



High Availability Routing for Postgres -HARP

リリース 2.1.1

EnterpriseDB Corporation

2022 年 07 月 29 日

目次

第 1 章	Release Notes	3
1.1	Version 2.0.1	3
1.2	Version 2.0.2	4
1.3	Version 2.0.3	4
1.4	Version 2.1.0	5
1.5	Version 2.1.1	5
1.6	Version 2.0.0	6
第 2 章	HARP functionality overview	7
2.1	クォラムの重要性	7
2.2	書き込みターゲットの削減	7
2.3	基本的なアーキテクチャ	8
2.4	仕組み	9
2.4.1	リーダーの決定	11
2.4.2	接続ルーティング	11
2.4.3	コロケーション	12
2.5	推奨されるアーキテクチャと使用	12
2.5.1	ロケーション構成	14
2.5.2	BDR 3.7 との互換性	14
第 3 章	Installation	15
3.1	ソフトウェアバージョン	15
3.2	TPAExec	15
3.3	パッケージのインストール	16
3.3.1	etcd パッケージ	16
3.4	コンセンサスレイヤー	16
3.4.1	etcd	16
3.4.2	BDR	17
第 4 章	Configuring HARP for cluster management	19
4.1	標準構成	19
4.1.1	クラスター名	20
4.1.2	DCS 設定	20
第 5 章	Cluster bootstrapping	21
5.1	クラスター全体のブートストラップ	21
5.2	ロケーションブートストラップ	22
5.3	ノードのブートストラップ	22
5.4	プロキシブートストラップ	23
第 6 章	HARP Manager	25

6.1	仕組み	25
6.2	構成	26
6.3	使用法	27
6.4	Postgres の HARP Manager 制御の無効化と再有効化	27
第 7 章	HARP Proxy	29
7.1	ビルトインプロキシ：仕組み	29
7.1.1	構成	29
7.2	PgBouncer：仕組み	30
7.2.1	PgBouncer 構成ファイル	30
7.2.2	HARP プロキシノード管理の無効化と再有効化	30
7.2.3	パススルーユーザー認証	31
7.2.4	構成	31
第 8 章	harpctl command-line tool	33
8.1	概要	33
8.2	構成	34
8.3	実行	34
8.4	harpctl apply	34
8.5	harpctl fence	35
8.6	harpctl get	36
8.6.1	harpctl get cluster	36
8.6.2	harpctl get leader	36
8.6.3	harpctl get location	37
8.6.4	harpctl get locations	37
8.6.5	harpctl get node	37
8.6.6	harpctl get nodes	38
8.6.7	harpctl get proxy	38
8.6.8	harpctl get proxies	38
8.7	harpctl manage	39
8.8	harpctl promote	39
8.9	harpctl set	40
8.9.1	harpctl set cluster	40
8.9.2	harpctl set node	40
8.9.3	harpctl set proxy	40
8.10	harpctl unfence	41
8.11	harpctl unmanage	41
第 9 章	Consensus layer considerations	43
9.1	BDR ドライバーの互換性	43
9.2	クォーラムの維持	43
9.2.1	マルチコンセンサスバリエーション	44
9.2.2	TPAexec とコンセンサス	45
第 10 章	Security and roles	47
10.1	Postgres のアクセス許可	47
10.2	BDR のアクセス許可	47
10.3	BDR コンセンサス権限	48

Postgres の高可用性ルーティング (HARP) は、Bi-directional Replication (BDR) クラスター用のクラスター管理ツールです。このツールのコアデザインは、単一のデータセンターまたはリージョン内のすべてのアプリケーショントラフィックを一度に 1 つのノードにのみルーティングすることです。リードマスターと指定されたこのノードは、データ競合の可能性を減らすための主な書き込みターゲットとして機能します。

HARP は分散コンセンサスモデルを活用して、クラスター内の BDR ノードの可用性を判断します。リードマスターに障害が発生するか使用できない場合、HARP は新しいリードマスターを選択し、アプリケーショントラフィックをリダイレクトします。

BDR のコア機能と合わせて、アプリケーショントラフィックをリードマスターノードにルーティングするこのメカニズムにより、データ損失のリスクなしで高速なフェイルオーバーとスイッチオーバーが可能になります。

HARP には BDR バージョン 3.6 以降が必要です。

第1章 Release Notes

Postgres の高可用性ルーティング (HARP) は、Bi-directional Replication (BDR) クラスター用のクラスター管理ツールです。このツールのコアデザインは、単一のデータセンターまたはリージョン内のすべてのアプリケーショントラフィックを一度に 1 つのノードにのみルーティングすることです。リードマスターと指定されたこのノードは、データ競合の可能性を減らすための主な書き込みターゲットとして機能します。

このセクションのリリースノートには、各リリースの新機能に関する情報が記載されています。

Version	Release Date
<i>Version 2.1.1</i>	2022 June 21
<i>Version 2.1.0</i>	2022 May 17
<i>Version 2.0.3</i>	2022 Mar 31
<i>Version 2.0.2</i>	2022 Feb 24
<i>Version 2.0.1</i>	2021 Jan 31
<i>Version 2.0.0</i>	2021 Dec 01

1.1 Version 2.0.1

これは、以前のバージョンで特定された問題の修正を含む HARP 2 のパッチリリースです。

Type	Description
Enhancement	etcd のような DCS に依存して異なるロケーションに個別にセットアップするのではなく、ロケーションごとにリーダーを選択するためのサポート。それでも、ロケーションの損失を生き残るために大部分のノードが必要になるため、ロケーションとデータベースノードの両方を奇数にすることをお勧めします。
Enhancement	BDR DCS は、ポーリングノードではなく、コンセンサスからのプッシュ通知を使用するようになりました。この変更により、新しいリーダーの選択にかかる時間が短縮され、HARP が BDR DCS で実行する負荷が削減されます。
Enhancement	TPA は各 HARP プロキシを 1 つずつ再起動し、戻ってくるまで待機して、ソフトウェアのアップグレード中にアプリケーションで発生したダウンタイムを削減します。
Enhancement	PGBouncer を HARP プロキシに直接埋め込むサポートは非推奨になり、HARP の次のメジャーリリースで削除されます。PGBouncer を HARPHARP をポイントするように TPA を構成できるようになりました。

次のページに続く

表 2 – 前のページからの続き

Type	Description
Bug Fix	<code>harectl promote <node_name></code> は、指定されたノードとは異なるノードを昇格させることができました。これは修正されました。 [サポートチケット # 75406]
Bug Fix	分散コンセンサスサービスとして BDR を使用すると、フェンシングが失敗することがありました。これは修正されました。
Bug Fix	<code>harectl apply</code> は、クラスターが確立された後にリーダーのルーティングをオフにになりました。 [サポートチケット # 80790]
Bug Fix	障害が発生したデータベースを起動できない場合に、Harp-manager が終了してなくなりました。Harp-manager は、ランダムに増加する期間で再試行し続けます。 [サポートチケット # 78516]
Bug Fix	内部 pgbouncer プロキシ実装でメモリリークが発生しました。これは修正されました。

1.2 Version 2.0.2

これは、以前のバージョンで特定された問題の修正を含む HARP 2 のパッチリリースです。

Type	Description
Enhancement	BDR コンセンサスが一般提供になりました。 HARP は、分散コンセンサスサービス (DCS) ソースに複数のオプションを提供します。BDR。BDR コンセンサスオプションは、etcd が存在しないデプロイメントで使用できます。BDR コンセンサスオプションの使用はベータとは見なされなくなり、実稼働環境での使用がサポートされるようになりました。
Enhancement	トランスポートレイヤープロキシの一般提供が開始されました。 HARP は、クライアントアプリケーションとデータベース間の接続をルーティングするための複数のプロキシオプションを提供します。アプリケーションレイヤー (L7) とトランスポートレイヤー (L4)。ネットワークレイヤー 4 またはトランスポートレイヤープロキシはネットワークパケットを転送するだけで、レイヤー 7 はネットワークトラフィックを終了します。以前はシンプルプロキシと呼ばれていたトランスポートレイヤープロキシは、ベータ版とは見なされなくなり、実稼働環境での使用がサポートされるようになりました。

1.3 Version 2.0.3

これは、以前のバージョンで特定された問題の修正を含む HARP 2 のパッチリリースです。

Type	Description
Enhancement	HARP プロキシは読み取り専用のユーザー専用 TLS 証明書をサポート (RT78516)

次のページに続く

表 4 – 前のページからの続き

Type	Description
Bug Fix	HARP プロキシは、50 秒後に終了する代わりに DCS への接続を試行し続けます。(RT75406)

1.4 Version 2.1.0

これは HARP 2 のマイナーリリースで、新機能と、以前のバージョンで特定された問題の修正が含まれています。

Type	Description
Feature	BDR DCS は、ポーリングノードではなく、コンセンサスからのプッシュ通知を使用するようになりました。 この変更により、新しいリーダーの選択にかかる時間が短縮され、HARP が BDR DCS で行う負荷が削減されます。これは、短い間隔でポーリングする必要がなくなったためです。
Feature	TPA は各 HARP プロキシを 1 つずつ再起動し、戻ってくるまで待機して、ソフトウェアのアップグレード中にアプリケーションで発生したダウンタイムを削減します。
Feature	PGBouncer を HARP プロキシに直接埋め込むサポートは非推奨になり、HARP の次のメジャーリリースで削除されます。 PGBouncer を HARP プロキシと同じノードに配置し、その HARP プロキシをポイントするように TPA を構成できるようになりました。
Bug Fix	<code>`harectl promote <node_name>`</code> は、指定されたノードとは異なるノードを昇格させることがありました。これは修正されました。(RT75406)
Bug Fix	分散コンセンサスサービスとして BDR を使用すると、フェンシングが失敗することがありました。これは修正されました。
Bug Fix	<code>`harectl apply`</code> は、クラスターが確立された後にリーダーのルーティングをオフにになりました。(RT80790)
Bug Fix	障害が発生したデータベースを起動できない場合に、Harp-manager が終了してなくなりました。Harp-manager は、ランダムに増加する期間で再試行し続けます。(RT78516)
Bug Fix	内部 pgbouncer プロキシ実装でメモリリークが発生しました。これは修正されました。

1.5 Version 2.1.1

これは、以前のバージョンで特定された問題の修正を含む HARP 2 のパッチリリースです。

Type	Description
Enhancement	DCS 接続の損失に関する警告をログに記録する

次のページに続く

表 6 – 前のページからの続き

Type	Description
Enhancement	メタデータの更新に時間がかかりすぎる場合に警告をログに記録します - 通常は高遅延ネットワークが原因です
Bug Fix	障害時に harp_proxy.service をリスタート
Bug Fix	haproctl の接続管理の同時実行の問題を修正
Bug Fix	アンマネージドクラスターで以前のリーダーへの接続をプロキシしないください
Bug Fix	場所が空の場合に haproctl でパニックにならない

1.6 Version 2.0.0

これは、製品の完全な書き換えで構成される HARP の新しいメジャーリリースです。

Type	Description
Engine	すべての操作を最適化するための golang のシステムの完全な書き換え
Engine	クラスターの状態を YAML 経由でブートストラップまたは修正できるようになりました
Feature	構成が YAML になり、構成ファイルが `harp.ini` から `config.yml` に変更
Feature	HARP プロキシは、サポートされているアーキテクチャでの HAProxy の必要性を非推奨にします。 HARP ルーターを使用して DCS コンテンツを HTTP ベースの URI 要求の適切なオンラインまたはオフライン状態に変換することは、リードマスターを決定するためにロードバランサーまたは HAProxy が必要であることを意味しました。 HARP プロキシは、定期的な反復ステータスチェックなしでこれを自動的に行うようになりました。
Feature	DCS キーサブスクリプションを利用して、状態の変更に直接応答します。 関連するクラスターの状態の変更により、クラスターはすぐに応答するため、フェールオーバーとスイッチオーバーの時間が向上します。
Feature	etcd SSL 設定との互換性。 SSL 暗号化を介して etcd と通信できるようになりました。
Feature	スイッチオーバー時のトランザクションラグはゼロです。 レプリケートされたすべてのトランザクションが再生されるまで、トランザクションは新しいリードノードにルーティングされないため、競合の可能性が減少します。
Feature	実験的な BDR コンセンサスレイヤー。 分散コンセンサスサービス (DCS) として BDR コンセンサスを使用すると、実装に必要な変更の量が削減されます。
Feature	実験的なビルトインプロキシ。 セッション制御を強化するためのプロキシ実装。

第2章 HARP functionality overview

HARP は、BDR クラスターの高可用性に対する新しいアプローチです。コンセンサス駆動のクォーラムを活用して、半排他的な方法で正しい接続エンドポイントを決定し、アプリケーションからの意図しないマルチノード書き込みを防ぎます。

2.1 クォーラムの重要性

HARP の主な目的は、管理する Postgres クラスターにフルクォーラムを適用することです。クォーラムとは、決定を行うために特定の最小限の出席者を義務付ける投票機関に適用される用語です。より簡単に：多数決ルール。

同点以外の結果に終わるには、奇数のノードが完全なクラスターメンバーシップを構成する必要があります。ただし、クォーラムはこの制限を厳密に要求しません。単純過半数で十分です。これは、N 個のノードのクラスターでは、クォーラムが意味のある投票を保持するために少なくとも $N/2+1$ ノードが必要であることを意味します。

これらすべてにより、クラスターが「担当」するノードに関して常に合意することが保証されます。マルチプルノードで構成される BDR クラスターの場合、プライマリ書き込み対象のノードを決定します。HARP はこのノードをリードマスターとして指定します。

2.2 書き込みターゲットの削減

クォーラムの概念を無視するか、それを十分に適用しないと、「正しい」書き込みターゲットがあいまいまたは不明な「スプリットブレイン」シナリオにつながる可能性があります。標準の Postgres クラスターでは、1つのノードのみが書き込み可能で、残りのノードにレプリケーショントラフィックを送信することが重要です。

BDR などのマルチマスター対応のアプローチでも、クラスター全体で同一のデータを取得するために必要な競合管理の量を削減するのに役立ちます。物理的な場所またはリージョンごとに複数の BDR ノードで構成されるクラスターでは、これは通常、単一の BDR ノードが「リーダー」として機能し、残りのノードが「シャドウ」であることを意味します。これらのシャドウノードは引き続き書き込み可能ですが、どうしても必要な場合を除き、書き込みはお勧めしません。

クォーラムを活用することで、すべてのノードがクラスター全体またはローカル BDR リージョンを表す正確な Postgres ノードに同意することができます。定義により、クォーラムの残りの部分との接続を失ったノード、またはクォーラムによって無効にされたノードはクラスターリーダーになることはできません。

この制限により、書き込みが意図せずに2つの Postgres ノードに到達するスプリットブレイン状況が防止されます。VPN、プロキシ、ロードバランサー、DNS などのテクノロジーとは異なり、構成ミスやネット

ワークパーティションによってクォーラムから派生したコンセンサスを回避することはできません。コンセンサスレイヤーに接続して、HARP によって維持されるクォーラムの状態を判断できる限り、有効なターゲットは 1 つだけです。

2.3 基本的なアーキテクチャ

HARP のデザインは、基本的にマネージャーとプロキシの 2 つの部分で構成されています。次の図は、これらが単一の Postgres インスタンスとどのように相互作用するかを説明しています。

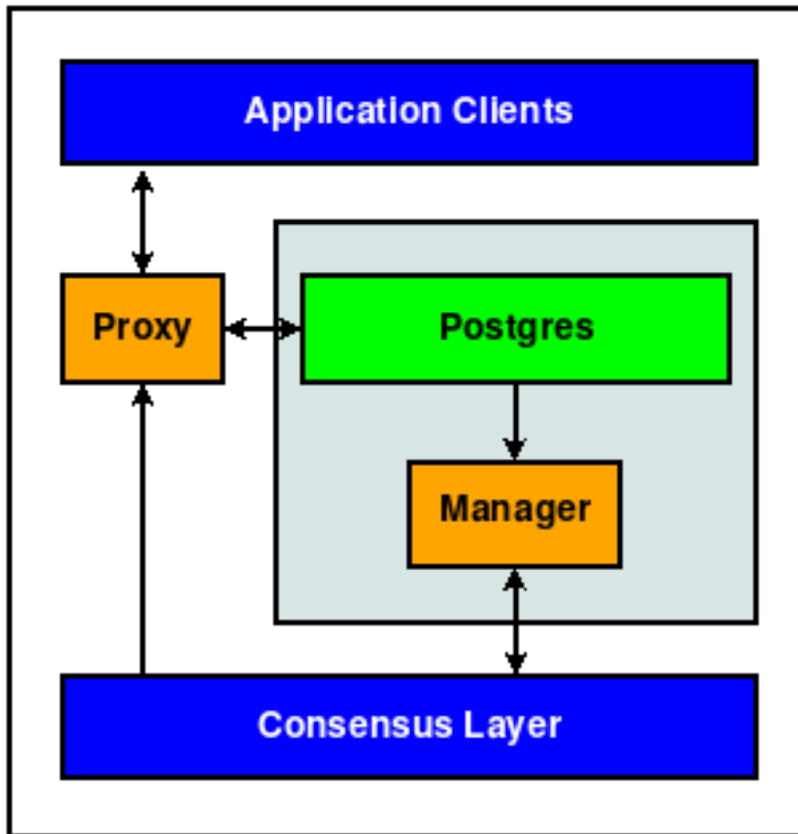


図 1: HARP Unit

コンセンサスレイヤーは、Harp Manager が割り当てられた Postgres ノードについて学習した情報を維持する外部エンティティであり、HARP プロキシはこの情報を有効な Postgres ノードターゲットに変換します。プロキシはコンセンサスレイヤーからノードターゲットを取得するため、このようなインスタンスが複数存在する場合があります。

コンセンサスレイヤーとして BDR を使用している間、各サーバーノードは代わりにこのバリエーションに似ています。

いずれの場合も、各ユニットは次の要素で構成されます。

- Postgres または EDB インスタンス
- Postgres インスタンスのさまざまな属性を追跡するためのコンセンサスレイヤーリソース
- Postgres ノードの状態をコンセンサスレイヤーに伝える HARP Manager プロセス

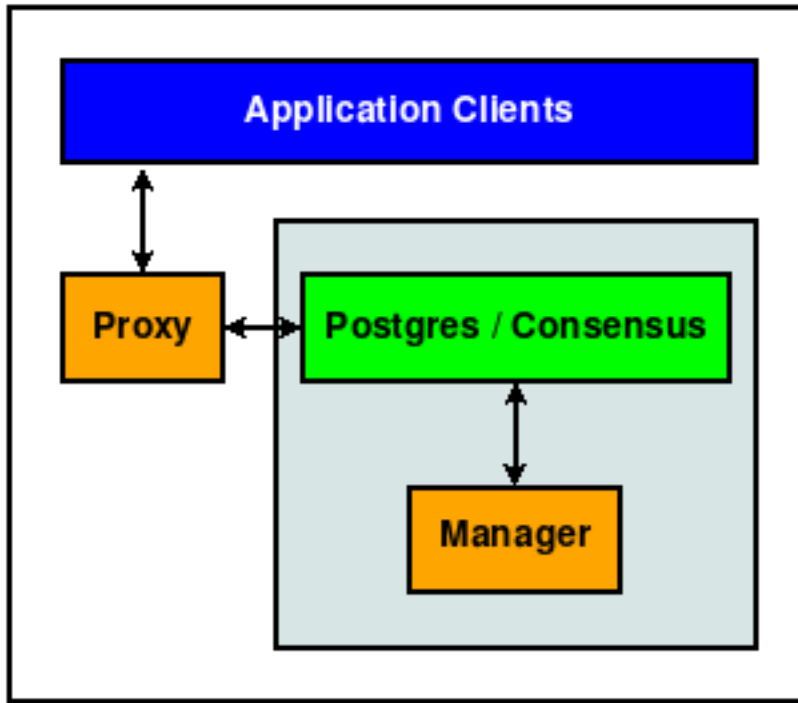


図 2: HARP Unit w/BDR Consensus

*コンセンサスレイヤーから派生した、適切なリードマスターノードにトラフィックを転送する HARP プロキシサービス

すべてのアプリケーションスタックがプロキシコンポーネント専用の追加ノードリソースにアクセスできるわけではないため、アプリケーションサーバーと組み合わせてスタックを簡素化できます。

これは、リードマスター/シャドウマスター構成で編成された単一のデータセンターで2つの BDR ノードを使用する一般的なデザインです。

BDR を HARP コンセンサスレイヤーとして使用する場合、クォーラムマジョリティを保証には、少なくとも3つの完全修飾 BDR ノードが存在する必要があります。(図には、BDR ノード間の接続は示されていません。)

2.4 仕組み

BDR クラスタを管理する場合、HARP は定義された場所ごとに最大1つのリーダーノードを維持します。これはリードマスターと呼ばれます。この位置を取得できる他の BDR ノードは、リーダーロールを取得するまでシャドウマスター状態です。

アプリケーションは、プロキシサービスを介してのみ現在のリーダーに連絡できます。コンセンサスレイヤーはリーダーの状態を伝える前にクォーラム契約を必要とするため、プロキシサービスはトラフィックをそのノードに転送します。

高レベルでは、このメカニズムは複数のノードとのアプリケーションの同時対話を防ぎます。

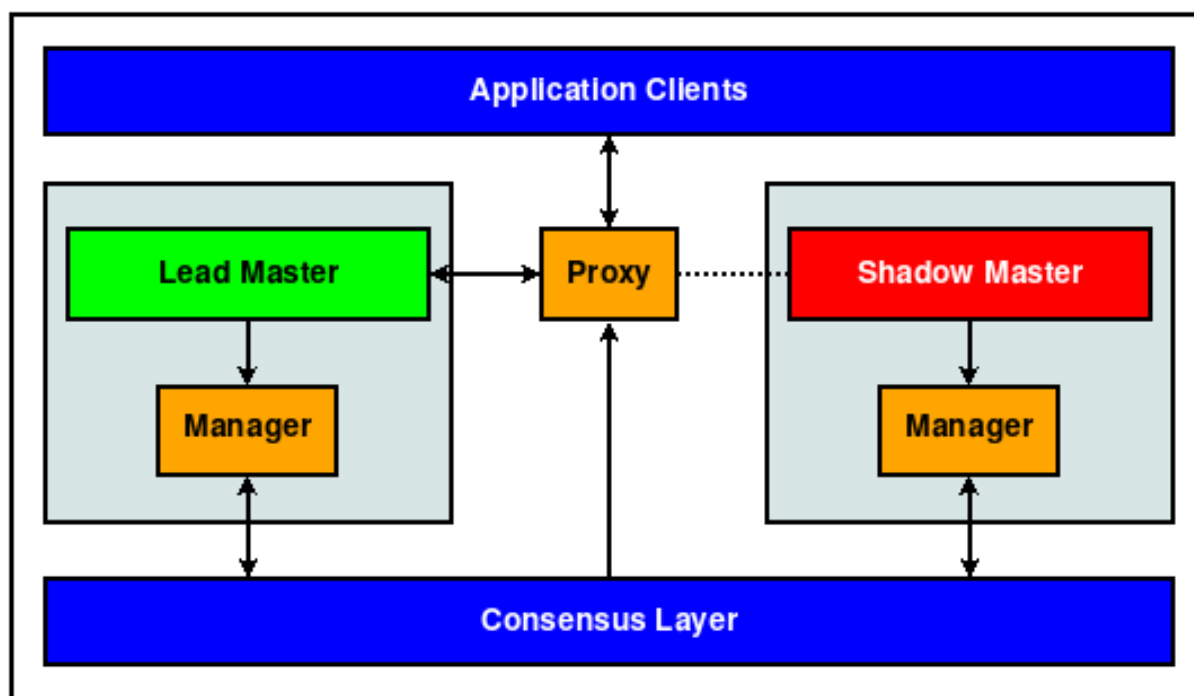


図 3: HARP Cluster

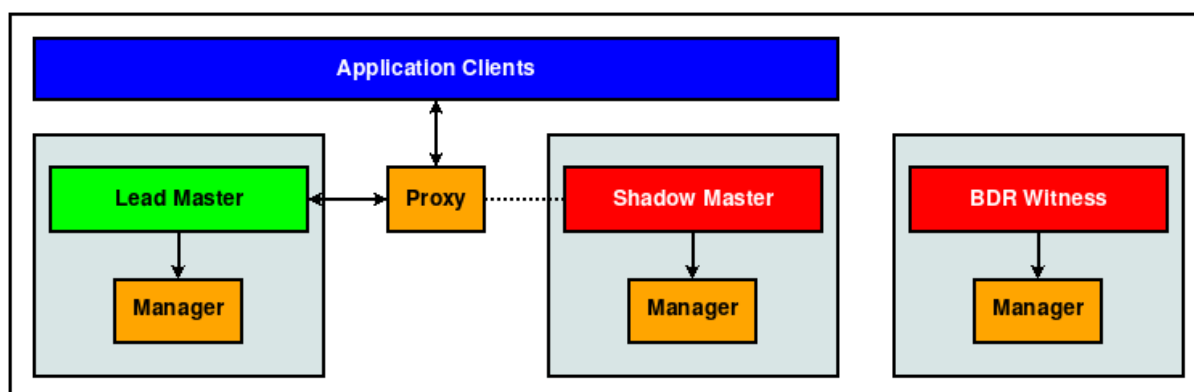


図 4: HARP Cluster w/BDR Consensus

2.4.1 リーダーの決定

例として、単一のデータセンターに存在する可能性がある、ローカルに細分化された BDR Always-On グループでのリードマスターのロールを検討します。Postgres または Manager リソースが起動され、構成可能な更新間隔の後、次が発生する必要があります。

1. Manager は、割り当てられた Postgres リソースのステータスを確認します。- Postgres が実行されていない場合は、構成可能なタイムアウト後に再試行します。- Postgres が実行されている場合は続行します。
2. Manager は、コンセンサスレイヤーのリーダーリースのステータスを確認します。- リースが要求されていない場合は、リースを取得し、このマネージャーに割り当てられた Postgres インスタンスの ID を割り当てます。このリース期間は構成可能ですが、設定が低すぎると、予期しないリーダーシップの移行が発生する可能性があります。- リースが既に当社によって要求されている場合は、リースの TTL を更新します。- それ以外の場合は何もしません。

もっと多くのことが起こりますが、この単純化されたバージョンは何が起きているのかを説明しています。リーダーリースは1つのノードでのみ保持でき、他の場所で保持されている場合、HARP マネージャーはあきらめて後で再試行します。

注釈: 選択されたコンセンサスレイヤーに応じて、リーダーリースのステータスを確認するために繰り返しループするのではなく、HARP は通知をサブスクライブします。この場合、リースの状態が変化するたびに、ポーリングではなくすぐに応答できます。現在、この機能は etcd コンセンサスレイヤーに制限されています。

これは、HARP 自分自身が選挙を行わず、コンセンサスレイヤーに委任されるクォーラムを管理しないことを意味します。コンセンサスレイヤーのクォーラムはリースを取得する行為を確認する必要があるため、要求が成功した場合、そのノードはその場所のクラスターをリードします。

2.4.2 接続ルーティング

リードマスターのロールが確立されると、接続は HARP プロキシに反映されると同様の決定論的な結果で処理されます。HARP プロキシが特定のバックエンドリソースの接続ターゲットを決定する必要がある場合を考えます。

1. HARP プロキシは、設定された場所にある現在のリードマスターのコンセンサスレイヤーを照会します。
2. これが設定されていないか移行中の場合：- Postgres への新しいクライアント接続は禁止されていますが、クライアントは蓄積され、リードマスターが表示されるまで一時停止状態になります。- 既存のクライアント接続は現在のトランザクションを完了することが許可され、新しい接続と同様の保留状態に戻ります。
3. クライアント接続はリードマスターに転送されます。

この場合に示される相互作用は、HARP Manager または Postgres との相互作用を必要としません。コンセンサスレイヤーは、プロキシの観点からのすべての真実のソースです。

2.4.3 コロケーション

ワークユニットの配置は、組織が次の原則に従う必要があります。

1. manager ユニットと Postgres ユニットが同一ノードに存在すること。
2. コンセンサスレイヤーの内容は、すべての運用作業単位の規範的な役割を指示します。

この取り決めは、クラスターフォーラムの責任をコンセンサスレイヤーに委任しますが、HARP は重要なロールの割り当てとキー/値のストレージにそれを活用します。コンセンサスレイヤーが動作不能または到達不能の場合、ストレージも取得も成功しないため、不正な Postgres ノードが接続を受け入れることを防ぎます。

その結果、コンセンサスレイヤーは一般に、安全性を最大限に高めるために HARP または HARP 管理対象ノードの外部に存在します。リファレンス図はこの分離を示していますが、必須ではありません。

注釈: クラスター状態を操作および管理するために、BDR には Raft コンセンサスモデルの独自の実装が含まれています。この同じレイヤーを利用するように HARP を構成して、外部依存関係への依存を減らし、サーバーリソースを保持できます。ただし、このアプローチの特定の欠点については、[コンセンサスレイヤー](#) で説明されています。

2.5 推奨されるアーキテクチャと使用

HARP は、主に 2 つ以上のデータセンターにあり、少なくとも 5 つの BDR ノードで構成される BDR Always-On アーキテクチャを表すように設計されました。この構成では、ロジカルスタンバイノードはカウントされません。

次の図は、現在および標準の表現を示しています。

この図では、HARP Manager は BDR ノード 1~4 に存在します。クラスターの初期状態は、BDR ノード 1 が DC A のリードマスターであり、BDR ノード 3 が DC B のリードマスターです。

この構成により、DC A の HARP プロキシリソースが BDR ノード 1 に接続され、DC B の HARP プロキシリソースが BDR ノード 3 に接続されます。

注釈: この図は DC ごとに 1 つの HARP プロキシのみを示していますが、これは例にすぎず、単一障害点とは見なされません。HARP プロキシノードはいくつでも存在でき、それらはすべてアプリケーショントラフィックを同じノードに転送します。

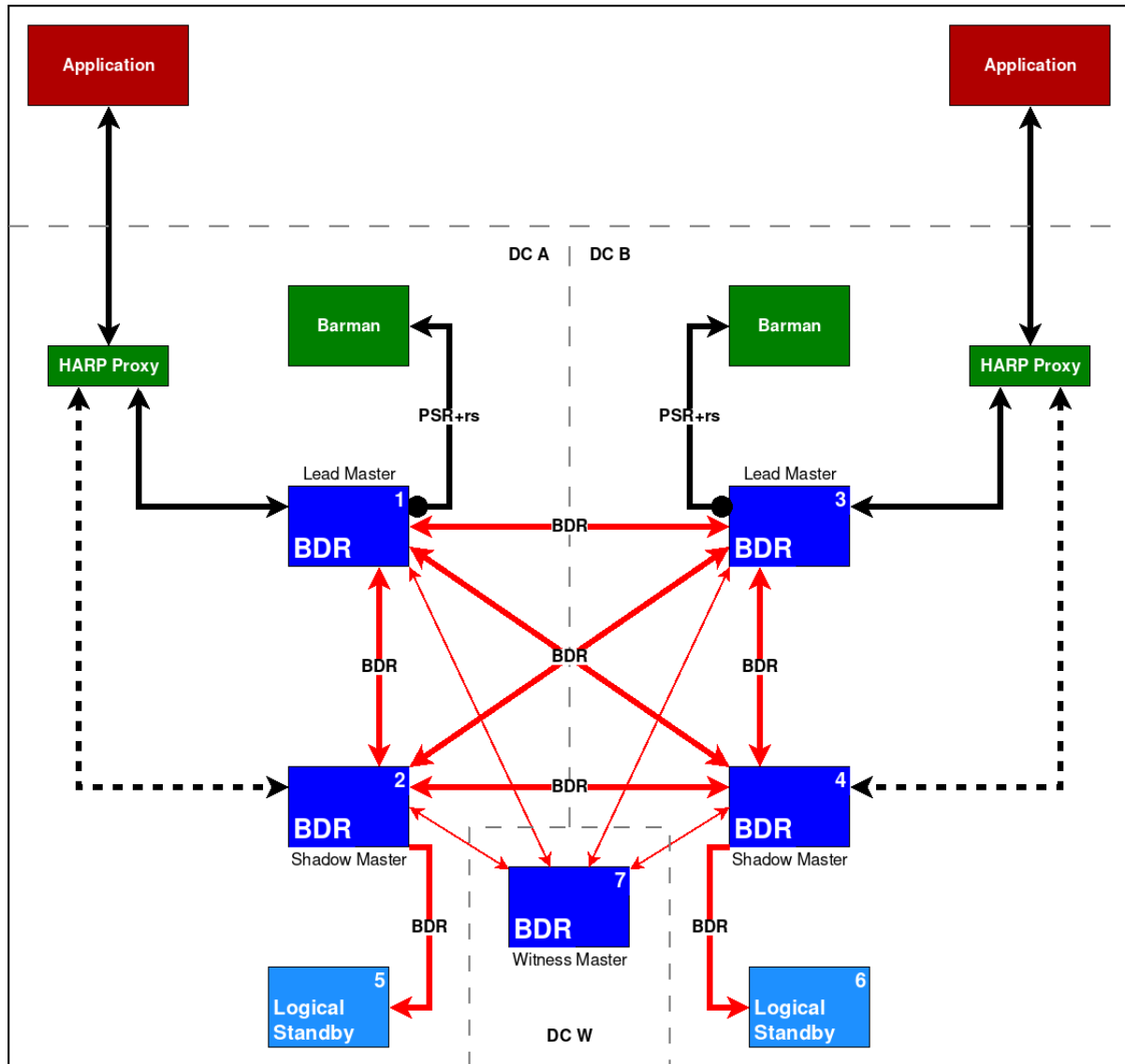


図 5: BDR Always-On Reference Architecture

2.5.1 ロケーション構成

複数の BDR ノードがロケーションでリードマスターロックを取得できるようにするには、`config.yml` 構成ファイルでロケーションを定義する必要があります。

図に示されている BDR Always-On リファレンスアーキテクチャを再現するには、BDR ノード 1 および 2 の `config.yml` 構成に次の行を含めます。

```
location: dca
```

BDR ノード 3 および 4 の場合、次を追加します。

```
location: dcb
```

これは、これらのそれぞれのデータセンターで指定された HARP プロキシノードにも適用されます。

2.5.2 BDR 3.7 との互換性

BDR 3.7 以降では、BDR ノードにロケーションを割り当てることにより、より直接的なロケーション定義を提供します。これは、BDR ノードに接続しているときに次の SQL API ファンクションを呼び出すことによって実行されます。したがって、BDR ノード 1 および 2 の場合、次のようにします。

```
SELECT bdr.set_node_location(dca);
```

BDR ノード 3 および 4 の場合：

```
SELECT bdr.set_node_location(dcb);
```

第3章 Installation

HARP の標準インストールには、2つのシステムサービスが含まれます。

- 管理対象ノードの HARP Manager (`harp-manager`)

*他の場所での HARP プロキシ (`harp-proxy`)

これらのサービスをインストールして構成し、適切なクォラムベースの接続ルーティングのために Postgres を管理するには2つの方法があります。

3.1 ソフトウェアバージョン

HARP は外部ソフトウェアに依存しています。これらは、ここにリストされている最小バージョンに適合する必要があります。

Software	Min version
etcd	3.4
PgBouncer	1.14

3.2 TPAExec

HARP をインストールして構成する最も簡単な方法は、クラスターの展開と管理に EDB TPAexec ユーティリティを使用することです。本ソフトウェアの詳細は、

[TPAexec product page](#) を参照してください。

注釈: TPAExec は現在、BDR クラスター展開専用の EULA でのみ入手できます。TPAExec URL にアクセスできない場合は、営業またはアカウント担当者にお問い合わせください。

TPA `config.yml` ファイルに次の属性が含まれていることを確認して、クラスタールーティングが HARP を介して管理されていることを認識するように TPAexec を構成します。

```
cluster_vars:
  failover_manager: harp
```

注釈: 21.1 より前の TPAexec のバージョンでは、少し異なるアプローチが必要です。

```
cluster_vars:
  enable_harp: true
```

この後、クラスターを変更するための `tpaexec` コマンドを呼び出して、HARP をインストールします。

```
tpaexec provision ${CLUSTER_DIR}
tpaexec deploy ${CLUSTER_DIR}
```

クラスター固有の考慮事項を除き、他の変更は必要ありません。

3.3 パッケージのインストール

現在、CentOS / RHEL パッケージは EDB パッケージインフラストラクチャによって提供されています。詳細は、[HARP product page](#) を参照してください。

3.3.1 etcd パッケージ

現在、多くの一般的な Linux ディストリビューションの `etcd` パッケージは、標準のパブリックリポジトリでは利用できません。したがって、EDB は RHEL および CentOS バージョン 7 および 8、Debian、および Ubuntu LTS などのバリエーション用に `etcd` をパッケージしました。これらのライブラリを使用するには、HARP パッケージリポジトリにアクセスする必要があります。

3.4 コンセンサスレイヤー

HARP の動作には分散コンセンサスレイヤーが必要です。現在、これは `bdr` または `etcd` のいずれかである必要があります。3 つ未満の BDR ノードを使用する場合、`etcd` に依存する必要がある場合があります。それ以外の場合、BDR サービスが停止すると、コンセンサスレイヤーが単一のノードに削減されるため、ノードのコンセンサスが妨げられ、Postgres ルーティングが無効になります。

3.4.1 etcd

コンセンサスレイヤーとして `etcd` を使用している場合、`etcd` は Postgres ノードに直接インストールするか、アクセスできる別の場所にインストールする必要があります。

`etcd` をコンセンサスレイヤーとして設定するには、このコードを `HARP config.yml` 構成ファイルに含めます。

```
dcs:
  driver: etcd
  endpoints:
    - host1:2379
```

(次のページに続く)

(前のページからの続き)

- host2:2379
- host3:2379

TPAExec を使用する場合、設定済みの etcd エンドポイントはすべてここに自動的に入力されます。

3.4.2 BDR

bdr ネイティブコンセンサスレイヤーは、BDR 3.6.21 および 3.7.3 から利用できます。このコンセンサスレイヤーモデルでは、BDR クラスターのルーティングを管理する際に補助ソフトウェアは必要ありません。

クラスターでクォーラムを使用できるようにするには、ノードのメンテナンスまたは停止中に BDR のコンセンサスレイヤーが応答し続けるように、常に 3 つ以上のノードを使用します。

BDR をコンセンサスレイヤーとして設定するには、これを `config.yml` 構成ファイルに含めます。

```
dcg:  
  driver: bdr  
  endpoints:  
    - host=host1 dbname=bdrdb user=harp_user  
    - host=host2 dbname=bdrdb user=harp_user  
    - host=host3 dbname=bdrdb user=harp_user
```

BDR コンセンサスレイヤーのエンドポイントは、標準の Postgres DSN 接続フォーマットに従います。

第4章 Configuring HARP for cluster management

HARP 構成ファイルは、読みやすいように簡素化された標準の YAML スタイルのフォーマットに従います。このファイルはデフォルトで/etc/harp ディレクトリにあり、`config.yml` という名前です

`-f/--config` 引数を使用して、すべての HARP 実行可能ファイルに構成ファイルの場所を明示的に提供できます。

4.1 標準構成

HARP は基本的に 3 つのコンポーネントとして動作します。

- HARP マネージャー
- HARP プロキシ
- `harptcl`

これらはそれぞれ、同じ標準の `config.yml` 構成フォーマットを使用します。これには、常に次のセクションが含まれます。

- `cluster.name` —すべてのオペレーションの対象となるクラスターの名前。
- `dc`s —すべてのエンドポイントの DCS ドライバーと接続構成。

これは、次のような HARP 操作には常に標準プリアンブルが含まれることを意味します。

```
cluster:
  name: mycluster

dc:
  ...
```

他のセクションは、オプションまたは名前付けた HARP コンポーネントに固有です。

4.1.1 クラスター名

`cluster` 見出しの下に `name` エントリは、HARP とのすべての相互作用に必要です。各 HARP クラスターには、特定のクラスターの DCS 内の明確化とデータのラベル付けの両方の名前があります。

HARP マネージャーは、HARP プロキシと `harpctl` が使用するクラスターに関する情報をここに書き込みます。HARP プロキシサービスは、このクラスター内のノードにトラフィックを誘導します。`harpctl` 管理ツールはこのクラスターと対話します。

4.1.2 DCS 設定

コンセンサスレイヤーの構成は、HARP 機能のキーとなります。DCS がなければ、HARP にはクラスターメタデータを保存する場所がなく、リーダーシップ選挙などを行うことができません。したがって、構成のこの部分は必須であり、特定の要素はオプションです。

次の複数の補助エントリを使用して、`dc`s という名前のセクションの下にすべての要素を指定します。

- `driver` : 使用するコンセンサスレイヤーの必須タイプ。現在は `etcd` または `bdr` です。コンセンサスレイヤーとしての `bdr` のサポートは実験的です。コンセンサスレイヤーとして `bdr` を使用すると、コンセンサスストレージ用の追加ソフトウェアが削減されますが、データベースのメンテナンス中にクォーラムを維持するために少なくとも 3 つの完全な BDR メンバーノードが必要です。
- `endpoints` : DCS に接続するために必要な接続文字列のリスト。可能であれば、ここに DCS のすべてのノードをリストします。これにより、DCS の大部分が引き続き動作し、ネットワークが到達できる限り、HARP が機能し続けます。

`etcd` をコンセンサスレイヤーとして使用する場合は次のとおりです。edb_notranlate_1 実験的な `bdr` コンセンサスレイヤーを使用する場合は次のとおりです。edb_notranlate_2 現在、`bdr` コンセンサスレイヤーは最初のエンドポイントがローカルの `postgres` インスタンスを指す必要があります。

- `request_timeout` : 要求が失敗したと見なされる時間（ミリ秒）。HARP は DCS に要求を行い、この時間内に応答を受信しない場合、操作は失敗したと見なされます。これにより、要求の性質に応じて、問題がエラーとしてログに記録されるか、再試行されます。デフォルト：250。

次の DCS SSL 設定は、edb_notranlate_3 yaml dcs:driver:etcd endpoints: - host1:2379 - host2:2379 - host3:2379 edb_notranlate_4 yaml クラスター: 名前: mycluster

dc: ドライバー: etcd エンドポイント: - host1:2379 - host2:2379 - host3:2379

manager: 名前: node1 log_level: INFO edb_notranlate_5 yaml cluster: 名前: mycluster

dc: ドライバー: etcd エンドポイント: - host1:2379 - host2:2379 - host3:2379

proxy: 名前: proxy1 場所: dc1 pgbouncer_bin_dir: /usr/sbin edb_notranlate_6 bash harpctl set node node1 lease_refresh_interval=100 edb_notranlate_7 bash harpctl set proxy global max_client_conn=1000 ``

第5章 Cluster bootstrapping

HARP を使用するには、最小限のメタデータが DCS に存在する必要があります。クラスターを「ブートストラップ」するプロセスとは、本質的に、ノード、場所、およびその他のランタイム構成を一度にまたはリソースごとに初期化することを意味します。

このプロセスは、`harpctl apply` コマンドによって制御されます。詳細については、[harpctl command-line tool](#) を参照してください。

DCS を設定し、ブートストラップの前に機能することを確認します。

重要: これらの例の一部またはすべてを単一の YAML ドキュメントに組み合わせて、一度に適用できます。

5.1 クラスター全体のブートストラップ

一部の設定はクラスター全体に適用され、ブートストラップ中に指定できます。現在、これは `event_sync_interval` ランタイム指示子にのみ適用されますが、他の人は後で追加される可能性があります。

フォーマットは次のとおりです。

```
cluster:
  name: mycluster
  event_sync_interval: 100
```

そのファイルの名前付けが `cluster.yml` であると仮定すると、次のように適用します。

```
harpctl apply cluster.yml
```

クラスター名が DCS でまだ定義されていない場合、これもその値を初期化します。

重要: ここで指定されたクラスター名パラメーターは、常に `config.yml` で指定されたクラスター名をオーバーライドします。ブートストラップファイルは、クラスターまたはその大規模な構成の一部をブートストラップするために必要なすべての要素を提供することが前提です。 `config.yml` ファイルは、主に HARP マネージャー、HARP プロキシ、または `harpctl` の実行を制御することを目的としています。

5.2 ロケーションブートストラップ

すべての HARP ノードは、最大 1 つの場所に関連付けられます。この場所は、単一のデータセンター、複数の基になるサーバーで構成されるグループ化されたリージョン、Amazon アベイラビリティゾーンなどです。これは、HARP がノードをグループ化して、その場所のノードを 1 つだけがリードマスターとして表すことができるようにする論理構造です。

したがって、1 つ以上の場所を初期化する必要があります。このフォーマットは次のとおりです。

```
cluster:
  name: mycluster

locations:
  - location: dc1
  - location: dc2
```

そのファイルの名前付けが `locations.yml` であると仮定すると、次のように適用します。

```
harpctl apply locations.yml
```

クラスターの操作を実行するときは、名前をプリアンブルとして含めて、変更が適切な場所に送られるようにします。

ロケーションがブートストラップされると、簡単な検査で表示されます。

```
> harpctl get locations

Cluster   Location Leader Previous Leader Target Leader Lease Renewals
- - - - -
mycluster dc1                                <nil>
mycluster dc2                                <nil>
```

どちらの場所も HARP によって認識され、ノードとプロキシの割り当てに使用できます。

5.3 ノードのブートストラップ

HARP ノードは名前付きクラスターに存在し、指定された名前が必要です。これを超えて、他のすべての設定は動的であり、HARP との相互作用に影響を与える可能性があるため、DCS に保持されます。このために、

ロケーション構成

で説明されている 1 つ以上のランタイム指示子を使用して各ノードをブートストラップします。

ノードのブートストラップ中に、いくつかの必須フィールドがあります。

- name
- location

- dsn
- pg_data_dir

その他はすべてオプションであり、クラスターに依存する場合があります。一度に複数のノードをブートストラップできるため、一般的にこのフォーマットは次の構造に適合します。

```
cluster:
  name: mycluster

nodes:
- name: node1
  location: dc1
  dsn: host=node1 dbname=bdrdb user=postgres
  pg_data_dir: /db/pgdata
  leader_lease_duration: 10
  priority: 500
```

そのファイルの名前付けが `node1.yml` であると仮定すると、次のように適用します。

```
harpctl apply node1.yml
```

ノードがブートストラップされると、簡単な検査で表示されます。

```
> harpctl get nodes
```

Cluster	Name	Location	Ready	Fenced	Allow Routing	Routing	Status	Role	Type
↪ Lock Duration									

↪-----									
mycluster	bdr1	dc1	true	false	true	ok		primary bdr	30

5.4 プロキシブートストラップ

ロケーションやノードとは異なり、プロキシは、ロケーション内のすべてのプロキシに適用される構成テンプレートを提供することもできます。これらは、`global` 指定の下で DCS に保存されます。各プロキシには名前がインスタンスとして存在する必要がありますが、一部の設定で特定のオーバーライドが必要な場合を除き、これ以上のカスタマイズは必要ありません。

これは、クラスターのデフォルトの構成設定が同じであるプロキシが複数存在する可能性があり、すべてのプロキシに対してこれらの値を繰り返す必要がないためです。

さらに、プロキシテンプレートをブートストラップするときは、接続の割り当て用に少なくとも 1 つのデータベースを定義します。これらの注意事項を念頭に置いて、このフォーマットは次のとおりです。

```
cluster:
  name: mycluster
```

(次のページに続く)

(前のページからの続き)

```
proxies:
  monitor_interval: 5
  default_pool_size: 20
  max_client_conn: 1000
  database_name: bdrdb
  instances:
    - name: proxy1
    - name: proxy2
      default_pool_size: 50
```

これにより、2つのプロキシ proxy1 と proxy2 の HARP が構成されます。 proxy2 のみにカスタム default_pool_size がありますが、それ以外の場合はグローバル設定を使用します。

そのファイルの名前付けが proxy.yml であると仮定すると、次のように適用します。

```
harpctl apply proxy.yml
```

プロキシテンプレートがブートストラップされると、簡単な検査で表示されます。

```
> harpctl get proxies
```

Cluster	Name	Pool Mode	Auth Type	Max Client Conn	Default Pool Size
mycluster	global	session	md5	1000	20
mycluster	proxy1	session	md5	1000	20
mycluster	proxy2	session	md5	1000	50

第6章 HARP Manager

HARP Manager は、ローカルの PostgreSQL / BDR ノードと対話し、その状態に関する情報をコンセンサスレイヤーに保存するデーモンです。Manager は、それぞれのロケーションのリーダステータスを現在保持しているノードを特定し、構成（ラグ、CAMO ラグなど）の制約を適用して、資格のないノードがリーダの検討対象とならないようにします。

クラスター内のすべての Postgres ノードには、HARP Manager が関連付けられている必要があります。他のノードは存在できますが、リードまたはシャドウマスターの役割、または HARP マネージャーを必要とするその他の機能として参加することはできません。

重要: HARP Manager は、データベースの起動および停止に使用されることを想定しています。HARP マネージャーを停止すると、データベースが停止します。HARP Manager を起動すると、データベースがまだ起動されていない場合は起動します。別の方法を使用してデータベースを停止すると、HARP マネージャーはデータベースを再起動します。

6.1 仕組み

起動時に、HARP Manager は `pg_ctl` を使用して Postgres をまだ実行していない場合は起動します。その後、`node.lease_refresh_interval` 設定で定義されたサーバーを定期的にチェックします。HARP Manager は、次を含む Postgres に関するさまざまなデータを収集します。

ノードの現在の LSN。

- Postgres が実行されており、接続を受け入れている場合。この特定のデータポイントは、定期的に更新する必要があるリースと見なされます。有効期限が切れると、HARP プロキシは既存のルーティングからノードを削除します。

すべてのアップストリーム BDR ピアノードの現在の適用 LSN 位置。

- CAMO が有効になっている場合： - CAMO パートナーの名前 - ピア CAMO の状態 (`is_ready`) - CAMO キューが受信して適用した LSN 位置

*ノードが BDR か通常の Postgres かなどのノードタイプ。

- ノードの現在のロール（読み取り/書き込み、物理ストリーミングレプリカ、ロジカルスタンバイなど）。
- BDR ノードの状態。限定された場合を除き `ACTIVE`。
- その他のメタデータ収集の BDR ノード ID。

※その他のトラッキング値です。

重要: HARP で BDR ノードに名前を付ける場合、BDR ノード名は *node.name* 構成属性で表されるノード名と一致する必要があります。これはブートストラッププロセスで発生します。

ここで収集されたデータは、他の HARP Manager プロセスで完全に利用でき、遅延、パートナーの準備状況、およびスイッチオーバーとフェイルオーバーの動作を指示するその他の基準を評価するために使用されます。

ノードのメタデータを更新した後、HARP マネージャーは、リードマスターリースがローカルノードによって既に保持されている場合は更新するか、リースが期限切れになっている場合はリースを取得しようとします。すべてのノードの現在の状態は他のすべてのノードに知られているため、以前のリードマスターであったノードが存在する場合は、自動的に優先順位が付けられます。そうでない場合、他のすべてのノードは LSN ラグ、ノードの優先度、およびその他の基準で自分自身をリストし、最も適格なノードがリードマスターリースを押収します。

この手順は、ノードが存在する定義されたすべての場所で発生します。したがって、ロケーション DC1 と DC2 には、それぞれにリードマスターノードがあり、両方に個別のリースと選択のプロセスがあります。

HARP Manager は、これらの Postgres ステータスチェック、リース更新、および選択を繰り返して、クラスターが HARP プロキシからの接続に対して常にリードマスターターゲットを持つようにします。

6.2 構成

HARP マネージャーは、`dcs`、`cluster`、および `manager` 構成スタanzas を想定しています。以下は機能的な例です。

```
cluster:
  name: mycluster

dcs:
  driver: etcd
  endpoints:
    - host1:2379
    - host2:2379
    - host3:2379

manager:
  name: node1
  postgres_bin_dir: /usr/lib/postgresql/13/bin
```

実行中のインスタンスに `SIGHUP` を発行するか、サービスレベルのリロードを呼び出すことにより、構成ファイル（デフォルト：`/etc/harp/config.yml`）に変更を適用できます。

詳細については、[ロケーション構成](#) を参照してください。

6.3 使用法

これは、HARP Manager の基本的な使用法です。

Usage of ./harp-manager:

```
-f string
    Optional path to config file (shorthand)
--config string
    Optional path to config file
```

`harp-manager` をフォークされたデーモンとして起動するための引数はありません。このソフトウェアは、`systemd` またはコンテナ内でトップレベルプロセスとして起動されるように設計されています。これは、`journald` または接続されたコンテナターミナルを介したキャプチャとアクセスのために、出力が `STDOUT` および `STDERR` に送られることも意味します。

6.4 Postgres の HARP Manager 制御の無効化と再有効化

Postgres の HARP Manager 制御を一時的に一時停止できます。これにより、デーモンの実行は継続されますが、クラスターの既存の動作に影響を与える可能性のある操作は実行されません。管理を再度有効にすると、操作が再開されます。

ノード管理を一時的に無効にする例は次のとおりです。

```
harpctl unmanage node node1
```

詳細については、[harpctl command-line tool](#) を参照してください。

HARP Manager によるノード管理はデフォルトで有効になっています。

第7章 HARP Proxy

HARP プロキシは、クライアントアプリケーションと Postgres 間の抽象化レイヤーとして機能するデーモンです。コンセンサスレイヤーとインターフェイスして、現在のリードマスターノードの ID を取得し、トラフィックをその場所に転送します。計画的なスイッチオーバーまたは計画外のフェールオーバー中に、DCS の指示に従って新しいリードマスターノードにリダイレクトします。

HARP プロキシは `pgbouncer` または `builtin` から選択できます。プロキシタイプを指定しない場合、デフォルトは `builtin` です。 `pgbouncer` を使用する場合、HARP プロキシは DCS と PgBouncer 間のインターフェイスレイヤーです。そのため、PgBouncer は前提条件であり、HARP プロキシがそのアクティビティを完全に管理するにはインストールする必要があります。

組み込みプロキシには、追加のソフトウェアは必要ありません。ビルトインを使用する場合、HARP プロキシはレベル 4 パススループロキシとして機能します。

7.1 ビルトインプロキシ：仕組み

起動時に、HARP プロキシは、プロキシインスタンスごとにブートストラップファイルで指定されたリスニングアドレスとリスニングポートで着信接続を監視します。すべてのアプリケーションクライアントトラフィックは、組み込みプロキシを介して、このプロキシが動作している場所の現在のリードマスターノードに渡されます。

リードマスターリースが設定されていない場合、HARP プロキシは、新しいリードマスターが確立されるまですべての接続トラフィックを切断します。これは、`harpctl promote` を使用して新しいリードマスターへの計画的な移行を呼び出す場合にも適用されます。切断は即時です。

7.1.1 構成

プロキシタイプを `builtin` に設定して、組み込みプロキシを選択します。組み込みプロキシに適用される他の唯一のオプションは、許可されるクライアント接続の最大数を指定します。`max_client_conn` がシステムが処理できるものよりも高い場合、プロキシがオンになっているシステムの能力内の設定に下げられます。

7.2 PgBouncer : 仕組み

注釈: Harp Proxy が提供するものよりも多くの構成可能性が必要な場合は、ビルトインプロキシを使用して `pgbouncer` がそれをポイントするように設定することをお勧めします。

HARP プロキシは、起動時に PgBouncer がまだ実行されていない場合に起動し、クライアント接続を一時停止します。その後、DCS に連絡してリードマスターの ID を判断し、これをデータベース接続のターゲットとして使用するように PgBouncer を構成し、接続アクティビティを再開します。すべてのアプリケーションクライアントトラフィックは、PgBouncer を介して、このプロキシが動作している場所の現在のリードマスターノードに渡されます。

PgBouncer の実行中、HARP プロキシは DCS の `monitor_interval` 構成設定に基づいてそのステータスを確認し、監視のために DCS に保存します。この構成により、`harpctl` を使用した問い合わせで、構成されているすべてのプロキシまたは 1 つのプロキシのステータスを取得できます。

リードマスターリースが設定されていない場合、HARP プロキシは新しいリードマスターが確立されるまですべての接続トラフィックを一時停止します。これは、`harpctl promote` を使用して新しいリードマスターへの計画的な移行を呼び出す場合にも適用されます。これには PgBouncer PAUSE コマンドを使用するため、既存のセッションは保留中のトランザクションが停止する前に完了することができます。

7.2.1 PgBouncer 構成ファイル

HARP プロキシが接続管理とリダイレクトに PgBouncer を使用する場合、`pgbouncer.ini` ファイルが存在する必要があります。HARP マネージャーは、Proxy directives で定義されているさまざまなランタイム指示子に基づいてこのファイルをビルドします。

このファイルは、HARP プロキシが使用する `config.yml` と同じフォルダーにあります。HARP プロキシによって起動される PgBouncer プロセスはこの構成ファイルを使用し、デバッグまたは情報の目的で使用できます。この自動的に生成された `pgbouncer.ini` ファイルへの変更は、HARP プロキシが再起動するたびに失われるため、代わりに `harpctl set proxy` を使用します。`harpctl set proxy` を呼び出しても、プロキシが再起動するまで `pgbouncer.ini` ファイルは更新されません。

7.2.2 HARP プロキシノード管理の無効化と再有効化

PgBouncer の HARP プロキシ制御を一時的に一時停止できます。これにより、デーモンの実行は継続されますが、クラスターの既存の動作に影響を与える可能性のある操作は実行されません。管理を再度有効にすると、操作が再開されます。

特定のプロキシの管理を一時的に無効にする例は次のとおりです。

```
harpctl unmanage proxy proxy1
```

詳細については、[harpctl command-line tool](#) を参照してください。

プロキシノード管理はデフォルトで有効になっています。

7.2.3 パススルーユーザー認証

pgbouncer では、`auth_user` および `auth_query` ランタイム指示子を使用するように HARP プロキシを構成することを強くお勧めします。これらが設定されていない場合、PgBouncer `userlist.txt` ファイルには、PgBouncer が Postgres に代わって認証する必要があるすべてのユーザーのユーザー名とパスワードのハッシュの組み合わせを含める必要があります。

pgbouncer ユーザーは使用しないでください。これは、基になる PgBouncer サービスを操作するための管理レベルのユーザーとして HARP プロキシによって使用されます。

7.2.4 構成

HARP プロキシは、`dcs`、`cluster`、および `proxy` 構成スタンザを想定しています。以下は機能的な例です。

```
cluster:
  name: mycluster

dcs:
  driver: etcd
  endpoints:
    - host1:2379
    - host2:2379
    - host3:2379

proxy:
  name: proxy1
```

`` 各プロキシは DCS に接続して、接続をリッスンするホストとポートを取得します。

使用法

これは、 HARP プロキシの基本的な使用法です。

```
```bash
Usage of ./harp-proxy:
 -f string
 Optional path to config file (shorthand)
 --config string
 Optional path to config file
```

`harp-proxy` をフォークされたデーモンとして起動するための引数はありません。このソフトウェアは、`systemd` またはコンテナ内でトップレベルプロセスとして起動されるように設計されています。これは、`journald` または接続されたコンテナターミナルを介したキャプチャとアクセスのために、出力が `STDOUT` および `STDERR` に送られることも意味します。



## 第8章 harpctl command-line tool

harpctl は、コンセンサスレイヤーのコンテンツを直接操作して、目的のクラスタージオメトリに合わせるためのコマンドラインツールです。これを使用して、ノードのステータスを調べたり、ノードをリードマスターに「昇格」させたり、クラスター管理を無効/有効にしたり、クラスター設定をブートストラップしたりすることができます。

### 8.1 概要

```
$ harpctl --help

Usage:
 harpctl [command]

Available Commands:
 apply Command to set up a cluster
 completion generate the autocompletion script for the specified shell
 fence Fence specified node
 get Command to get resources
 help Help about any command
 manage Manage Cluster
 promote Promotes the sepcified node to be primary,
 set Command to set resources.
 unfence unfence specified node
 unmanage Unmanage Cluster
 version Command to get version information

Flags:
 -c, --cluster string name of cluster.
 -f, --config-file string config file (default is /etc/harp/config.yml)
 --dcs stringArray Address of dcs endpoints. ie: 127.0.0.1:2379
 -h, --help help for harpctl
 -o, --output string json, yaml.
 -t, --toggle Help message for toggle

Use "harpctl [command] --help" for more information about a command.
```

この基本的な概要に加えて、使用可能な各コマンドには、独自の一連の使用可能なサブコマンドとフラグがあります。

## 8.2 構成

harpctl は、動作するためにコンセンサスレイヤーと対話する必要があります。これは、正常に実行するには、`config.yml` に特定の最小量の設定を定義する必要があることを意味します。これには以下が含まれます。

- `dc.driver`
- `dc.endpoints`
- `cluster.name`

etcd を使用した例：

```
cluster:
 name: mycluster
dc:
 driver: etcd
 endpoints:
 - host1:2379
 - host2:2379
 - host3:2379
```

詳細は [ロケーション構成](#) を参照してください。

## 8.3 実行

次のように harpctl を実行します。

```
harpctl command [flags]
```

各コマンドには、独自のシリーズのサブコマンドとフラグがあります。次のコマンドを実行すると、これらのヘルプを利用できます。

```
harpctl <command> --help
```

## 8.4 harpctl apply

`apply` コマンドを使用して、目的のクラスターのさまざまな属性を定義するファイルを使用して HARP クラスターを「ブートストラップ」する必要があります。

次のような `apply` コマンドを実行します。

```
harpctl apply <filename>
```

これにより、基本的に、初期クラスターメタデータ、デフォルトまたはカスタムの管理設定などがすべて作成されます。これは、DCS がクラスターを管理するための究極のソースとして使用され、これらの設定を動的に変更できるためです。

これは、クラスター全体をブートストラップするために 1 回、オブジェクトのタイプごとに 1 回、または簡単にするためにノードごとに実行できます。

これは、単一ノードのブートストラップファイルの例です。

```
cluster:
 name: mycluster
nodes:
 - name: firstnode
 dsn: host=host1 dbname=bdrdb user=harp_user
 location: dc1
 priority: 100
 db_data_dir: /db/pgdata
```

ここで見られるように、複数の環境が同じ環境で動作している場合、すべての変更が正しい HARP クラスターをターゲットと保証に、常にクラスター名前のプリアンブルを含めることをお勧めします。

apply がエラーなしで完了すると、ノードはクラスターの残りの部分と統合されます。

**注釈:** 定義されたすべてのセクションが同時に適用されるため、このコマンドを使用してクラスター全体を一度にブートストラップすることもできます。ただし、初期定義中にクラスターの各部分を検証するのが難しくなるため、テスト以外でのこの使用はお勧めしません。

## 8.5 harpctl fence

ローカルまたは指定されたノードをフェンス済みとしてマークします。このステータスのノードは、基本的にクラスターから完全に除外されます。HARP プロキシはトラフィックを送信せず、その代表的な HARP マネージャーはリードマスターリースを要求せず、さらなる手順も実行されます。実行されている場合、HARP Manager はノードで Postgres も停止します。

次のような fence コマンドを実行します。

```
harpctl fence (<node-name>)
```

node-name はオプションです。省略された場合、harpctl はローカルに構成されたノードの名前を使用します。

## 8.6 harpctl get

クラスターのさまざまな要素のコンセンサスレイヤーに保存されている情報を取得します。これには、ノード、場所、クラスターなどが含まれます。完全なリストには次のものが含まれます。

- **cluster** — クラスターの状態を返します。
- **leader** — 現在または指定されたロケーションリーダーを返します。
- **location** — 現在または指定された場所情報を返します。
- **locations** — すべてのロケーションのリストを返します。
- **node** — 指定された Postgres ノードを返します。
- **nodes** — すべての Postgres ノードのリストを返します。
- **proxy** — 現在のプロキシ情報または指定されたプロキシ情報を返します。
- **proxies** — すべてのプロキシノードのリストを返します。

### 8.6.1 harpctl get cluster

現在のクラスターのコンセンサスレイヤーに保存されている情報を取得します。

```
> harpctl get cluster
```

```
Name Enabled
- - - - -
mycluster true
```

### 8.6.2 harpctl get leader

指定された場所の DCS に保存されている現在のリードマスターのノード情報を取得します。ロケーションが渡されない場合、harpctl は、実行された現在のノードのロケーションに基づいてそれを導き出そうとします。

例：

```
> harpctl get leader dc1
```

```
Cluster Name Ready Role Type Location Fenced Lock Duration
- - - - -
mycluster mynode true primary bdr dc1 false 30
```

### 8.6.3 harpctl get location

指定された場所の場所情報を取得します。ロケーションが渡されない場合、harpctl は、実行された現在のノードのロケーションに基づいてそれを導き出そうとします。

例：

```
> harpctl get location dc1
```

Cluster	Location	Leader	Previous Leader	Target Leader	Lease	Renewals
mycluster	dc1	mynode	mynode			<nil>

### 8.6.4 harpctl get locations

DCS に現在存在するすべてのロケーションの情報を取得します。

例：

```
> harpctl get locations
```

Cluster	Location	Leader	Previous Leader	Target Leader	Lease	Renewals
mycluster	dc1	mynode	mynode			<nil>
mycluster	dc2	thatnode	thatnode			<nil>

### 8.6.5 harpctl get node

指定されたノードの DCS に格納されているノード情報を取得します。

例：

```
> harpctl get node mynode
```

Cluster	Name	Location	Ready	Fenced	Allow Routing	Routing	Status	Role	Type
mycluster	mynode	dc1	true	false	true	ok		primary	bdr 30

### 8.6.6 harpctl get nodes

クラスター内のすべてのノードの DCS に格納されているノード情報を取得します。

例：

```
> harpctl get nodes
```

Cluster	Name	Location	Ready	Fenced	Allow Routing	Routing	Status	Role	Type	Lock
myclusters	bdra1	dc1	true	false	true	ok		primary	bdr	30
myclusters	bdra2	dc1	true	false	false	N/A		primary	bdr	30
myclusters	bdra3	dc1	true	false	false	N/A		primary	bdr	30

### 8.6.7 harpctl get proxy

指定されたプロキシの DCS に格納されているプロキシ情報を取得します。このクラスターのプロキシのデフォルトを表示するには、`global` を指定します。

例：

```
> harpctl get proxy proxy1
```

Cluster	Name	Pool Mode	Auth Type	Max Client Conn	Default Pool Size
mycluster	proxy1	session	md5	1000	20

### 8.6.8 harpctl get proxies

クラスター内のすべてのプロキシの DCS に保存されているプロキシ情報を取得します。さらに、デフォルトのプロキシ設定の `global` 擬似プロキシをリストします。

例：

```
> harpctl get proxies
```

Cluster	Name	Pool Mode	Auth Type	Max Client Conn	Default Pool Size
mycluster	global	session	md5	500	25
mycluster	proxy1	session	md5	1000	20
mycluster	proxy2	session	md5	1500	30

## 8.7 harpctl manage

クラスターが管理状態にない場合、Postgres のモニタリングとコンセンサスレイヤーの更新を再開するようにすべての HARP Manager サービスに指示します。これは、HARP ソフトウェアの更新またはクラスター全体に影響を与える可能性のあるその他の重要な変更が続いて、メンテナンスが完了した後に実行します。

次のような `manage` コマンドを実行します。

```
harpctl manage cluster
```

**注釈:** 現在、クラスター管理は `cluster` レベルでのみ有効または無効にできます。後のバージョンでは、個々のノードまたはプロキシに対してこれを行うこともできます。

## 8.8 harpctl promote

リーダーシップ要件を満たす次の利用可能なノードを現在の場所のリードマスターに昇格させます。これは要求されたイベントであるため、次の場合にスムーズなハンドオーバーを呼び出します。

1. 既存のリードマスターがリードマスターリースを解放する。-古いリードマスターと昇格したノード間のレプリケーションラグは `node.maximum_lag` 未満です。
2. 昇格したノードは、リードマスターリースを取得する唯一の有効な候補であり、現在の所有者によって解放されるとすぐに取得します。他のすべてのノードは、設定されていないリードマスターリースを無視します。-CAMO が有効になっている場合、昇格したノードは、リードマスターリースを保持していても、CAMO キューが完全に適用されるまでクライアントトラフィックを一時的に無効にします。
3. HARP プロキシは、pgbouncer を使用している場合、進行中のトランザクションを完了できるように PAUSE 接続します。昇格したノードがリードマスターリースを要求すると、新しい接続ターゲットの PgBouncer を再構成し、データベーストラフィックを再開します。HARP プロキシがビルトインプロキシを使用している場合、クライアントから新しい接続が要求されると、既存の接続を終了し、リードマスターへの新しい接続を作成します。

次のような `promote` コマンドを実行します。

```
harpctl promote (<node-name>)
```

リードマスターになるための基準を満たさない場合でも、ノードを強制的にリードマスターに設定する `--force` オプションを提供します。これにより、CAMO ステータスまたはレプリケーションラグの検証が回避され、昇格したノードに即時に移行されます。これは、昇格するノードを正確に指定する唯一の方法です。

これを成功させるには、ノードがオンラインで動作している必要があります。このオプションは注意して使用してください。

## 8.9 harpctl set

クラスター内の特定の属性を指定された値に設定します。これは、`apply` を使用するのではなく、特定のノード、プロキシ、場所、またはクラスターの構成設定を調整するために使用されます。これは、次のオブジェクトタイプに使用できます。

- `cluster` — クラスター関連の属性を設定します。
- `location` — 特定のロケーション属性を設定します。
- `node` — 特定のノード属性を設定します。
- `proxy` — 特定のプロキシ属性を設定します。

### 8.9.1 harpctl set cluster

クラスター関連の属性のみを設定します。

例：

```
harpctl set cluster event_sync_interval=200
```

### 8.9.2 harpctl set node

指定されたノードのノード関連の属性を設定します。ここでは、Node directives に記載されているオプションが有効です。

例：

```
harpctl set node mynode priority=500
```

### 8.9.3 harpctl set proxy

名前付けたプロキシのプロキシ関連の属性を設定します。

Proxy directives に記載されているオプションはここで有効です。この方法で設定されたプロパティでは、新しい値を有効にする前にプロキシを再起動する必要があります。

例：

```
harpctl set proxy proxy1 max_client_conn=750
```

クラスター全体のプロキシのデフォルトには `global` を使用します。

```
harpctl set proxy global default_pool_size=10
```

## 8.10 harpctl unfence

ローカルまたは指定されたノードから **fenced** 属性を削除します。これにより、以前に適用されたすべてのクラスター除外がノードから削除され、再びトラフィックを受信したり、リードマスターリースを保持したりできます。Postgres も実行されていない場合は起動されます。

次のような **unfence** コマンドを実行します。

```
harpctl unfence (<node-name>)
```

**node-name** はオプションです。省略すると、**harpctl** はローカルに構成されたノードの名前を使用します。

## 8.11 harpctl unmanage

クラスター内のすべての HARP Manager サービスを実行したままにしますが、Postgres をアクティブに監視したり、コンセンサスレイヤーのコンテンツを変更したりしません。これは、ノードの停止などの通常のフェイルオーバーイベントが原因でリーダーシップの移行が行われないことを意味します。これは、HARP ソフトウェアに変更を加える前の、またはクラスターにその他の重要な変更を加える前のシステムまたは HARP のメンテナンスを対象としています。

次のような **unmanage** コマンドを実行します。

```
harpctl unmanage cluster
```

---

**注釈:** 現在、クラスター管理は `cluster` レベルでのみ有効または無効にできます。後のバージョンでは、個々のノードまたはプロキシに対してこれを行うこともできます。

---



## 第9章 Consensus layer considerations

HARP は、分散制御システム (DCS) としても知られるコンセンサスレイヤーのさまざまな実装で動作するように設計されています。

現在、次の DCS 実装がサポートされています。

- etcd - BDR

この情報は、サポートされている DCS 実装と HARP の相互作用に固有です。

### 9.1 BDR ドライバーの互換性

bdr ネイティブコンセンサスレイヤーは、BDR バージョン 3.6.21 および 3.7.3 から利用できます。

投票クォーラムを維持するために、BDR 論理スタンバイノードは BDR クラスターでのコンセンサス通信に参加しません。DCS クォーラム要件を満たすために、これらを合計ノードリストでカウントしないでください。

### 9.2 クォーラムの維持

どのアーキテクチャのクラスターでも、投票クォーラムを介してコンセンサスを維持するには、少なくとも  $n/2 + 1$  個のノードが必要です。したがって、3 ノードのクラスターは単一ノードの停止を許容でき、5 ノードのクラスターは 2 ノードの停止を許容できます。コンセンサスが失われると、DCS が特定の場所のリードマスターであるノードを決定的に識別できないため、HARP は動作不能になります。

その結果、どちらの DCS を選択しても、ノードの半分以上が常に\_クラスター全体で\_使用可能である必要があります。これは、DCS ノードを 2 つ以上のデータセンターに分散する場合、重要な要素になる可能性があります。ネットワークパーティションは、投票過半数を維持できない場所でクォーラムを防止するため、HARP は動作を停止します。

したがって、コンセンサスレイヤーを構築するときは、奇数のノード（最小 3 つ）が重要です。理想的なケースは、ノードを少なくとも 3 つの独立した場所に分散して、単一のネットワークパーティションがコンセンサスを混乱させないようにすることです。

1 つの構成例は、2 つのデータセンターの 2 つの DCS ノードをプライマリ BDR ノードと指定し、5 番目の DCS ノード（BDR など）を指定することです。このような設計を使用すると、2 つの BDR データセンター間のネットワークパーティションは、独立して配置されたノードのおかげでコンセンサスを混乱させません。

## 9.2.1 マルチコンセンサスバリエント

HARP は、設定されたロケーションごとに1つのリードマスターを想定しています。通常、各場所は `location` 構成設定を使用して HARP で指定されます。ロケーションごとに個別の DCS クラスターを作成することにより、HARP とは無関係にこの動作をエミュレートできます。

これを行うには、目的の場所ごとに異なる DCS 接続ターゲットを使用するように `config.yml` で HARP を構成します。

DC-A の HARP ノードは次のようなものを使用します。

```
location: dca
dcs:
 driver: etcd
 endpoints:
 - dcs-a1:2379
 - dcs-a2:2379
 - dcs-a3:2379
```

DC-B は、正規の場所にあるノードに対応する異なるホスト名を使用しますが：

```
location: dcb
dcs:
 driver: etcd
 endpoints:
 - dcs-a1:2379
 - dcs-a2:2379
 - dcs-a3:2379
```

このデザインには異なるデータセンター間の DCS 通信がないため、それらの間のネットワークパーティションは HARP オペレーションに影響しません。この結果、HARP は他の場所のノードを完全に認識せず、各場所は本質的に別個の HARP クラスターとして動作します。

BDR は DCS として使用する場合、BDR がすべての参加者ノードにわたってコンセンサスレイヤーを維持するため、これは不可能です。

このアプローチの潜在的な欠点は、`harpctl` が現在の場所の外部のノードと対話できないことです。ノード情報の取得、リードマスターの取得または設定、または他の場所をターゲットとするその他の操作を実行することはできません。基本的に、この組織は `harpctl` への `---location` パラメーターを使用不可にします。

## 9.2.2 TPAexec とコンセンサス

これらの考慮事項も TPAexec に統合されています。etcd を使用してクラスターをデプロイすると、ロケーションごとに個別の DCS クラスターが構築され、厳密な一貫性を維持しながら高可用性が促進されます。

したがって、この構成例では、**first** の場所に割り当てられた DCS ノードをグループ化し、**second** の場所は別のクラスターです。

```
cluster_vars:
 failover_manager: harp
 harp_consensus_protocol: etcd

locations:
 - Name: first
 - Name: second
```

この動作をオーバーライドするには、特定のグループ化を強制するように **harp\_location** を暗黙的に構成します。

したがって、この例では、すべての etcd ノードを単一の凝集 DCS レイヤーに返します。

```
cluster_vars:
 failover_manager: harp
 harp_consensus_protocol: etcd

locations:
 - Name: first
 - Name: second
 - Name: all_dcs

instance_defaults:
 vars:
 harp_location: all_dcs
```

**harp\_location** オーバーライドは、従来のデータセンターよりもリージョンのアベイラビリティゾーンを優先する Amazon などのクラウドプロバイダーを使用する場合、特定のノードグループを優先するためにも必要になる場合があります。



## 第10章 Security and roles

基本的なパッケージのインストールと構成を超えて、HARP を動作させるには Postgres の権限が必要です。これらにより、コンセンサスレイヤーでノードステータスを維持するために必要に応じて Postgres または BDR に関する情報を収集できます。

### 10.1 Postgres のアクセス許可

次のいずれかの方法で、`node.dsn` パラメーターで指定されたロールを作成します。

- `CREATE USER ...`
- `CREATE ROLE ... WITH LOGIN`

この構文により、ロールはデータベースにログインして診断情報を収集できます。

同様に、このロールの `pg_hba.conf` にエントリが存在する必要があります。これにはさまざまな方法があります。例として、すべてのデータベースホストが `10.10.*` のどこかにある VPN サブネットを考えます。このような場合、最も簡単なアプローチは特定の行を追加することです。

#	TYPE	DATABASE	USER	ADDRESS	METHOD
hostssl		all	harp_user	10.10.1.1/16	scram-sha-256

---

**注釈:** この場合、現在非推奨の ``md5`` ではなく、より新しい ``scram-sha-256`` 認証を使用しています。 `hostssl` を指定して、SSL 認証を要求することも選択しました。

---

### 10.2 BDR のアクセス許可

BDR ノードには、特定のロールが HARP 対応ユーザに付与されている場合にのみ表示されるメタデータとビューがあります。この場合、HARP ユーザーには以下が必要です。

```
GRANT bdr_monitor TO harp_user;
```

HARP BDR ロールは、クラスターオペレーションに関する進行中の情報を維持するためのステータスマonitoring ツールを対象としています。

## 10.3 BDR コンセンサス権限

`dcs.driver` 構成パラメーターが `bdr` に設定されている場合、HARP は BDR をコンセンサスレイヤーとして使用します。そのため、現在 `bdr_superuser` ロールでのみ使用可能な API メソッドへのアクセスが必要です。これは、HARP 対応のユーザーが以下を必要とすることを意味します。

```
GRANT bdr_superuser TO foobar;
```

現在、BDR コンセンサスモデルにアクセスするには、スーパーユーザーと同等の権限が必要です。

---

**重要:** BDR スーパーユーザーは Postgres スーパーユーザーではありません。 `bdr_superuser` ロールには、制限された機能、テーブル、ビュー、その他のオブジェクトへのアクセスなど、BDR で昇格した権限が付与されるだけです。

---