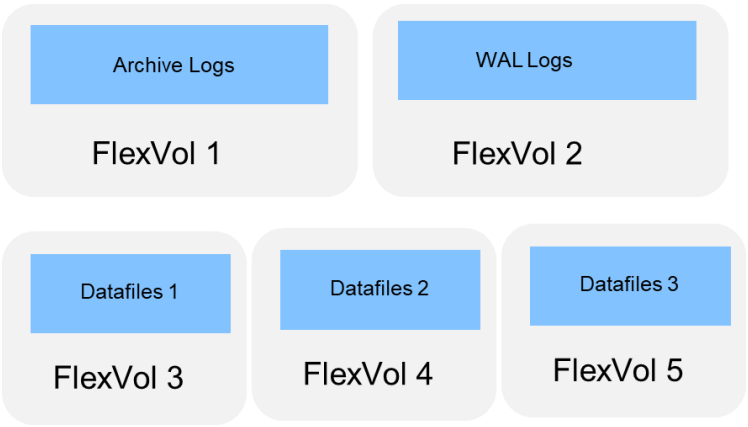


図3）複数の表領域を含むPostgreSQLデータベース専用のストレージレイアウト



LVMストライピング

論理ボリュームマネージャ（LVM）ストライピングとは、複数のLUNにデータを分散することです。フラッシュドライブが登場する以前は、回転式ドライブのパフォーマンス上の制限を克服するためにストライピングが使用されていました。たとえば、OSが1MBの読み取り操作を実行する必要がある場合、1つのドライブからその1MBのデータを読み取るには、1MBがゆっくり転送されるため、多くのドライブヘッドのシークと読み取りが必要になります。この1MBのデータが8つのLUNにストライピングされている場合、OSは8つの128K読み取り処理を並行して問題できるため、1MB転送の完了に必要な時間が短縮されます。

回転式ドライブを使用したストライピングは、I/Oパターンを事前に把握しておく必要があったため、より困難でした。ストライピングが実際のI/Oパターンに合わせて正しく調整されていない場合、ストライピングされた構成ではパフォーマンスが低下する可能性があります。オールフラッシュ構成では、ストライピングは設定がはるかに簡単で、パフォーマンスが劇的に向上することが実証されています。

ほとんどのLVM実装では、デフォルトで分散エクステントが使用されます。これはストライピングに似ていますが、粗いです。ボリュームグループ内のLUNはエクステントと呼ばれる大きな部分に分割され、通常は数メガバイト単位で測定されます。データベースで使用される論理ボリュームは、それらのエクステントに分散されます。その結果、データファイルに対するランダムI/Oが発生し、LUN間に適切に分散されますが、シーケンシャルI/O処理はそれほど効率的ではありません。

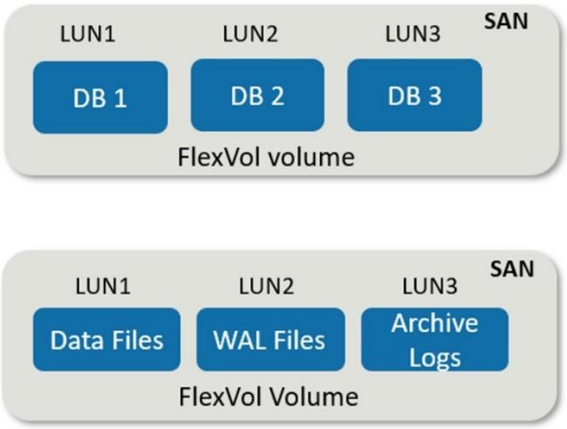
主な目的は、ストライピングされたボリューム内のすべてのLUNでI/Oを均等に並列化し、ストライプサイズをデータベースのブロックサイズより小さくしないことです。

共有ストレージレイアウト

- 重要度の低い環境では、同じボリュームまたはLUN上で複数のデータベースを共有することで、運用コストと管理コストを削減できます（図4）。
- 1つのLUNまたはボリュームに複数のデータベースのデータおよびWALログをホストすると、データセット全体に対して1つのSnapshotコピーが作成されるため、NetApp Snapshot™の統合プロセスが最適化されます。

注：このレイアウトは、ミッションクリティカルなシステムには適していない場合があります。

図4）共有ストレージレイアウト



仮想レイアウト

データベースの仮想化を検討する場合は、ビジネスニーズに基づいてストレージの意思決定を行う必要があります。iSCSI、FC、NFS、NVMe over Fabrics（NVMe-oF）などのいずれかのプロトコルを使用して、さまざまな方法でストレージを仮想マシン（VM）にプロビジョニングできます。

- ゲストVM上のVMのiSCSIイニシエータで管理されるiSCSI LUN
- ゲストVMに直接マウントされるNFSファイルシステム
- FC rawデバイスマッピング（RDM）
- ハイパーバイザー（VMware）によってNFSおよびVMware Virtual Machine File System（VMFS）データストアとして管理されるONTAPボリューム、LUN、またはNVMeネームスペース

図5）共有ストレージを使用した仮想化されたデータベースレイアウト

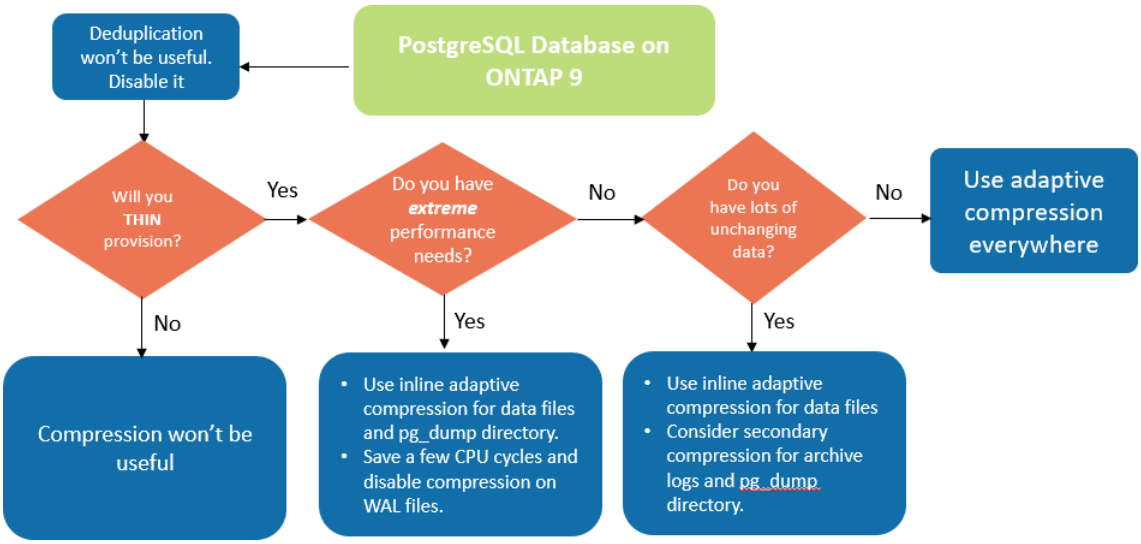
主な推奨事項

- VMが自身のファイルシステムを所有している場合は、データのファイルシステムのソースを簡単に特定できます。
- ハイパーバイザーデータストアを介してすべてのI/Oをチャネリングすると、パフォーマンスにプラスの影響が生じる可能性があります。
- データストアを選択すると、全体的な管理性や運用オーバーヘッドは向上しますが、タイムアウト、パラメータ、ログファイル、潜在的なバグなどが原因で複雑さが増します。
- VMware Virtual Machine File System（VMFS）データストアではなくNFSを選択すると、災害時にNetApp SnapRestore®テクノロジを使用する際に役立ちます。

ストレージ効率化

Storage Efficiencyを使用すると、最大限のデータを最小限のスペースに最小限のコストで格納できます。ただし、一部のテクノロジを有効にすると、データベースに影響する可能性があります。図6 のフローチャートは、ストレージ効率化に関する考慮事項を示しています。

図6）Storage Efficiencyのフローチャート



主な推奨事項

- ストレージの動的ディスクプール（アグリゲート）を共有するテナントが1つしかない場合は、ボリューム間でシンプロビジョニングを有効にすることを推奨します。一方、複数のテナントでストレージプール（アグリゲート）を共有しているマルチテナント環境では、テナントの使用パターンが不確実であるため、スペースが不足する可能性が高くなります。そのため、NetAppでは、マルチテナント環境でシックプロビジョニングを使用することを推奨します。
- ボリュームがシックプロビジョニングされている場合は `sis undo` 、コマンドを使用して、そのボリュームの効率化機能（圧縮や重複排除の削除など）をすべて無効にする必要があります。ボリュームは `volume efficiency show` 出力に表示されません。有効になっている場合、ボリュームは部分的に効率化機能用に設定されたままであり、スペースが不足してデータベースI/Oエラーが発生する可能性があります。
- 仮想化レイアウトでは、データストアにシンまたはシックの遅延初期化プロビジョニングを使用します。
- PostgreSQLのワークロードについてはNetApp、データボリュームとWALログボリュームの重複排除を無効にすることを推奨します。`pg_dump` または `tar` ファイルシステムバックアップボリュームに対して重複排除、圧縮、またはその両方を有効にすると、スペースを節約できますが、お客様の環境によって異なります。
- LUNアライメントとは、基盤となるファイルシステムのレイアウトに合わせてI/Oを最適化することです。NetAppシステムの場合、ストレージのまとまりは4KB単位なので、PostgreSQLデータファイルの8KBブロックを、4KBブロック2つに正確にアライメントします。LUNの設定エラーによってアライメントがいずれかの方向に1KBずれた場合、8KBのPostgreSQLブロックは、4 KBのストレージブロックが2つではなく3つに配置されます。この状態は、レイテンシの増大やストレージ システム内で実行されるI/Oの増大につながります。
- 可能な場合はNetApp FlexClone®テクノロジを使用して、レポート作成、クイックテストの実行、またはデータのサブセットのリストアを目的として、ごく短期間でデータベースのコピーを作成します。FlexCloneの方が効率的で、データベースコピー処理の開始時に追加のスペースが消費されることはありません。

データ保護

ストレージ設計の主な側面の1つは、PostgreSQLボリュームの保護を有効にすることです。お客様は、ダンプアプローチを使用するか、ファイルシステムバックアップを使用して、PostgreSQLデータベースを保護できます。このセクションでは、個 々 のデータベースまたはクラスタ全体をバックアップするさまざまな方法について説明します。

標準搭載のデータ保護

PostgreSQLデータをバックアップするには、次の3つの方法があります。

- SQL Serverダンプ
- ファイルシステムレベルのバックアップ
- 継続的アーカイブ

SQL Serverダンプ方式の背後にある考え方は、SQL Serverコマンドを使用してファイルを生成することです。このコマンドをサーバに返すと、ダンプ時と同じようにデータベースを再作成できます。PostgreSQLはユーティリティプログラムを提供し `pg_dump`、`pg_dump_all` 個別およびクラスタレベルのバックアップを作成します。これらのダンプは論理的であり、WAL再生で使用するのに十分な情報が含まれていません。

別のバックアップ戦略として、PostgreSQLがデータベースにデータを保存するために使用するファイルを管理者が直接コピーするファイルシステムレベルのバックアップを使用する方法があります。この方法はオフラインモードで実行されます。データベースまたはクラスタをシャットダウンする必要があります。もう1つの方法は、を使用して `pg_basebackup` PostgreSQLデータベースのホットストリーミングバックアップを実行する方法です。

Snapshotテクノロジ

NetAppは、企業があらゆるビジネスニーズに対応するバックアップとリカバリの計画を作成または調整するのに役立つ、テクノロジとツールの完全なポートフォリオを開発しました。ONTAPソフトウェアに搭載されたNetApp Snapshotテクノロジを使用すると、あらゆるサイズのデータセットについて、わずか数秒でバックアップを作成したりリストア処理を実行したりできます。PostgreSQLでは、データファイル、WALファイル、およびアーカイブされたWALファイルのスナップショットを有効にして、フルリカバリまたはポイントインタイムリカバリを提供する必要があります。

PostgreSQLデータベースの場合、Snapshotコピーを使用した平均バックアップ時間は数秒~数 分です。このバックアップ速度は `pg_basebackup`、他のファイルシステムベースのバックアップアプローチと比べて60～100倍高速です。

NetAppストレージ上のSnapshotコピーには、crash-consistentとアプリケーション整合性の両方を設定できます。データベースを休止せずにストレージ上にcrash-consistent Snapshotコピーが作成されます。一方、アプリケーションと整合性のあるSnapshotコピーは、データベースがバックアップモードの間に作成されます。NetAppでは、以降のSnapshotコピーが永久増分バックアップになるため、ストレージの節約とネットワーク効率が向上します。

Snapshotコピーは高速で、システムパフォーマンスに影響しないため、他のストリーミングバックアップテクノロジと同様に、1日1回のバックアップを作成するのではなく、1日に複数のSnapshotコピーをスケジュールできます。リストアとリカバリの処理が必要な場合は、次の2つの主な機能によってシステムのダウンタイムが短縮されます。

- NetApp SnapRestoreのデータリカバリテクノロジにより、リストア処理が数秒で実行されます。
- Recovery Point Objective（RPO；目標復旧時点）が頻繁に発生するため、適用するデータベースログの数が減り、フォワードリカバリも高速化されます。

PostgreSQLをバックアップするには、データボリュームが（コンシステンシグループ）WALとアーカイブログと同時にコピーされていることを確認する必要があります。Snapshotテクノロジを使用してWALファイルをコピーする場合は、を実行して、`pg_stop` アーカイブする必要があるすべてのWALエントリをフラッシュしてください。リストア中にWALエントリをフラッシュする場合は、データベースを停止するか、既存のデータディレクトリをアンマウントまたは削除し、ストレージでSnapRestore操作を実行するだけで済みます。

リストアが完了したら、システムをマウントして現在の状態に戻すことができます。ポイントインタイムリカバリの場合は、WALとアーカイブログをリストアすることもできます。次に、PostgreSQLは最も整合性のあるポイントを決定し、自動的にリカバリします。

整合グループはONTAPの機能であり、1つのインスタンスまたは複数の表領域を含むデータベースに複数のボリュームがマウントされている場合に推奨されます。整合性グループSnapshotを使用すると、すべてのボリュームがグループ化されて保護されます。整合グループはONTAP System Managerから効率的に管理できます。また、整合グループをクローニングして、テストや開発用にデータベースのインスタンスコピーを作成することもできます。

ストレージ レプリケーション

ストレージレプリケーションは、目標復旧時間（RTO）の要件が低~中程度の場合に適しています。プライマリストレージ上のSnapshotコピーは、セカンダリストレージまたはターシャリストレージ（NetAppのディザスタリカバリおよびバックアップのデスティネーション（NetApp SnapMirror® およびSnapVault® テクノロジ）にレプリケートできます。この方法を使用すると、アプリケーション透過的な迅速なフェイルオーバーと、バックアップとクローンの長期保存が可能になります。

SnapMirrorデータレプリケーションソフトウェアは、同期レプリケーションと非同期レプリケーションを提供します。レプリケーションはストレージ **ボリューム** レベルで構成されますが、NetApp MetroCluster™の 高可用性とディザスタリカバリは、ストレージ **システム** レベルで機能する同期レプリケーションを提供します。MetroClusterはSnapMirrorと統合されているため、アプリケーションの距離やSLAの要件に対応し、最大300kmの距離でデータ損失ゼロ（RPO）とフェイルオーバー時間ほぼゼロ（RTOは120秒未満）を実現します。

SnapMirrorは、従来のソリューションに比べてコスト効率に優れた機能を提供します。たとえば、数千ものPostgreSQLデータベースを保護して、他のワークロードと一緒に迅速なフェイルオーバーを実現したい場合は、SnapMirrorを使用すると、個 々 のアプリケーションフェイルオーバー手法よりも効率性とTCOが向上します。

NetAppのディザスタリカバリバックアップストレージは、オンプレミス、クラウド、コロケーションデータセンターのいずれでも実行できるため、企業はデータファブリック内の任意の場所にデータを簡単に移動でき、コピーインフラ間の透過性も高くなります。

NetApp SnapMirror® ビジネス継続性は、SANワークロード向けONTAPの機能の1つで、サイトやストレージのフェイルオーバー時に自動的にストレージを透過的にフェイルオーバーします。データベースのバックアップ時には、データベースデータファイルが複数のボリュームに分散されます。SnapMirrorビジネス継続性により、サイト間でデータが一貫してレプリケートされます。データベースが非常に重要で、RTOをほぼゼロにしたい場合は、NetAppでSnapMirrorビジネス継続性を検討することを推奨します。

データ保護ソフトウェア

PostgreSQLデータベース用NetApp SnapCenter®プラグインにSnapshotおよびNetApp FlexCloneテクノロジを組み合わせると、次のようなメリットがあります。

- 高速なバックアップとリストア：
- スペース効率に優れたクローン：
- 迅速で効果的なディザスタリカバリシステムを構築する能力。

注：以下の状況では、Veeam SoftwareやCommvaultなど、NetAppのプレミアムバックアップパートナーを選択することもできます。

  - 異機種混在環境全体でのワークロードの管理
  - バックアップをクラウドまたはテープに保存して長期保持
  - さまざまなバージョンと種類のOSをサポート

SnapCenter plugin for PostgreSQL is community supported plugin and the setup and documentation is available on NetAppオートメーションストア。SnapCenterを使用すると、データベースのバックアップ、データのクローニング、リストアをリモートで実行できます。

データベースノセキュリティ

ランサムウェアの脅威のケースが大幅に増加しているため、ユーザデータとビジネスデータの保護は組織にとって非常に重要です。管理者は適切な権限を設定しますが、ファイアウォールポリシーを設定して、データベースの選択的なアクセスをユーザに許可することができます。攻撃者はまだデータをハイジャックする方法を見つける可能性があります。ONTAPは、NASおよびSAN上のデータを保護するためのさまざまな機能を提供します。

改ざん防止Snapshotコピー：データベースを迅速にバックアップするためのSnapshotコピーの使用について、前述しました。攻撃者は、データが破損した場合にデータベースをリストアする最後のオプションであった可能性のあるSnapshotコピーを削除する可能性があります。改ざん防止Snapshotコピーを有効にすると、Snapshotコピーが誤ってまたは悪意を持って削除されないように、指定した期間ロックされます。改ざん防止Snapshotコピー機能はONTAP 9.12.1から利用でき、SnapLockライセンスとコンプライアンスクロックをオンにする必要があります。

ONTAPにはMulti-Admin Verification（MAV；マルチ管理者検証）機能があります。つまり、Snapshotコピーの削除やボリュームの変更を実行できる完全な権限を持つ管理者は1人もいません。これにより、セキュリティが強化され、不正な管理者によるボリュームやSnapshotコピーの改ざんを防ぐことができます。

PostgreSQLデータベースのベストプラクティス

このセクションでは、PostgreSQLデータベースを使用するためのより多くのベストプラクティスについて説明します。

表領域

- データベース・クラスタが初期化されると、2つの表領域が自動的に作成されます。`pg_global` 表領域は共有システムカタログに使用されます。`pg_default` 表領域は、template1およびtemplate0データベースのデフォルトの表領域です。クラスタが初期化されたパーティションまたはボリュームの容量が不足し、拡張できない場合は、別のパーティションに表領域を作成して、システムを再構成できるようになるまで使用できます。
- 頻繁に使用されるインデックスは、ソリッドステートデバイスのような高速で可用性の高いディスクに配置できます。また、ほとんど使用されない、またはパフォーマンスが重要でないアーカイブデータを格納するテーブルは、SASドライブやSATAドライブなどの低コストで低速なディスクシステムに格納できます。
- 表領域はデータベースクラスタの一部であり、データファイルの自律的なコレクションとして扱うことはできません。これらは、メインデータディレクトリに含まれるメタデータに依存するため、別のデータベースクラスタに接続したり、個別にバックアップしたりすることはできません。同様に、（ファイル削除やディスク障害などによって）テーブルスペースが失われると、データベースクラスタが読み取り不能になったり、起動できなくなったりすることがあります。RAMディスクのような一時ファイルシステムに表領域を配置すると、クラスタ全体の信頼性が低下します。
- 作成後、要求元ユーザに十分な権限があれば、任意のデータベースから表領域を使用できます。PostgreSQLは、テーブルスペースの実装を簡素化するためにシンボリックリンクを使用します。PostgreSQLは `pg_tablespace` テーブル(クラスタ全体のテーブル)に行を追加し、その行に新しいオブジェクト識別子(OID)を割り当てます。最後に、サーバはOIDを使用して、クラスタと指定されたディレクトリの間にシンボリックリンクを作成します。ディレクトリ `$PGDATA/pg_tblspc` には、クラスタで定義されている組み込み以外の各表領域を指すシンボリックリンクが含まれています。
- パフォーマンスを向上させるために、NetAppでは、論理ファイルシステムごとに最大1つの表領域を保持することを推奨しています。また、論理ファイルシステム内の個々のファイルの場所を制御することはできません。ただし、PostgreSQLはこのような制限を適用しておらず、システム上のファイルシステムの境界を直接認識していません。使用するように指示したディレクトリにファイルを保存するだけです。

ファイルシステムオプション

- NetAppでは、バージョン3にはない統合ロック機能があるため、バージョン4のNFSプロトコルを使用することを推奨しています。また、ステートフル、セキュリティの向上、強力な認証などの新機能も備えています。
- PostgreSQLでは、NFSファイルシステムのソフトマウントは推奨していません。

- PostgreSQLでは、データの不整合の問題を回避するために、NFSファイルシステムを同期的にマウントすることも推奨しています。詳細はPostgreSQLのドキュメントを参照してください。
- NFSファイルシステムのマウントには、次のマウントオプションを使用します。

```
nfs4 rw, hard,nointr,bg,vers=4,sync,proto=tcp,noatime,rsize=65536,wsize=65536
```

- NetAppでは、データファイルI/Oをサポートするために4~8個のLUNを使用することを推奨しています。
- NetApp LUNは、4KBの物理ブロックにデータを格納します。同じブロックサイズを設定しないと、I/Oが物理ブロックと正しくアライメントされず、RAIDグループ内の2つの異なるディスクに書き込まれる可能性があります。2本のディスクに書き込むとレイテンシが発生するため、NetAppでは、検出されたLUNに4KBのブロックサイズを使用してパーティションを作成することを推奨しています。

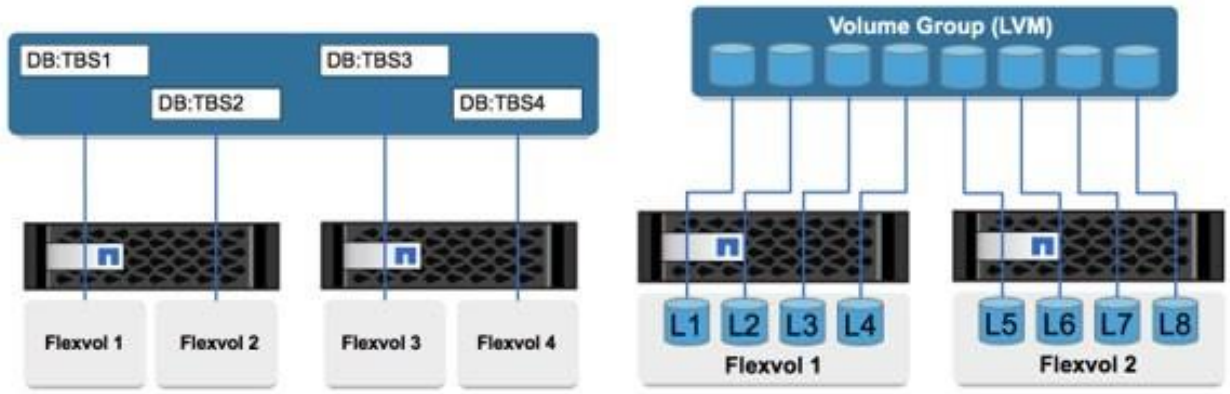
PostgreSQLのパフォーマンスのベストプラクティス

効率的でパフォーマンスに優れたデータベースファイルシステムのレイアウトを設計することは、パフォーマンスと拡張性の両方の観点から重要ですが、データベースのサイズは重要ではありません。大規模なデータベースには、ハイパフォーマンスなディスクアーキテクチャが必要であるという誤解があります。データベースサイズが50GBであっても、適切なディスクアーキテクチャが必要です。

データベースレイアウトの設計に関するヒントを次に示します。

- データベースに複数のテーブルスペースがあり、トランザクションレートに従ってテーブルとインデックスがグループ化されていることを確認します。
- I/Oを分散させるには、テーブルスペースを複数のディスクファイルシステムに配置する必要があります。このバランスを実現するには、LVMを使用するか、重要なテーブルを別のボリュームまたはLUNに配置します。（ボリュームまたはLUNは、同じアグリゲートにも別のアグリゲートにも配置できます）。このようにテーブルを配置することで、図7に示すように、複数のCPUが複数のディスク間でトランザクションを実行できるようになります。

図7）複数のディスク（NFS、SAN）に格納されたユーザ定義の表領域



- `pg_xlog` または `pg_wal` ディレクトリを高トランザクションデータベースの別のディスク（ボリュームまたはLUN）に配置することを検討してください。
- I/O監視ツールを使用して、すべてのディスクのI/O統計を把握し、それに応じてデータベースオブジェクトを管理します。

データベース構成のベストプラクティス

パフォーマンスを向上させるPostgreSQLのチューニング設定がいくつかあります。最も一般的に使用されるパラメータは次のとおりです。

- `max_connections = <num>`:一度に持つデータベース接続の最大数。このパラメータを使用して、ディスクへのスワップを制限し、パフォーマンスを強制終了します。アプリケーションの要件に応じて、このパラメータを接続プールの設定に合わせて調整することもできます。

- `shared_buffers = <num>`：データベース・サーバのパフォーマンスを向上させる最も簡単な方法最新のほとんどのハードウェアでは、デフォルトはlowです。導入時に、システム上の使用可能なRAMの約25%に設定されます。このパラメータ設定は、特定のデータベースインスタンスでの動作によって異なります。試行錯誤して値を増減しなければならない場合があります。ただし、この値をHighに設定すると、パフォーマンスが低下する可能性があります。
- `effective_cache_size = <num>`:この値は、PostgreSQLのオプティマイザに、PostgreSQLがデータをキャッシュするために使用できるメモリ量を伝え、インデックスを使用するかどうかを判断するのに役立ちます。値を大きくすると、インデックスを使用する可能性が高くなります。このパラメータには、割り当てられたメモリ容量 `shared_buffers` と使用可能なOSキャッシュ容量を加えた値を設定する必要があります。多くの場合、この値はシステムメモリ全体の50%を超えています。
- `work_mem = <num>`:このパラメータは、ソート操作およびハッシュテーブルで使用するメモリの量を制御します。アプリケーションで大量のソートを行う場合は、メモリの量を増やす必要があるかもしれませんが、注意が必要です。これはシステム全体のパラメータではなく、操作ごとのパラメータです。複雑なクエリに複数のソート操作が含まれている場合、複数の `work_mem` メモリユニットを使用し、複数のバックエンドが同時にこれを実行する可能性があります。このクエリを実行すると'値が大きすぎると'データベース・サーバがスワップされることがよくありますこのオプションは、以前は `sort_mem` PostgreSQLの古いバージョンで呼び出されていました。
- `fsync = <boolean> (on or off)`:このパラメータは、`fsync()`トランザクションがコミットされる前にを使用して、すべてのWALページをディスクに同期するかどうかを決定します。電源をオフにすると、書き込みパフォーマンスが向上し、オンにすると、システムクラッシュ時の破損のリスクからの保護が強化されます。
- CHECKPOINT_TIMEOUT：チェックポイントプロセスは、コミットされたデータをディスクにフラッシュします。これには、ディスク上で多くの読み取り/書き込み処理が含まれます。値は秒単位で設定され、値を小さくするとクラッシュリカバリ時間が短縮されます。値を大きくすると、チェックポイントコールが削減されるため、システムリソースの負荷が軽減されます。アプリケーションの重要度、使用状況、データベースの可用性に応じて、checkpoint_timeoutの値を設定します。
- `commit_delay = <num>` および `commit_siblings = <num>`:これらのオプションを組み合わせて使用すると、一度にコミットする複数のトランザクションを書き出すことで、パフォーマンスを向上させることができます。`commit_siblings` トランザクションがコミットされた瞬間に複数のオブジェクトがアクティブになっている場合、サーバは `commit_delay` マイクロ秒待機して複数のトランザクションを一度にコミットしようとします。
- max_worker_processes/max_parallel_workers :プロセスの最適なワーカー数を設定します。max_parallel_workersは、使用可能なCPUの数に対応します。アプリケーションの設計によっては、クエリで並列処理に必要なワーカーの数が少なくて済みます。両方のパラメータの値は同じにし、テスト後に値を調整することをお勧めします。
- `random_page_cost = <num>`:この値は、PostgreSQLが非シーケンシャルディスク読み取りを表示する方法を制御します。値を大きくすると、PostgreSQLはインデックススキャンではなくシーケンシャルスキャンを使用する可能性が高くなります。これは、サーバーに高速ディスクがあることを示します。計画ベースの最適化、真空化、クエリやスキーマの変更に対するインデックス付けなど、他のオプションを評価した後で、この設定を変更してください。
- `effective_io_concurrency = <num>`:このパラメータは、PostgreSQLが同時に実行を試みる同時ディスクI/O処理の数を設定します。この値を大きくすると、個 々 のPostgreSQLセッションが並行して開始しようとするI/O処理の数が増加します。指定できる範囲は1～1、000です。非同期I/O要求の発行を無効にする場合は0にします。現在、この設定はビットマップヒープスキャンにのみ影響します。ソリッドステートドライブ（SSD）やその他のメモリベースストレージ（NVMe）は、多数の同時要求を処理できることが多いため、数百の数を推奨します。

PostgreSQL設定パラメータの完全なリストについては、PostgreSQLのドキュメントを参照してください。

ベストプラクティスのまとめ

ONTAPデータ管理ソフトウェアにPostgreSQLを導入する際のベストプラクティスと推奨事項を以下に示します。

ベストプラクティス

- マルチテナントのPostgreSQL環境では、各テナントのデータ（データベースまたはインスタンス）をONTAPストレージ上に専用のSVMを配置できます。
- NFS LIFの設計では、LACPとIPのロードバランシングを有効にする方法が最も一般的です。それ以外の場合は、通常の方法でフェイルオーバーポリシーとフェイルオーバーグループを設定する必要があります。
- SANマルチパス環境では、高可用性を確保するために、NetApp LUNホストを異なるスイッチからゾーニングすることを推奨します。
- FCゾーンに複数のイニシエータを含めることはできません。イニシエータ間のクロストークは、最終的にパフォーマンスと安定性を妨げます。
- HAクラスタでLUNまたはボリュームを移動するときは、移動イベントを開始する前に、デスティネーションストレージコントローラを使用してLUNを提供する必要があります。移動する前に、選択的LUNマップのレポートノードリストを変更します。
- ミッションクリティカルな環境では、パフォーマンスを向上させるために、NetAppではデータファイルとWALログを専用のボリュームまたはLUNに保持することを推奨しています。この環境は、SANレイアウトとNFSレイアウトの両方に対応しています。
- データファイルとWALログを専用ボリュームに分離することで、データベースのリカバリを高速化することもできます。
- データベースのパフォーマンスを向上させるには、重要な表領域を専用のボリュームまたはLUNに分離します。
- データストアを選択すると、全体的な管理性や運用オーバーヘッドが向上する可能性があります。
- データストアのタイムアウト、パラメータ、ログファイル、および潜在的なバグが増えることで、さらに複雑さが増す可能性があります。
- VMFSデータストアではなくNFSを選択すると、災害時にNetApp SnapRestoreテクノロジを使用する際に役立ちます。
- 頻繁に使用されるインデックスは、高価なソリッドステートデバイスなど、高速で可用性の高いディスクに配置できます。また、ほとんど使用されない、またはパフォーマンスが重要でないアーカイブデータを格納するテーブルは、低コストで低速なディスクシステムに格納できます。
- 表領域はメイン・データ・ディレクトリ内のメタデータに依存するため、別のデータベース・クラスタに接続したり、個別にバックアップしたりすることはできません。同様に、（ファイル削除やディスク障害などによって）テーブルスペースが失われると、データベースクラスタが読み取り不能になったり、起動できなくなったりすることがあります。RAMディスクのような一時ファイルシステムに表領域を配置すると、クラスタ全体の信頼性が低下します。
- NetAppでは、論理ファイルシステム内の個々のファイルの場所を制御できないため、論理ファイルシステムごとに複数の表領域を作成することは推奨されていません。
- NetAppでは、バージョン3にはない統合ロック機能があるため、バージョン4のNFSプロトコルを使用することを推奨しています。また、ステートフル、セキュリティの向上、強力な認証などの新機能も備えています。
- PostgreSQLでは、NFSファイルシステムのソフトマウントは推奨していません。
- PostgreSQLでは、データの不整合の問題を回避するために、NFSファイルシステムを同期的にマウントすることも推奨しています。詳細については、PostgreSQLのドキュメントを参照してください。
- データベースに複数のテーブルスペースがあり、トランザクションレートに従ってテーブルとインデックスがグループ化されていることを確認します。
- I/Oを分散させるには、テーブルスペースを複数のディスクファイルシステムに配置する必要があります。このバランスにより、複数のCPUを使用して複数のディスク間でトランザクションを実行できます。

付録A：ホストの設定

- ESXホストのFC Host Bus Adapter（HBA；ホストバスアダプタ）が実行されており、ドライバ、ファームウェア、および正しいBIOSバージョンが更新されていることを確認します。
- ホストにNetApp Host Utilitiesをインストールして、特定のLUNの直接パスと間接パスを表示します。キットには、LUN SVM情報に関する詳細情報も含まれています。
- ホストでは必ず再スキャンSCSIバススクリプトを使用して、次の処理を実行します。
  - 新しいLUNの検索
  - 古いLUNを破棄する
  - LUNへの新しいパスの更新
- パス障害が発生した場合にストレージへの複数のルートを確立するには、ホストオペレーティングシステムにマルチパスソフトウェアをインストールして設定します。
- マルチパスソフトウェアを使用しない場合は、各LUNのパスを1つに制限する必要があります。
- Linuxカーネルでは、ブロックデバイスへのI/Oのスケジュールを低レベルで制御できます。デフォルトの設定はLinuxのバージョンによって大きく異なる場合があります。一般に、NetAppをご利用のお客様や社内テストでは、データベース向けのNoOpsスケジューラを使用した方が効果的です。ベンチマークを実行して、ユースケースに最適なI/Oスケジューラを特定する必要があります。

SELinux

Security-Enhanced Linux（SELinux）は、アクセス制御セキュリティポリシーをサポートするメカニズムを提供するLinuxカーネルセキュリティモデルです。

`/etc/selinux/config` 設定ファイルは、SELinuxを有効にするか無効にするかを制御し、有効にすると、SELinuxがパーミッシブモードで動作するか強制モードで動作するかを制御します。`SELINUX` 変数は、無効、許可、または強制モードに設定できます。

- disabledオプションを指定すると、SELinuxカーネルとアプリケーションコードが完全に無効になり、システムはSELinux保護なしで実行されたままになります。
- permissiveオプションを使用すると、SELinuxコードが有効になり、ポリシー拒否アクセスは許可されますが、監査は許可されます。
- enforcingオプションを指定すると、SELinuxコードが有効になり、アクセス拒否を強制して監査されます。
- このテストでは、ほとんどの導入環境で、LinuxホストにNFSファイルシステムをマウントするとき、またはPostgreSQLデーモンを起動するときにエラーが発生していました。一般的なエラーは次のとおりです。

```
Error message: not found; Error message: access denied;
Error message: Permission denied; Cannot mount / access an NFS volume or file system; Error
Message: Cannot find home Directory
```

これらのエラーを解決するには、SELinux構成ファイルを修正し、NetApp ONTAPのエクスポートオプションが正しいことを確認します。その他のトラブルシューティング手順については、NetAppの技術情報アーティクルを参照してください。

FC構成の詳細については、次のドキュメントを参照してください。

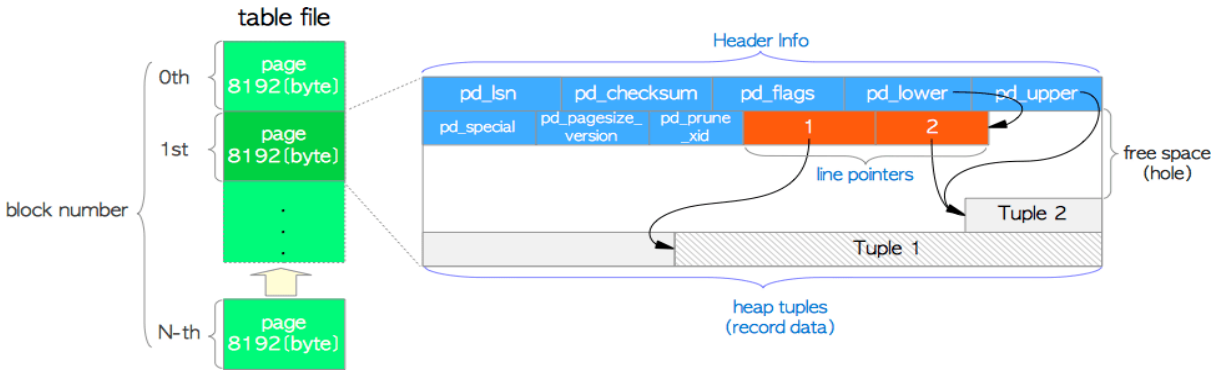
- SAN構成
  クラスタ内のストレージ コントローラにホスト コンピュータを接続するための、FC、iSCSI、およびFCoEトポロジについて説明しています。
- SANの管理
  クラスタSAN環境で使用するiSCSI、FCoE、およびFCの各プロトコルを設定および管理する方法（LUN、イニシエータグループ（igroup）、ターゲットの設定など）について説明します。

付録B：PostgreSQLのページとファイルシステムのレイアウト

すべてのテーブルとインデックスは固定サイズ（通常は8KB）のページの配列として格納されますが、構成を変更することで別のページサイズを選択できます。各ページには、ブロックに関するメタデータを格納するヘッダーが含まれています。ヘッダーには、チェックサム、空きスペースの開始、および空きスペースの終了に関する詳細が表示されます。ヘッダーの後ろの項目は、実際の項目を指すオフセット長ペアで構成される配列識別子です。

テーブルの格納に使用される構造体はヒープファイルです。ヒープファイルは、可変サイズの順序付けられていないレコードのリストです。ヒープファイルは、ページまたはブロックのコレクションとして構成され、それぞれにアイテムのコレクションが含まれています。図8 では、PostgreSQLが行全体、およびテーブルの各ページまたはブロックにどのようにデータを保存するかを詳しく見ていきます。

図8）PostgreSQLのデータストレージ



トースト

TOASTは、特大属性ストレージテクニックを表しています。PostgreSQLは固定のページサイズ（通常は8KB）を使用しており、タプルを複数のページにまたがることはできません。したがって、大きなフィールド値を直接保存することはできません。このサイズを超える行を格納しようとすると、トーストは大きな列のデータを小さな「ピース」に分割してトーストテーブルに格納します。

トーストされた属性の大きな値は'結果セットがクライアントに送信されるときにのみ(選択されている場合)プルアウトされますテーブル自体は非常に小さく、アウトオブラインストレージ（TOAST）を使用しない場合よりも多くの行を共有バッファキャッシュに格納できます。

バキューム

通常のPostgreSQL操作では、更新によって削除または廃止されたタプルはテーブルから物理的に削除されず、 VACUUM が実行されるまで存在したままになります。そのため VACUUM 、特に頻繁に更新されるテーブルでは、を定期的に実行する必要があります。ディスクスペースが使用されているスペースは、ディスクスペースが停止しないように、新しい行で再利用できるように再利用する必要があります。ただし、スペースはオペレーティングシステムに返されません。

ページ内の空き領域は断片化されません。VACUUM ブロック全体を書き換え、残りの行を効率的にパッキングして、1つの連続した空き領域ブロックをページに残します。

対照的に、は、 VACUUM FULL デッドスペースのないまったく新しいバージョンのテーブルファイルを書き込むことによって、積極的にテーブルを圧縮します。この操作により、テーブルのサイズは最小限に抑えられますが、時間がかかることがあります。また、処理が完了するまで、テーブルの新しいコピー用に追加のディスクスペースが必要になります。ルーチンの目的 VACUUM は VACUUM FULL 活動を避けることです。このプロセスでは、テーブルが最小サイズに維持されるだけでなく、ディスクスペースの安定した使用量も維持されます。

PostgreSQLファイルシステムのレイアウト

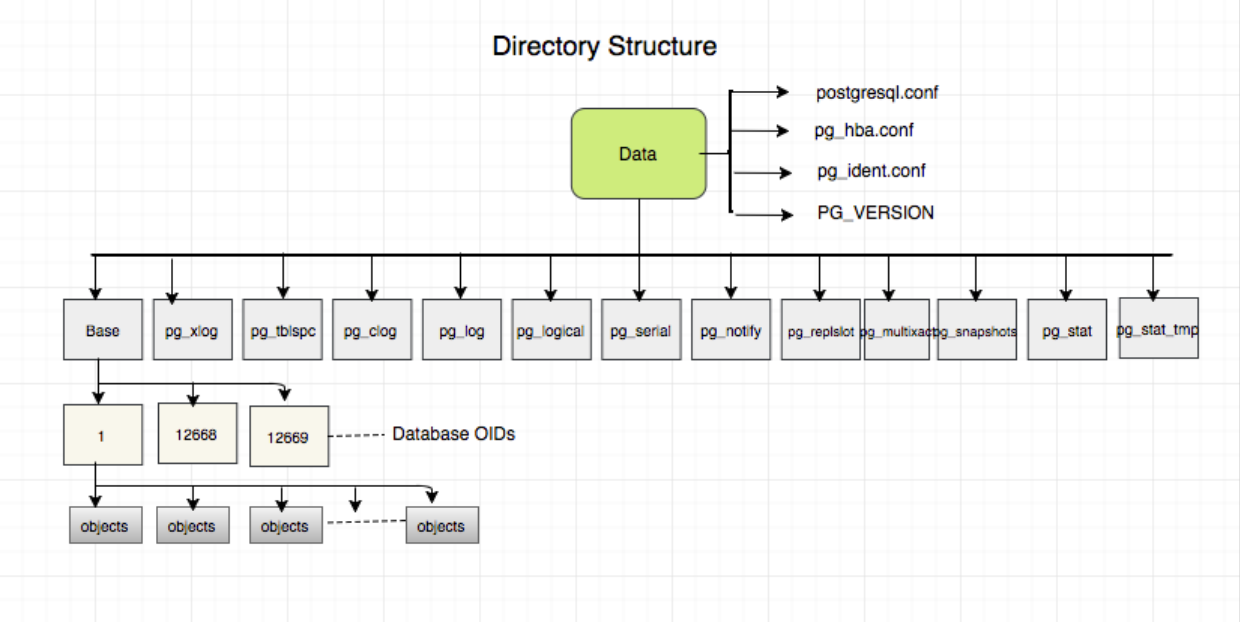
`initdb` プログラムを使用して、新しいデータベースクラスタを作成します。`initdb` スクリプトは、クラスタを定義するデータファイル、システムテーブル、およびテンプレートデータベース（template0およびtemplate1）を作成します。テンプレートデータベースはストックデータベースを表します。システムテーブル、標準ビュー、関数、およびデータ型の定義が含まれています。`pgdata initdb` データベースクラスタの場所を指定するスクリプトの引数として機能します。

PostgreSQLのすべてのデータベースオブジェクトは、それぞれのOIDによって内部的に管理されます。テーブルとインデックスは、個々のOIDによっても管理されます。データベースオブジェクトとそれぞれのOIDとの関係は、オブジェクトのタイプに応じて適切なシステムカタログテーブルに格納されます。たとえば、データベースとヒープテーブルのOIDは `pg_database`、`pg_class`それぞれとに格納されます。OIDを確認するには、PostgreSQLクライアントでクエリを発行します。

各データベースには、1GBに制限された個別のテーブルとインデックスファイルがあります。各テーブルには、それぞれ `_fsm` とのサフィックス付きの2つのファイルが関連付けられ `_vm`ています。これらは、フリースペースマップおよび可視性マップと呼ばれます。これらのファイルには空きスペース容量に関する情報が格納され、テーブルファイルの各ページに表示されます。インデックスには個々の空き領域マップのみがあり、可視性マップはありません。

`pg_xlog/pg_wal` ディレクトリには、先行書き込みログが格納されます。先行書き込みログは、データベースの信頼性とパフォーマンスを向上させるために使用されます。テーブル内の行を更新するたびに、PostgreSQLは先読みログに変更内容を書き込み、その後実際のデータページに変更内容をディスクに書き込みます。`pg_xlog` 通常、ディレクトリには複数のファイルが含まれていますが、`initdb` 最初のファイルだけが作成されます。必要に応じて追加のファイルが追加されます。各xlogファイルの長さは16MBです。

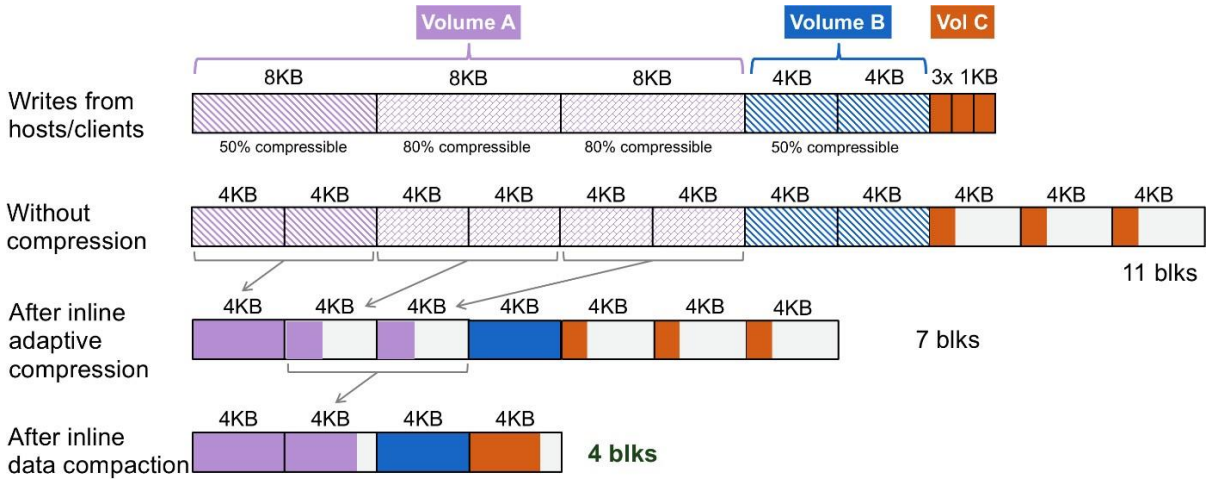
図9）ディレクトリ構造



付録C：Storage Efficiency

図10 に、ONTAPストレージ上のデータベースで圧縮とコンパクションがどのように機能するかを示します。

図10）圧縮とコンパクション



本ドキュメントに記載されている製品や機能のバージョンがお客様の環境でサポートされるかどうかについては、NetApp サポート サイトで Interoperability Matrix Tool (IMT)を参照してください。NetApp IMT には、NetApp がサポートする構成を構築するために使用できる製品コンポーネントやバージョンが定義されています。サポートの可否は、お客様の実際のインストール環境が公表されている仕様に従っているかどうかによって異なります。

機械翻訳に関する免責事項

原文は英語で作成されました。英語と日本語訳の間に不一致がある場合には、英語の内容が優先されます。公式な情報については、本資料の英語版を参照してください。翻訳によって生じた矛盾や不一致は、法令の順守や施行に対していかなる拘束力も法的な効力も持ちません。

著作権に関する情報

商標に関する情報

TR-4770-0423-JP

NetApp