Red Hat Directory Server 11

パフォーマンスチューニングガイド

Directory Server のパフォーマンスチューニング

Last Updated: 2022-03-19

Red Hat Directory Server 11 パフォーマンスチューニングガイド

Directory Server のパフォーマンスチューニング

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

このガイドでは、サーバーとデータベースのパフォーマンスを向上するためのヒントを提供します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、「Red Hat CTO である Chris Wright のメッセージ」をご覧ください。

第1章 DIRECTORYSERVER パフォーマンスチューニングの概要

この記事では、管理者が Red Hat Directory Server デプロイメントのパフォーマンスを最適化するために使用できる手順とオプションについて説明します。Directory Server インスタンスのパフォーマンスチューニングは、マシン環境、ディレクトリーサイズとデータタイプ、負荷とネットワークの使用、さらにはユーザーとクライアントが実行する操作のタイプがサーバーごとに異なるため、各サーバーで異なります。

このガイドの目的は、Red Hat Directory Server がサーバーとデータベースのパフォーマンスを追跡および評価するために提供する機能を強調することです。サーバーのパフォーマンスを調整するのに役立つ手順もあります。ただし、詳細なプランニング情報は、『For more in-depth planning information, however, check out the 『Red Hat Directory Server Deployment Guide』』、コマンドラインおよび UI ベースの管理手順については、『『Red Hat Directory Server Administration Guide』』を参照してください。

1.1. DIRECTORY SERVER パフォーマンスに関するゴールの設定

パフォーマンスチューニング は、サーバーの通常の動作環境における潜在的な（または実際の）ボトルネックを特定し、それらのボトルネックを軽減するための手順を実行するための簡単な方法です。

パフォーマンスチューニングの一般的な計画は以下のとおりです。

1. 環境を評価します。その使用法、負荷、ネットワーク接続と信頼性、最も一般的な操作、その上の物理マシン、およびそのリソースをめぐって競合するサービスなど、Directory Server の周りをすべて確認します。
2. 現在の Directory Server のパフォーマンスを測定し、ベースラインを確立します。
3. 改善できるサーバー領域を特定します。
4. Directory Server 設定に変更を加え、場合によってはホストマシンにも変更を加えます。
5. Directory Server のパフォーマンスを再度測定し、変更がどのようにパフォーマンスにどのように影響したかを確認します。

Directory Server は、次の 3 つの領域で何らかの監視を提供します。

- サーバープロセス（カウンターおよびログ）
- データベース (カウンター)
- データベースリンク（カウンター）

Directory Server では、ほとんどのパフォーマンス測定値は、Director yServer が情報を取得してクライアントに配信する度合いになります。これを考慮すると、これらは最高の Directory Server のパフォーマンスを実現するために調整できるサーバー領域です（これらをこの記事で取り上げます）。

- 検索操作
- インデックス作成のパフォーマンス（検索操作と書き込み操作の両方に影響します）
- データベーストランザクション
- データベースおよびエントリーキャッシュの設定
- データベースリンク

ホストマシンの設定またはハードウェアに他の変更を加えることができます。これは、Directory Server のパフォーマンスにも影響を与える可能性があります。

- 利用可能なメモリー（ディレクトリーのサイズに基づく）
- 同じマシン（リソースと競合する可能性がある）で実行されている他のサーバー
- 他のマシンにある他の Directory Server インスタンスへのユーザーデータベースの分散
- ネットワークパフォーマンスによるサーバー負荷の分散

これらの変更は、インスタンスに追加可能な変更よりも、効果的な Directory Server デプロイメントの計画に大きく関係します。『Deployment Guide』 で、最適なエンタープライズデプロイメントを計画する方法の詳細を確認できます。

第2章 サーバーおよびデータベースパフォーマンスの追跡

Red Hat Directory Server には、パフォーマンスデータを記録および追跡する、パフォーマンスカウンターとログの 2 つの方法があります。カウンターは、特にデータベースのパフォーマンスにおいて、Directory Server のパフォーマンスを判断するために使用されます。ログは、サーバーと LDAP の操作および設定に関する問題領域を診断するために使用されます。

パフォーマンスカウンターは、サーバーの Directory Server、設定されたすべてのデータベース、データベースリンク（データベースのチェーン）の操作および情報に集中します。

ログには、アクセス（クライアント接続用）、エラー（エラー、警告、およびイベントの詳細）、および監査（Directory Server 設定の変更）の 3 つのタイプがあります。アクセスログとエラーログはデフォルトで実行されます（サーバーの実行にはエラーログが必要です）。オーバーヘッドにより、監査ロギングは手動で有効にする必要があります。

注記

アクセスログはバッファーされます。これにより、負荷の高いサーバーであっても完全なアクセスロギングが可能になりますが、サーバーでイベントが発生してからログに書き込まれるまでに時間差が生じます。

2.1. サーバーアクティビティーの監視

Directory Server の現在のアクティビティーは、Web コンソールまたはコマンドラインから監視できます。また、すべてのデータベースのキャッシュアクティビティーを監視することもできます。

注記

サーバーによって監視される Directory Server データベース属性のカウンターによっては、32 ビットシステム上であっても、64 ビットの整数（合計接続、操作の開始、操作の完了、送信されたエントリー、および送信されたバイト数）が使用されます。大規模なシステムでは、これにより、カウンターのローリングが速すぎたり、監視データが歪んだりするのを防ぎます。

2.1.1. コマンドラインを使用した Directory Server の監視

コマンドラインを使用してサーバーを監視するには、以下を実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com monitor server
```

以下の表は、コマンドが返す属性について説明しています。

表2.1 サーバー監視属性

属性	説明
version	ディレクトリーの現在のバージョン番号を指定します。
threads	リクエストの処理に使用される現在アクティブなスレッドの数。追加のスレッドは、レプリケーションやチェーンなどの内部サーバータスクによって作成されることがあります。

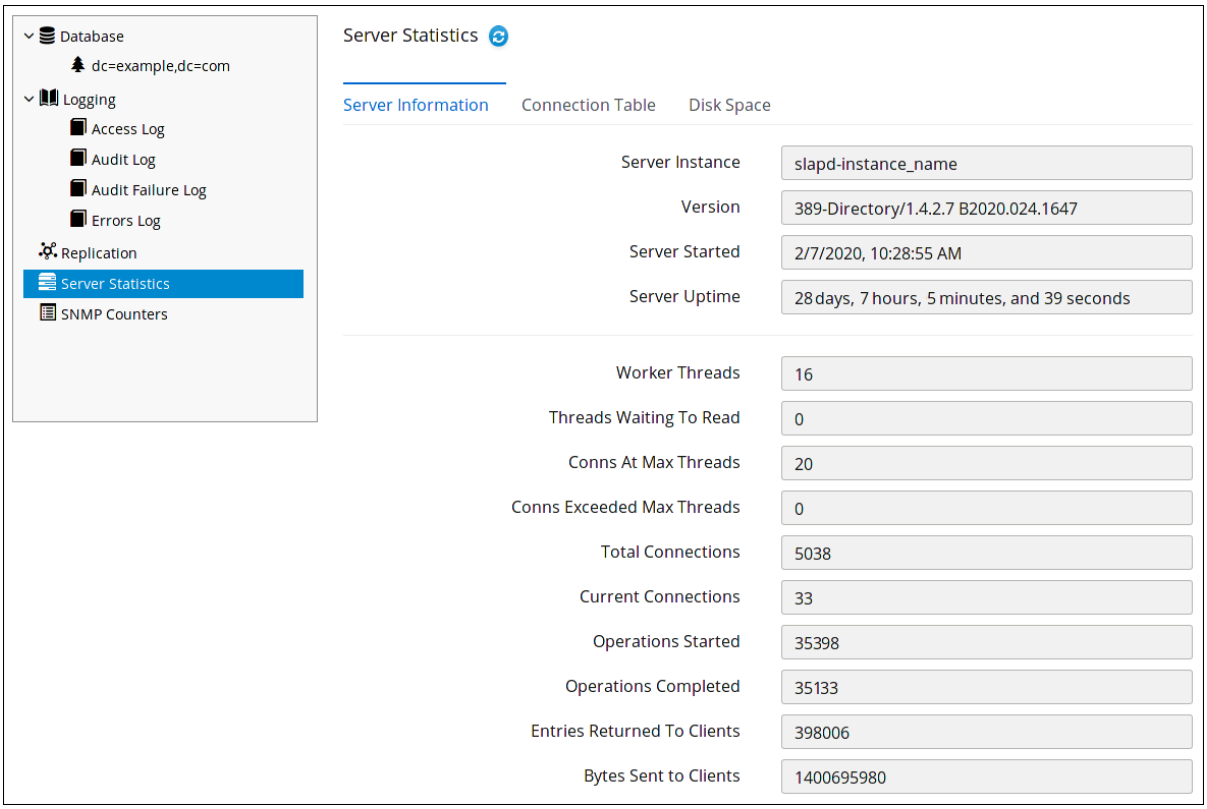
属性	説明
connection	開いている接続ごとに次の要約情報を提供します（ディレクトリーマネージャーとしてディレクトリーにバインドする場合にのみ使用できます）。 **fd** — この接続に使用するファイル記述子。 **opentime** — 接続が開かれた時間。 **opsinitiated** — この接続によって開始される操作の数。 **opscompleted** — 完了した操作の数 **BindDN** —ディレクトリーに接続するためにこの接続によって使用される識別名。 **rw** — 接続が読み取りまたは書き込みのためにブロックされると表示されるフィールド。 デフォルトでは、この情報は Directory Manager で利用できます。ただし、この情報に関連付けられている ACI を編集して、他のユーザーが情報にアクセスできるようにすることができます。
currentconnections	ディレクトリーによって現在サービス中の接続の数を識別します。
totalconnections	起動してからディレクトリーによって処理される接続の数を特定します。
currentconnectionsatmaxthreads	現在、**max thread** 状態にあるすべての接続を表示します。
maxthreadsperconnhits	接続が **max thread** に達した回数を表示します。
dtablesize	ディレクトリーで利用可能なファイル記述子の数を示します。接続ごとに 1 つのファイル記述子が必要で、オープンインデックスごとに 1 つ、ログファイル管理用に 1 つ、**ns-slapd** 自体に 1 つ必要です。基本的に、この値は、ディレクトリーが提供できる追加の同時接続の数を示します。ファイル記述子の詳細は、オペレーティングシステムのドキュメントを参照してください。
readwaiters	クライアントからデータの読み取りを待機するスレッドの数を特定します。
opsinitiated	起動後にサーバーが開始した操作の数を特定します。
opscompleted	起動後にサーバーが完了した操作の数を特定します。
entriessent	サーバー起動以降にクライアントに送信されたエントリーの数を特定します。

属性	説明
bytessent	サーバーが起動してからクライアントに送信されたバイト数を特定します。
currenttime	サーバーのこのスナップショットを作成した時間を指定します。時刻は、グリニッジ標準時（GMT）で UTC 形式で表示されます。
starttime	サーバーが起動した時刻を識別します。時刻は、グリニッジ標準時（GMT）で UTC 形式で表示されます。
nbackends	サーバーサービスのバックエンド（データベース）の数を識別します。

2.1.2. Web コンソールを使用したサーバーの監視

Web コンソールを使用してサーバーを監視するには、以下を行います。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Monitoring** タブで **Server Statistics** を選択します。

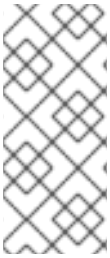


以下の表は、このメニューに表示されるフィールドについて説明しています。

表2.2 一般情報（サーバー）

フィールド	説明
Server Instance	Directory Server インスタンスの名前を表示します。
Version	現在のサーバーバージョンを識別します。
サーバー起動	サーバーが開始した日時。
Server Uptime	インスタンスが実行されている時間。
ワーカースレッド	リクエストの処理に使用される現在アクティブなスレッドの数。追加のスレッドは、レプリケーションやチェーンなどの内部サーバータスクによって作成されることがあります。
Threads Waiting To Read	クライアントからの読み取りを待つスレッドの合計数。サーバーがクライアントからのリクエストの受信を開始した後、何らかの理由でそのリクエストの送信が停止された場合、スレッドがすぐに読み取られないことがあります。一般に、読み取りを待機しているスレッドは、ネットワークまたはクライアントが遅いことを示しています。
Conns At Max Threads	現在、**max thread** 状態にあるすべての接続を表示します。
Conns Hit Max Threads	接続が **max thread** に達した回数を表示します。
Total Connections	この Directory Server インスタンスに確立された接続の総数。
現在の接続	オープン接続の合計数。各接続は複数の操作、したがって複数のスレッドで構成されます。
Operations Started	このコネクションによって開始される操作の数。
Operations Completed	この接続のためにサーバーが完了した操作の数。
Entries Returned to Clients	サーバー起動以降にクライアントに送信されたエントリーの数。
Bytes Sent to Clients	サーバーが起動してからクライアントに送信されたバイト数。

2.2. データベースアクティビティーの監視



注記

サーバーによって監視される Directory Server データベース属性のカウンターによっては、32 ビットシステム上であっても、64 ビットの整数（エントリーキャッシュヒット数、エントリーキャッシュ試行数、現在のカッシュサイズ、および最大キャッシュサイズ）が使用されます。大規模なシステムでは、これにより、カウンターのローリングが速すぎたり、監視データが歪んだりするのを防ぎます。

2.2.1. コマンドラインを使用したデータベースアクティビティーの監視

データベースの現在のアクティビティーを監視するには、以下を実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com monitor backend
```

以下の表は、コマンドが返す属性について説明しています。

表2.3 データベースモニタリングの属性

属性	説明
readonly	データベースが読み取り専用モード(**1**)であるか、読み取り/書き込みモード(**0**)であるかを示します。
entrycachehits	成功したエントリーキャッシュルックアップの合計数。この値は、データベースからリロードせずにサーバーがエントリーキャッシュからエントリーを取得できる合計回数です。
entrycachetries	インスタンスを起動した後のエントリーキャッシュルックアップの合計数。この値は、インスタンスが起動しているため、合計数です。{DS} はエントリーキャッシュからエントリーを取得しようとしました。
entrycachehitratio	エントリーキャッシュが正常に検索されるよう試行するエントリーキャッシュの数。この数は、最後にインスタンスを起動した後の合計ルックアップとヒットに基づいています。エントリーキャッシュヒット率は 100% に近いため、パフォーマンスが向上します。 操作がエントリーキャッシュに存在しないエントリーの検索を試みるたびに、サーバーはエントリーを取得するためにデータベースにアクセスする必要があります。したがって、この比率はゼロに切り替わり、ディスクアクセスの数が増大し、ディレクトリー検索のパフォーマンスが低下します。この比率を改善するには、データベースのエントリーキャッシュのサイズを増やします。 この比率を改善するには、cn=database_name, **cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-cachememsize*** 属性の値を増やすことでエントリーキャッシュのサイズを増やします。
currententrycachesize	エントリーキャッシュに現在存在するディレクトリーエントリーの合計サイズ（バイト単位）。 キャッシュに存在するエントリーのサイズを増やすには、cn=database_name, **cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-cachememsize*** 属性の値を増やします。

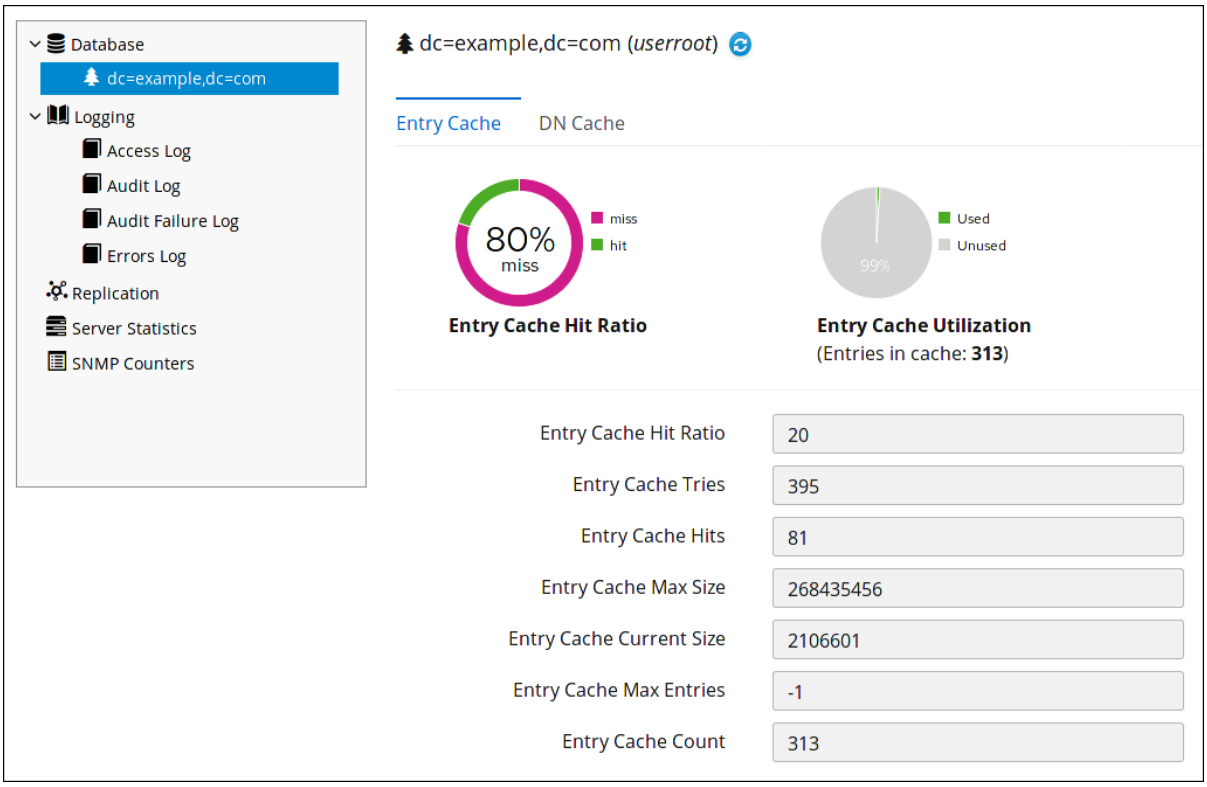
属性	説明
maxentrycachesize	{DS} がエントリーキャッシュで保持できるディレクトリーエントリーの最大サイズ（バイト単位）。 キャッシュに存在するエントリーのサイズを増やすには、cn=**database_name**, **cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-cachememsize*** 属性の値を増やします。
currententrycachecount	特定のバックエンドのエントリーキャッシュに保存されているエントリーの現在の数。
maxentrycachecount	データベースのエントリーキャッシュに保存されるエントリーの最大数。 この値をチューニングするには、cn=*database_**name**,cn=ldbm database,cn=plugins,cn=config* の ***nsslapd-cachesize*** 属性の値を増やします。
dncachehits	再度正規化するのではなく、DN キャッシュから正規化された識別名(DN)を取得して、サーバーが要求を処理できる回数。
dncachetries	インスタンスを起動した後の DN キャッシュアクセスの合計数。
dncachehitratio	キャッシュが正常な DN キャッシュヒットに対する比率。この値が 100% に近いほど、優れています。
currentdncachesize	DN キャッシュに現在存在する DN の合計サイズ（バイト単位）。 DN キャッシュに存在するエントリーのサイズを増やすには、cn=**database_name**, **cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-dncachememsize*** 属性の値を増やします。
maxdncachesize	{DS} が DN キャッシュで維持できる DN の最大サイズ（バイト単位）。 キャッシュに存在するエントリーのサイズを増やすには、cn=**database_name**, **cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-dncachememsize*** 属性の値を増やします。
currentdncachecount	DN キャッシュに現在存在する DN の数。
maxdncachecount	DN キャッシュで許可される DN の最大数。

2.2.2. Web コンソールを使用したデータベースアクティビティーの監視

Web コンソールを使用してデータベースアクティビティーを監視するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Monitoring** タブで、表示するデータベースエントリーを選択します。

4. **Entry Cache** を選択して、エントリーキャッシュのパフォーマンス値を表示します。



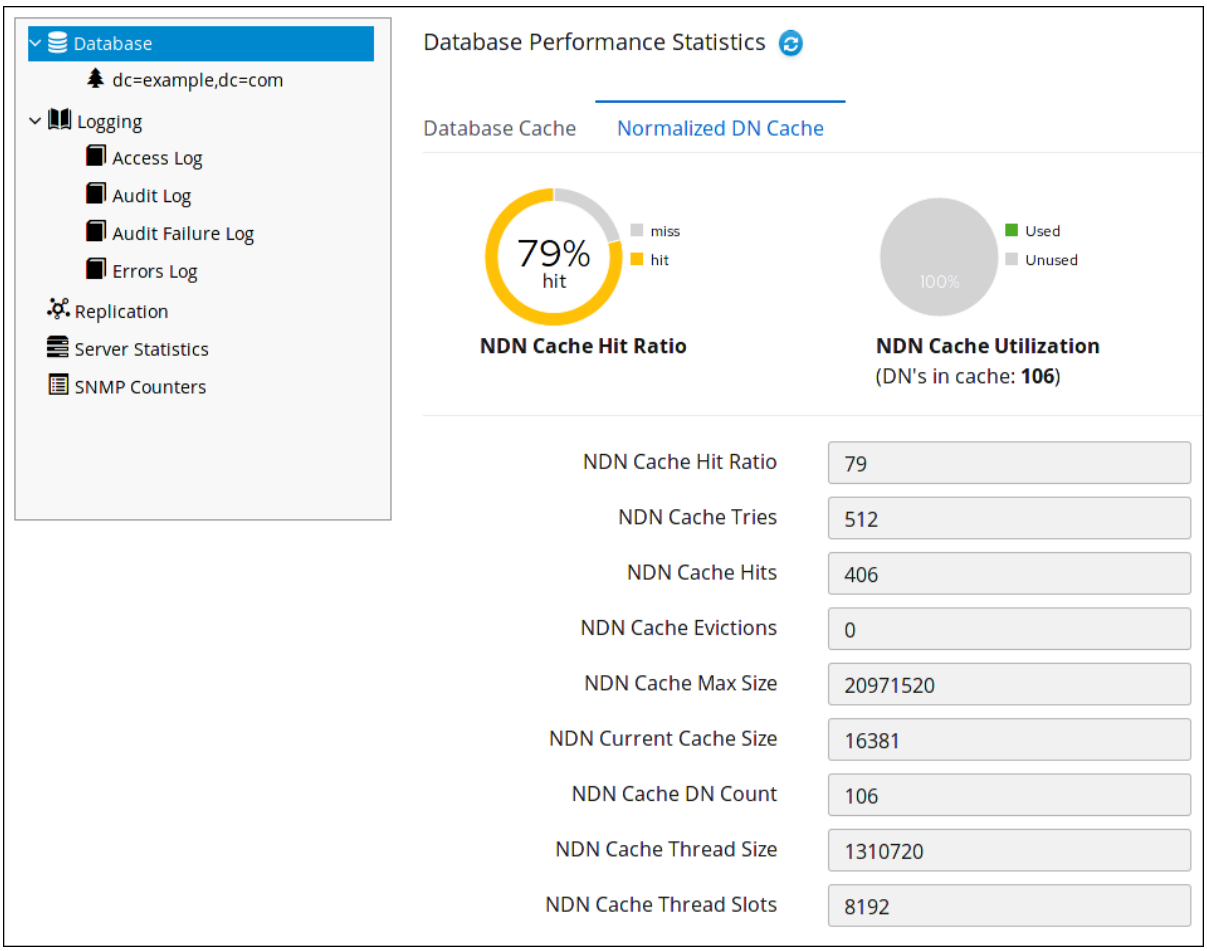
以下の表は、このタブに表示されるフィールドについて説明しています。

表2.4 エントリーキャッシュタブのフィールド

フィールド名	説明
Entry Cache Hit Ratio	エントリーキャッシュのルックアップを成功させるために試行するエントリーキャッシュの数を示す比率。この数は、ディレクトリーが最後に開始された後のルックアップおよびヒットの合計に基づいています。この値が 100% に近いほど、優れています。操作がエントリーキャッシュに存在しないエントリーの検索を試みるたびに、ディレクトリーがエントリーを取得するためにディスクアクセスを実行する必要があります。したがって、この比率がゼロに近づくと、ディスクアクセスの数が増え、ディレクトリ検索のパフォーマンスが低下します。 この比率を改善するには、データベースの **cn=**database_name, **cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-cachememsize*** 属性の値を増やして、エントリーキャッシュのサイズを増やします。
Entry Cache Tries	ディレクトリーが最後に開始されてからのエントリーキャッシュルックアップの合計数。つまり、サーバー起動以降に要求されたエントリーの合計数です。
Entry Cache Hits	成功したエントリーキャッシュルックアップの合計数。つまり、ディスクに移動するのではなくキャッシュからデータを取得することで、サーバーが検索要求を処理できる合計回数です。

フィールド名	説明
Entry Cache Max Size	ディレクトリーによって維持されるエントリーキャッシュのサイズ（バイト単位）。 この値は、データベースの **cn=database_name, cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-cachememsize*** 属性によって管理されます。
Entry Cache Current Size	エントリーキャッシュに現在存在するディレクトリーエントリーの数。
Entry Cache Max Entries	**非推奨。** エントリーキャッシュで保持できるディレクトリーエントリーの最大数。 許可される最大エントリー数を設定してキャッシュサイズの管理を試行しないでください。これにより、ホストが RAM を効果的に割り当てることが難しくなる可能性があります。***nsslapd-cachememsize*** 属性を使用して、キャッシュで利用可能な RAM の量を設定してキャッシュサイズを管理します。
Entry Cache Count	エントリーキャッシュに現在存在するディレクトリーエントリーの数。

5. **DN Cache** を、DN キャッシュのパフォーマンス値に対して選択します。



2.3. データベースリンクアクティビティーの監視

データベースリンク（チェーンされたデータベース）のアクティビティーも表示できますが、コマンドラインだけを使用します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com monitor chaining
```

以下の表は、コマンドが返す属性について説明しています。

表2.5 データベースリンク監視属性

属性名	説明
nsAddCount	受信した追加操作の数。
nsDeleteCount	受信した削除操作の数。
nsModifyCount	受信した変更操作の数。
nsRenameCount	受信した名前変更操作の数。
nsSearchBaseCount	受け取ったベースレベルの検索の数。
nsSearchOneLevelCount	受信した 1 レベル検索の数。
nsSearchSubtreeCount	受信したサブツリー検索の数。
nsAbandonCount	受信した破棄操作の数。
nsBindCount	受信したバインド要求の数。
nsUnbindCount	受信したバインド解除の数。
nsCompareCount	受信した比較操作の数。
nsOperationConnectionCount	通常操作のオープン接続の数。
nsBindConnectionCount	バインド操作のオープン接続の数。

2.4. シャットダウンのローカルディスクの監視

システムで利用可能なディスク領域が小さすぎると、Directory Server プロセスが終了します。結果として、データベースが破損したり、データが失われたりするリスクがあります。

この問題を回避するには、Directory Server を設定して、空きディスク領域を監視します。監視スレッドは、設定、トランザクションログ、データベースディレクトリーを含むファイルシステムの空き領域をチェックします。

残りの空きディスク容量によって、Directory Server の動作は異なります。

- 空きディスク領域が定義されたしきい値に達すると、Directory Server は以下を行います。
  - 詳細ログを無効にします。
  - アクセスログを無効にします。
  - アーカイブされたログファイルを削除します。

注記

Directory Server は、しきい値に達した場合でもエラーログの書き込みを続行します。

- 空きディスク領域が設定されたしきい値の半分未満の場合、Directory Server は定義された猶予期間内にシャットダウンします。
- 利用可能なディスク領域が 4 KB 未満の場合は、Directory Server はすぐにシャットダウンします。

ディスク領域が解放されると、Directory Server はシャットダウンプロセスを中止し、以前に無効にしたすべてのログ設定を再度有効にします。

2.4.1. コマンドラインを使用したローカルディスク監視の設定

コマンドラインを使用してローカルディスクの監視を設定するには、以下を実行します。

1. ディスクの監視機能を有効にし、しきい値および猶予期間を設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-disk-monitoring=on nsslapd-disk-monitoring-threshold=3000000000 nsslapd-disk-monitoring-grace-period=60
```

   このコマンドは、空きディスク領域のしきい値を 3 GB に設定し、猶予期間を 60 秒に設定します。

2. 必要に応じて、***nsslapd-disk-monitoring-logging-critical*** パラメーターを有効にして、Directory Server がアクセスロギングを無効にしたり、アーカイブされたログを削除したりしないように設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-disk-monitoring-logging-critical=on
```

3. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

2.4.2. Web コンソールを使用したローカルディスク監視の設定

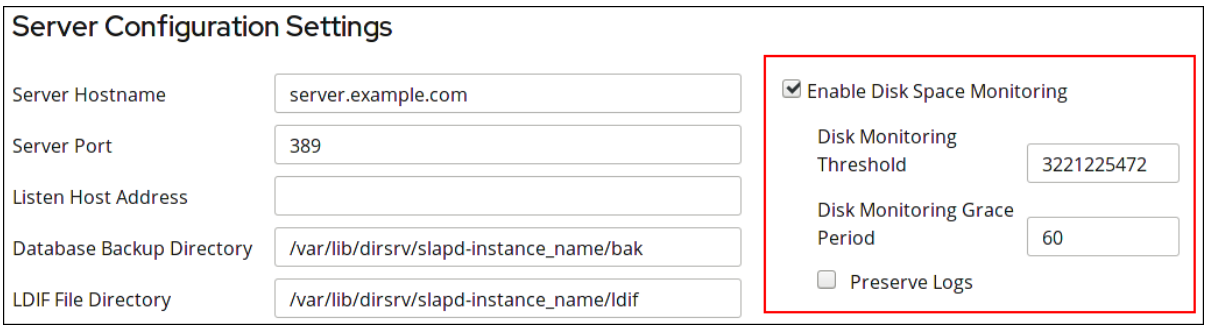
Web コンソールを使用してローカルディスクの監視を設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。

2. インスタンスを選択します。

3. **Server Settings** メニューを開き、**Server Configuration** を選択します。

4. **Enable Disk Space Monitoring** を有効にし、しきい値をバイト単位で設定し、猶予期間を分単位で設定します。



この例では、監視しきい値を 3 GB（3,221,225,472 バイト）に設定し、Directory Server がしきい値に達してからインスタンスをシャットダウンするまでの時間を **60** 分に設定します。

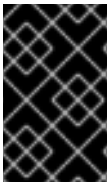
5. 必要に応じて **Preserve Logs** を選択して、Directory Server がアクセスロギングを無効にしたり、アーカイブされたログを削除したりしないように設定します。

6. **Save Configuration** をクリックします。

7. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

2.5. ロギングパフォーマンスの改善

大規模な Directory Server デプロイメントでは、大量のログコンテンツを作成できます。高負荷でのパフォーマンスを向上するには、アクセスログバッファーを無効にします。アクセスログのバッファーが無効になっていると、Directory Server はログエントリーをディスクに直接書き込みます。



重要

アクセスロギングは、サーバーの問題のデバッグや、クライアント接続および失敗した接続の試行の監視に非常に役立ちます。通常の操作環境では、アクセスログを無効にしないでください。

2.5.1. コマンドラインを使用したアクセスログバッファーの無効化

コマンドラインを使用してアクセスログバッファーを無効にするには、以下を実行します。

1. ***nsslapd-accesslog-logbuffering*** パラメーターを **off** に設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-accesslog-logbuffering=off
```

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

2.5.2. Web コンソールを使用したアクセスログバッファーの無効化

Web コンソールを使用してアクセスログバッファーを無効にするには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Server Settings → Logging → Access Log** を開きます。
4. **Disable Access Log Buffering** を選択します。
5. **Save Configuration** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

第3章 ロック数の調整

Directory Server のロックメカニズムは、Directory Server プロセスのコピーを同時に実行できる数を制御します。たとえば、インポートジョブ中に Directory Server は、**/run/lock/dirsrv/slapd-*instance_name*/imports/** ディレクトリーにロックを設定し、**ns-slapd** (Directory Server) プロセスや別のインポートまたはエクスポート操作が実行されないようにします。

サーバーが利用可能なロックが不足すると、以下のエラーが **/var/log/dirsrv/slapd-*instance_name*/errors** ファイルに記録されます。

```
libdb: Lock table is out of available locks
```

ロックテーブルに使用可能なロックがないことをエラーメッセージが示している場合は、ロックの数を2倍にします。問題が解決しない場合は、値を再度 2 倍にします。

3.1. ロック数の手動監視

コマンドラインを使用してロックの数を監視するには、次のコマンドを実行します。

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
    -s sub -b "cn=database,cn=monitor,cn=ldbm database,cn=plugins,cn=config"
    nsslapd-db-current-locks nsslapd-db-max-locks
```

監視属性の詳細は、「Directory Server Configuration, Command, and File Reference 」の説明を参照してください。

3.2. 無料のデータベースロックを監視することによるデータ破壊の回避

データベースロックが不足すると、データが破損する可能性があります。これを回避するために、Directory Server は、デフォルトで、残りの空きデータベースロック数を 500 ミリ秒間隔で監視し、アクティブなデータベースロックの数が 90% 以上の場合、Directory Server はすべての検索を中止します。

間隔としきい値を変更できます。

1. たとえば、間隔を **600** ミリ秒に設定し、しきい値を **85** パーセントに設定するには、次のコマンドを実行します。

   ```
   # dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --locks-
   monitoring-enabled on --locks-monitoring-pause 600 --locks-monitoring-threshold 85
   ```

   --locks-monitoring-enabled on オプションは、この機能が有効であることを確認します。

   注記

   設定した間隔が長すぎると、次の監視チェックが行われる前にサーバーのロックが不足する可能性があります。間隔が短すぎると、サーバーが遅くなる可能性があります。

2. インスタンスを再起動します。

   ```
   # dsctl instance_name restart
   ```

3.3. コマンドラインを使用したロック数の設定

コマンドラインを使用してロックの数を設定するには、以下を実行します。

1. **dsconf backend config set** コマンドを使用して、ロックの数を更新します。たとえば、値を **20000** に設定するには、以下を実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --
locks=20000
```

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

3.4. WEB コンソールを使用したロック数の設定

Web コンソールを使用してロックの数を設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** メニューを開き、**Global Database Configuration** を選択します。
4. **Show Advanced Settings** をクリックします。
5. **Database Locks** フィールドの値を更新します。
6. **Save Configuration** をクリックします。
7. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

第4章 検索パフォーマンスの改善（および読み取りパフォーマンスの調整）

ディレクトリーに対して検索操作を改善する最も効果的な方法は、検索結果の妥当な制限と組み合わせて、エントリーの完全なインデックスを設定することです。

4.1. インデックスの使用

インデックスは、それが意味するように、特定のエントリーに特定の属性が含まれていることを示すタグであり、エントリーに関する他の詳細を含める必要はありません（スペースを節約し、検索結果をより速く返すことができます）。各インデックスは、Directory Server 属性と、その属性を照合する特定の方法を中心に編成されます。

- **Presence index (pres)** は、どのエントリーに属性が含まれているかを示します。
- **Equality index (eq)** は、特定の検索文字列に一致する属性値を示します。
- **Approximate index (approx)** は、文字列と音声的に一致する値を持つエントリーを表示する、効率的な**発音類似**検索に使用されます。
- **Substring index (sub)** は、指定の検索文字列に属性値のサブ文字列を照合します。（サーバーを維持するのに非常にコストが高い場合はこのインデックス）
- **International index** は、マッチングルールを使用して、英語以外の言語の値を含むディレクトリー内の文字列を照合します。

注記

インデックスの詳細は、『『Red Hat Directory Server Administration Guide』』の「『Managing Indexes』」を参照してください。

ただし、インデックスを作成するだけでは、サーバーのパフォーマンスは直接向上しません。インデックスを維持すると、サーバーによって維持される各インデックスに対して、変更に含まれる各属性を検証する必要があるため、変更、追加、および削除のすべての操作で Directory Server に負担がかかります。

1. Directory Server は add 操作または modify 操作を受け取ります。
2. Directory Server は indexing 属性を調べ、属性値に対してインデックスが維持されているかどうかを判断します。
3. 作成した属性値がインデックス化されると、Directory Server は新しいインデックスエントリーを生成します。
4. サーバーがインデックス化を完了すると、クライアントの要求に応じて実際の属性値が作成されます。

たとえば、Directory Server はエントリーを追加します。

```
dn: cn=John Doe, ou=People,dc=example,dc=com
objectclass: top
objectClass: person
objectClass: orgperson
objectClass: inetorgperson
```

```
cn: John Doe
cn: John
sn: Doe
ou: Manufacturing
ou: people
telephoneNumber: 408 555 8834
description: Manufacturing lead for the Z238 line of widgets.
```

Directory Server は以下のインデックスを維持しています。

- ***cn*** (一般名 (common name)) 属性および ***sn*** (姓 (surname)) 属性の等価、概算、および部分文字列インデックス。
- 電話番号属性の等価および部分文字列のインデックス。
- 説明属性の部分文字列インデックス。

そのエントリーをディレクトリーに追加する場合は、Directory Server で以下の手順を実行する必要があります。

1. **John** および **John Doe** の ***cn*** 等価インデックスエントリーを作成します。
2. **John** および **John Doe** の適切な ***cn*** の概算インデックスエントリーを作成します。
3. **John** および **John Doe** の適切な ***cn*** 部分文字列インデックスエントリーを作成します。
4. **Doe** の ***sn*** 等価インデックスエントリーを作成します。
5. **Doe** に対する ***sn*** 概算インデックスエントリーを作成します。
6. **Doe** に適切な ***sn*** 部分文字列インデックスエントリーを作成します。
7. **408 555 8834** に電話番号の等価インデックスエントリーを作成します。
8. **408 555 8834** に適切な電話番号の部分文字列インデックスエントリーを作成します。
9. **Manufacturing lead for the Z238 line of widgets** の適切な説明部分文字列インデックスエントリーを作成します。この文字列に対して多数の部分文字列エントリーが生成されます。

新しいインデックスを作成する前に、インデックスを維持するためのオーバーヘッドと、検索パフォーマンスの潜在的な改善とのバランスを調整するようにしてください。特に重要なのは、保持するインデックスの種類を、ディレクトリーに格納されている情報の種類と、ユーザーが日常的に検索する情報の**種類**に一致させることです。

- 概算インデックスは、通常、数字を含む属性 (電話番号など) には効率的ではありません。
- 部分文字列のインデックスはバイナリー属性では機能しません。
- 等価インデックスは、値が大きい場合に使用しないようにしてください (例: 暗号化データを含む写真やパスワードを含む属性など)。
- 検索であまり使用されない属性のインデックスを維持することは、グローバル検索のパフォーマンスを向上させることなく、オーバーヘッドを増加させます。
- インデックス化されていない属性は、検索要求で依然として指定できますが、検索のタイプによっては検索パフォーマンスが大幅に低下する可能性があります。

- 保守するインデックスが多いほど、必要なディスク領域が多くなります。

注記

インデックスの作成は、検索操作の負荷が高く、変更操作の負荷が低いディレクトリーに対して大きな効果があります。

4.2. DIRECTORY SERVER リソース設定のチューニング

複数のパラメーターを設定して、Directory Server が使用するリソース量を管理および改善できます。

4.2.1. コマンドラインを使用した Directory Server リソース設定の更新

コマンドラインを使用してサーバーリソース設定を更新するには、以下を実行します。

1. パフォーマンス設定を更新します。

```
dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
parameter_name=setting
```

以下のパラメーターを設定できます。

- ***nsslapd-threadnumber***: ワーカースレッドの数を設定します。
- ***nsslapd-maxdescriptors***: ファイル記述子の最大数を設定します。
- ***nsslapd-timelimit***: 検索時間の制限を設定します。
- ***nsslapd-sizelimit***: 検索サイズの制限を設定します。
- ***nsslapd-pagedsizelimit***: ページングされた検索サイズの制限を設定します。
- ***nsslapd-idletimeout***: アイドル接続のタイムアウトを設定します。
- ***nsslapd-ioblocktimeout***: 入出力 (I/O) ブロックのタイムアウトを設定します。
- ***nsslapd-ndn-cache-enabled***: 正規化された DN キャッシュを有効または無効にします。
- ***nsslapd-ndn-cache-max-size***: ***nsslapd-ndn-cache-enabled*** が有効な場合は、正規化された DN キャッシュサイズを設定します。
- ***nsslapd-outbound-ldap-io-timeout***: アウトバウンド I/O タイムアウトを設定します。
- ***nsslapd-maxbersize***: BER(Basic Encoding Rules) の最大サイズを設定します。
- ***nsslapd-maxsasliosize***: SASL (Simple Authentication and Security Layer) の最大 I/O サイズを設定します。
- ***nsslapd-listen-backlog-size***: 受信接続をの受信に利用できるソケットの最大数を設定します。
- ***nsslapd-max-filter-nest-level***: ネストされたフィルターの最大レベルを設定します。
- ***nsslapd-ignore-virtual-attrs***: 仮想属性検索を有効または無効にします。
- ***nsslapd-connection-nocanon***: 逆引き DNS ルックアップのを有効または無効にします。

- ***nsslapd-enable-turbo-mode***: ターボモード機能を有効または無効にします。

これらのパラメーターの詳細は、「『Red Hat Directory Server Configuration, Command, and File Reference.』」を参照してください。

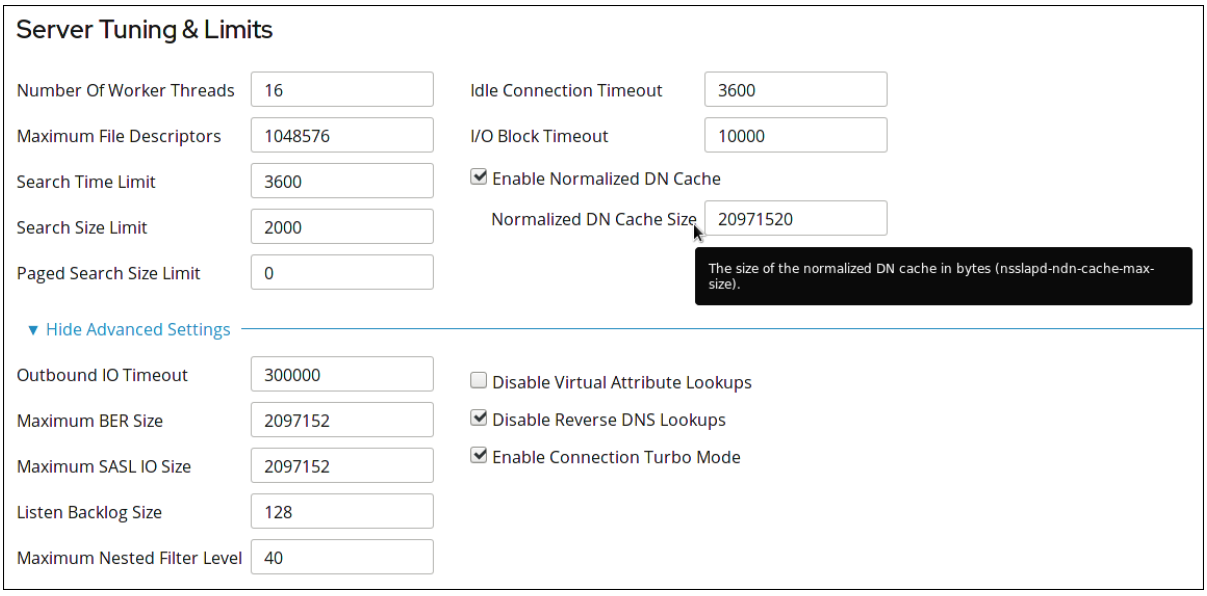
2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

4.2.2. Web コンソールを使用した Directory Server リソース設定の更新

Web コンソールを使用してサーバーリソース設定を更新するには、以下を行います。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Server Settings** メニューを開き、**Tuning & Limits** を選択します。
4. 設定を更新します。必要に応じて、**Show Advanced Settings** をクリックし、すべての設定を表示します。



ツールヒントとパラメーターの **cn=config** エントリーで対応する属性名を表示するには、設定の上にマウスのカーソルを合わせます。詳細は、「『Red Hat Directory Server Configuration, Command, and File Reference』」のパラメーターの説明を参照してください。

5. **Save Configuration** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

4.3. インデックススキャン制限の設定

大規模なディレクトリーでは、検索結果リストが膨大になる可能性があります。100 万の **inetorgperson** エントリーがあるディレクトリーには、**(objectclass=inetorgperson)** のようなフィルターで返される 100 万のエントリーがあり、***sn*** 属性のインデックスには少なくとも 100万 のエントリーがあります。

データベースから長い ID リストを読み込むと、検索パフォーマンスが大幅に低下します。設定パラメーター **nsslapd-idlistscanlimit** は、キーがプライマリインデックス全体と一致すると見なされる前に読み取られるIDの数に制限を設定します（つまり、検索は、異なるリソース制限のセットでインデックス化されていない検索として処理されます）。

大規模なインデックスでは、インデックスに一致する検索をインデックス化されていないの検索として扱う方が実際には効率的です。検索操作では、ディレクトリーとほぼ同じサイズのインデックスとディレクトリー自体を検索するのではなく、結果（ディレクトリー全体）を処理するために 1 か所を検索することのみが必要です。

nsslapd-idlistscanlimit 属性のデフォルト値は **4000** です。これは、データベースのサイズとアクセスパターンの一般的な範囲で優れたパフォーマンスを実現します。通常、この値は変更する必要はありません。データベースインデックスが 4000 エントリーより若干大きくても、全体的なディレクトリーよりも大幅に小さい場合には、スキャン制限を引き上げると検索が改善されます。そうでないとデフォルトの制限である 4000 に達します。

一方、制限を下げると、4000 エントリーの上限に達する検索が大幅に高速になりますが、すべてのエントリーをスキャンする必要はありません。

4.3.1. コマンドラインを使用したインデックススキャン制限の設定

コマンドラインを使用してインデックススキャン制限を設定するには、以下を実行します。

1. たとえば、検索操作中に Directory Server が検索するエントリー ID の数を **8000** に設定するには、次のコマンドを実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --
idlistscanlimit=8000
```

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

4.3.2. Web コンソールを使用したインデックススキャン制限の設定

Web コンソールを使用してインデックススキャン制限を設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** タブで、**Global Database Configuration** を選択します。
4. **ID List Scan Limit** フィールドの値を更新します。
5. **Save Configuration** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

4.4. 粒度の細かい ID リストサイズ

大規模なデータベースでは、一部のクエリーが大量の CPU および RAM リソースを消費する可能性があ

ります。パフォーマンスを改善するには、**nsslapd-idlistscanlimit** 属性を使用して、データベースのすべてのインデックスに適用されるデフォルトの ID スキャン制限を設定します。ただし、特定インデックスの制限や、ID リストの不使用を定義すると便利な場合があります。**nsIndexIDListScanLimit** 属性を使用して、さまざまなタイプの検索フィルターの ID リストのスキャン制限に個別の設定を指定できます。

objectClass 属性の制限を設定するには、**nsIndexIDListScanLimit** パラメーターを DN **cn=objectclass,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config** に追加します。

NsIndexIDListScanLimit 属性は複数値であり、以下のパラメーターのリストを値として取ります。

```
nsIndexIDListScanLimit: limit=NNN [type=eq[,sub,...]] [flags=AND[,XXX,...]] [values=val[,val,...]]
```

- **limit**: ID リストの最大サイズ。有効な値は以下のとおりです。
  - **-1**: 無制限。
  - **0**: インデックスを使用しない。
  - **1 から最大 32 ビットの整数 (2147483647)**: ID の最大数。
- **type**: 任意。インデックスのタイプ。**eq**、**sub**、**pres** など。値は、インデックス定義に指定された実際の ***nsIndexType*** の 1 つにする必要があります。たとえば、**nsIndexType=eq** が定義されていない場合は、**type=eq** を使用することはできません。
- **flags**: 任意。スキャン制限を適用する動作を変更するフラグ。有効な値は以下のとおりです。
  - **AND**: **AND** 句に属性が含まれる検索にのみスキャン制限を適用します。
  - **OR**: **OR** 句に属性が含まれる検索にのみスキャン制限を適用します。
- **値**: 任意。制限を適用するために検索フィルターと一致する必要がある値のコンマ区切りリスト。一致は一度に 1 回ずつ行われるため、いずれかの値が一致すると値は一致します。

  値は、一度に 1 つのタイプでのみ使用する必要があります。

  値はインデックスタイプに対応する必要があり、インデックスが適用される属性の構文に対応する必要があります。たとえば、整数ベースの属性 ***uidNumber*** を指定し、**eq** にインデックス化されている場合、 **type=eq values=abc** を使用することはできません。

  値にスペース、コンマ、NULL、またはエスケープが必要なその他の値が含まれている場合は、LDAP フィルターエスケープ構文を使用する必要があります。バックスラッシュ（\）の後に文字の2 桁の 16進 コードを続けます。以下の例では、DN 値のコンマは **\2C** でエスケープされます。

  ```
  nsIndexIDListScanLimit: limit=0 type=eq
  values=uid=user\2Cou=People\2Cdc=example\2Cdc=com
  ```

例4.1 nsIndexIDListScanLimit の設定

オブジェクトクラス **inetOrgPerson** を含む 1,000万のエントリーを持つ大規模なデータベースでは、**(&(objectClass=inetOrgPerson)(uid=user))** を検索すると、最初に **objectClass=inetOrgPerson** と一致する 1,000 万の ID すべてが含まれる IDリストが作成されます。データベースがフィルターの 2 番目の部分を適用すると、**uid=user** に一致するオブジェクトの結果一覧を検索します。この場合、特定のインデックスの制限を定義するか、ID リストをまったく使用しないと便利です。

AND 句で **objectClass=inetOrgPerson** に ID リストが作成されないように設定するには、以下の **nsIndexIDListScanLimit** を追加します。

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=objectclass,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsIndexIDListScanLimit
nsIndexIDListScanLimit: limit=0 type=eq flags=AND values=inetOrgPerson

modifying entry "cn=objectclass,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config"
```

AND 句で使用すると、**objectclass=inetOrgPerson** に ID リストは作成されません。他のすべての状況では、***nsslapd-idlistscanlimit*** の値が適用されます。

4.5. 検索用のデータベースキャッシュの調整

検索パフォーマンスに影響するデータベース属性は、主にサーバーが利用可能なメモリー量を定義します。データベースのキャッシュサイズ属性に設定できる最大値は、マシン上の実際のメモリー量によって異なります。大まかに言って、マシンで使用可能なメモリーの量は、デフォルトのデータベースキャッシュサイズの合計と各エントリーキャッシュサイズの合計よりも常に大きい必要があります。

これらのキャッシュサイジング属性を変更する場合は注意が必要です。これらの属性でサーバーパフォーマンスを向上させる機能は、データベースのサイズ、マシンで利用可能な物理メモリー量、およびディレクトリー検索がランダムかどうか（つまり、ディレクトリークライアントがランダムで広範囲に分散したディレクトリデータを検索するかどうか）によって異なります。

データベースがメモリーに収まらず、検索がランダムである場合は、これらの属性に設定した値を増やしても、ディレクトリーのパフォーマンスは向上しません。実際、これらの属性を変更すると、全体的なパフォーマンスが低下する可能性があります。

ディレクトリーデータを保存するために使用される各データベースの属性は、サイズを変更できます。

検索操作でキャッシュヒット比率を改善するには、「データベースキャッシュサイズの設定」 の説明のように、***nsslapd-dbcachesize*** パラメーターの値を編集して、Directory Server がデータベースキャッシュに保持するデータの量を増やします。

4.6. 特殊なエントリーの管理

Directory Server は、**cn=config** エントリーを **/etc/dirsrv/slapd-*instance_name*/dse.ldif** 設定ファイルに保存し、通常のエントリーと同じ拡張性の高いデータベースには格納しません。このため、通常のユーザーまたはグループを **cn=config** に保存しないでください。

第5章 トランザクションログのチューニング

すべての Directory Server には、管理するすべてのデータベースの操作を書き込むトランザクションログが含まれています。変更などのディレクトリーデータベース操作が実行されるたびに、サーバーは LDAP 操作の結果として呼び出されるすべてのデータベース操作に対して単一のデータベーストランザクションを作成します。これには、エントリーインデックスファイルのエントリーデータの更新と、すべての属性インデックスの更新の両方が含まれます。**すべて**の操作に成功すると、サーバーはトランザクションをコミットし、トランザクションログに操作を書き込み、トランザクション全体がディスクに書き込まれたことを確認します。操作の**いずれか** が失敗すると、サーバーはトランザクションをロールバックし、すべての操作は破棄されます。サーバー内のこのall-or-nothing アプローチは、更新操作が**アトミック** であることを保証します。操作全体が永続的かつ変更不可能で成功するか、失敗します。

定期的に、Directory Server（内部ハウスキーピングスレッドを介して）はトランザクションログの内容を実際のデータベースインデックスファイルにフラッシュし、トランザクションログにトリミングが必要であるかどうかを確認します。

サーバーで停電などの障害が発生し、異常にシャットダウンした場合でも、最近のディレクトリー変更に関する情報はトランザクションログに保存されます。サーバーを再起動すると、ディレクトリーはエラー状態を自動的に検出し、データベーストランザクションログを使用してデータベースを復元します。

データベーストランザクションのロギングやデータベースの復元は、介入を必要としない自動プロセスですが、パフォーマンスを最適化するためにデータベーストランザクションロギング属性の一部を調整することが推奨されます。

警告

トランザクションのロギング属性は、システムの変更や診断のためにのみ提供されます。これらの設定は、Red Hat テクニカルサポートのアドバイスでのみ変更する必要があります。これらの属性と他の設定属性の設定に一貫性がないと、ディレクトリーが不安定になる可能性があります。

5.1. データベースディレクトリーの別のディスクまたはパーティションへの移動

より高いパフォーマンスを実現するには、ディレクトリーサーバーデータベースとトランザクションログを、NVMe (不揮発性メモリーエクスプレス) ドライブや SSD などの高速ドライブに保存します。

たとえば、Directory Server インスタンスを実行し、**/dev/nvme0n1p1** パーティションを **/var/lib/dirsrv/slapd-*instance_name*/db/** ディレクトリーにマウントする場合は、以下を実行します。

1. インスタンスを停止します。

```
# systemctl stop dirsrv@instance_name
```

2. **/dev/nvme0n1p1** パーティションを一時ディレクトリーにマウントします。例:

```
# mount /dev/nvme0n1p1 /mnt/
```

3. **/var/lib/dirsrv/slapd-*instance_name*/db/** ディレクトリーのコンテンツを一時的なマウントポイントにコピーします。

```
# mv /var/lib/dirsrv/slapd-instance_name/db/* /mnt/
```

4. 一時ディレクトリーをアンマウントします。

```
# umount /mnt/
```

5. **/var/lib/dirsrv/slapd-*instance_name*/db/** も個別のマウントポイントである場合は、ディレクトリーをアンマウントします。

```
# umount /var/lib/dirsrv/slapd-instance_name/db/
```

6. **/etc/fstab** ファイルを更新して、システムの起動時に **/dev/nvme0n1p1** パーティションを **/var/lib/dirsrv/slapd-*instance_name*/db/** に自動的にマウントします。詳細は、「『Red Hat System Administrator's Guide』」の該当セクションを参照してください。

7. ファイルシステムをマウントします。エントリーを **/etc/fstab** に追加した場合:

```
# mount /var/lib/dirsrv/slapd-instance_name/db/
```

8. SELinux が **Enforcing** モードで実行している場合は、SELinux コンテキストを復元します。

```
# restorecon -Rv /var/lib/dirsrv/slapd-instance_name/db/
```

9. インスタンスを起動します。

```
# systemctl start dirsrv@instance_name
```

5.2. データベースチェックポイント間隔の変更

Directory Server は一定間隔でトランザクションログに記録された操作をデータベースインデックスファイルに書き込み、データベーストランザクションログのチェックポイントエントリーをログに記録します。チェックポイントエントリーは、データベースインデックスに既に書き込まれた変更を示すことにより、トランザクションログからの復元を開始する場所を示し、復元プロセスを高速化します。

デフォルトでは、Directory Server は、60 秒ごとに チェックポイントエントリーをデータベーストランザクションログに送信するように設定されています。チェックポイントの間隔を長くすると、ディレクトリーの書き込み操作のパフォーマンスが向上する可能性があります。ただし、チェックポイントの間隔を長くすると、異常シャットダウンの後にディレクトリーデータベースを復元するのに必要な時間が長くなり、データベーストランザクションログファイルが大きいため、より多くのディスク領域が必要になる場合があります。したがって、データベースの最適化に精通し、変更の影響を完全に評価できる場合にのみ、この属性を変更します。

5.2.1. コマンドラインを使用したデータベースチェックポイント間隔の変更

コマンドラインを使用してデータベースのチェックポイント間隔を変更するには、以下を入力します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --checkpoint-interval=120
```

この例では、間隔を 120 秒に変更します。

5.2.2. Web コンソールを使用したデータベースチェックポイント間隔の変更

Web コンソールを使用してデータベースのチェックポイント間隔を変更するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** タブで、**Global Database Configuration** を選択します。
4. **Show Advanced Settings** をクリックします。
5. **Database Checkpoint Interval** フィールドの値を更新します。
6. **Save Configuration** をクリックします。

5.3. 永続的なトランザクションの無効化

永続的なトランザクションログとは、トランザクション内の一連のデータベース操作で構成される各 LDAP 更新操作が物理的にディスクに書き込まれることを意味します。各 LDAP 操作は複数のデータベース操作で構成できますが、各 LDAP 操作は単一のデータベーストランザクションとして処理されます。各 LDAP 操作はアトミックおよび永続的です。

警告

永続トランザクションを無効にすると、データ損失のリスクがありますが、Directory Server の書き込みパフォーマンスが向上します。

永続トランザクションログが無効の場合、すべてのディレクトリーデータベース操作はデータベーストランザクションのログファイルに書き込まれますが、即座および物理的にディスクに書き込まれない場合があります。ディレクトリの変更が論理データベースのトランザクションログファイルに書き込まれ、システムのクラッシュ時にディスクには物理的に書き込まれなかった場合、変更を復元することはできません。永続トランザクションが無効の場合、復元されたデータベースは一貫性がありますが、システムクラッシュの直前に完了した LDAP 書き込み操作の結果は反映されません。

デフォルトでは、永続データベーストランザクションログが有効になっています。永続トランザクションログを無効にするには、以下を実行します。

1. Directory Server インスタンスを停止します。

```
# dsctl instance_name stop
```

2. **/etc/dirsrv/slapd-*instance_name*/dse.ldif** ファイルを編集し、**cn=config,cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-db-durable-transaction*** パラメーターを **off** に設定します。

```
dn: cn=config,cn=ldbm database,cn=plugins,cn=config
...
nsslapd-db-durable-transaction: off
...
```

3. Directory Server インスタンスを停止します。

```
# dsctl instance_name start
```

5.4. トランザクションのバッチ処理の指定

完全なトランザクションの永続性が必要ない場合に更新パフォーマンスを向上するには、以下のコマンドを使用します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --txn-batch-val=value
```

--txn-batch-val は、Directory Server がトランザクションログにコミットする前にバッチ処理するトランザクションの数を指定します。この値を **0** より大きな値に設定すると、キューに置かれたトランザクション数がこの値と等しくなるまで、サーバーはトランザクションのコミットを遅延します。

第6章 データベースキャッシュ設定の管理

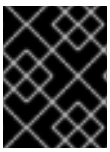
Directory Server は以下のキャッシュを使用します。

- 個々のディレクトリーエントリーが含まれる **Entry キャッシュ**。
- **DN キャッシュ**は、DN と RDN をエントリーに関連付けるために使用されます。
- データベースインデックスファイル ***.db** および ***.db4** が含まれる**データベースキャッシュ**。

最高のパフォーマンス向上を実現するには、すべてのキャッシュサイズですべてのレコードを保存できる必要があります。推奨される自動サイズ設定機能を使用せず、使用可能な RAM を十分確保できない場合は、前に示した順序でキャッシュに空きメモリーを割り当てます

6.1. データベースおよびエントリーキャッシュの自動サイズ設定機能

デフォルトでは、Directory Server は、データベースおよびエントリーキャッシュ向けに最適化されたサイズを自動的に決定します。自動サイズ調整は、インスタンスの起動時にサーバーのハードウェアリソースに基づいて両方のキャッシュのサイズを最適化します。



重要

Red Hat は、自動チューニング設定の使用を推奨します。エントリーキャッシュサイズは手動で設定しないでください。

6.1.1. データベースおよびエントリーキャッシュの自動サイズ調整を手動で再有効化

10.1.1 より前のバージョンからインスタンスをアップグレードしたり、エントリーキャッシュサイズを手動で設定した場合は、エントリーキャッシュの自動チューニングを有効にできます。

cn=config,cn=ldbm database,cn=plugins,cn=config エントリーの以下のパラメーターは自動サイズ設定を制御します。

nsslapd-cache-autosize

この設定では、データベースおよびエントリーキャッシュの自動サイズ設定に割り当てる空き RAM の量を制御します。自動サイズ調整が有効になります。

- データベースおよびエントリーキャッシュの両方で、***nsslapd-cache-autosize*** パラメーターが **0** より大きい値に設定されている場合は、
- データベースキャッシュの場合は、***nsslapd-cache-autosize*** パラメーターおよび ***nsslapd-dbcachesize*** パラメーターが **0** に設定されている場合。
- エントリーキャッシュの場合は、***nsslapd-cache-autosize*** パラメーターおよび ***nsslapd-cachememsize*** パラメーターが **0** に設定されている場合。

nsslapd-cache-autosize-split

この値は、データベースキャッシュに使用される RAM の割合を設定します。残りの割合はエントリーキャッシュに使用されます。

データベースキャッシュに 1.5 GB を超える RAM を使用しても、パフォーマンスは向上しません。そのため、Directory Server はデータベースキャッシュを 1.5 GB に制限します。

データベースおよびエントリーキャッシュの自動サイズ設定を有効にするには、以下を実行します。

1. Directory Server インスタンスを停止します。

```
# systemctl stop dirsrv@instance_name
```

2. **/etc/dirsrv/slapd-*instance_name*/dse.ldif** ファイルをバックアップします。

```
# cp /etc/dirsrv/slapd-instance_name/dse.ldif \
    /etc/dirsrv/slapd-instance_name/dse.ldif.bak.$(date "+%F_%H-%M-%S")
```

3. **/etc/dirsrv/slapd-*instance_name*/dse.ldif** ファイルを編集します。

   a. データベースおよびエントリーキャッシュに使用する空きシステム RAM の割合を設定します。たとえば、10% を設定するには、以下を実行します。

```
nsslapd-cache-autosize: 10
```

注記

nsslapd-cache-autosize パラメーターを **0** に設定する場合は、さらに以下を設定する必要があります。

- **cn=config,cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-dbcachesize*** を **0** に設定して、自動サイズ設定のデータベースキャッシュを有効にします。
- **cn=*database_name*,cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-cachememsize*** を **0** に設定して、データベースの自動サイズ設定エントリーキャッシュを有効にします。

   b. 必要に応じて、空きシステム RAM からデータベースキャッシュに使用する割合を設定します。たとえば、40% を設定します。

```
nsslapd-cache-autosize-split: 40
```

   Directory Server は、エントリーキャッシュに残り 60% の空きメモリーを使用します。

   c. 変更を保存します。

4. Directory Server インスタンスを停止します。

```
# systemctl start dirsrv@instance_name
```

例6.1 *nsslapd-cache-autosize* および *nsslapd-cache-autosize-split* パラメーター

以下の設定は、両方のパラメーターのデフォルトです。

```
nsslapd-cache-autosize: 10
nsslapd-cache-autosize-split: 40
```

この設定により、システムの空き RAM の 10% が使用されます(***nsslapd-cache-autosize***)。このメモリーから、データベースキャッシュ(***nsslapd-cache-autosize-split***)に 40% が使用され、エントリーキャッシュに残りの 60% が使用されます。

空き RAM に応じて、以下のキャッシュサイズになります。

空き RAM (GB 単位)	データベースキャッシュサイズ	エントリーキャッシュサイズ
1 GB	40 MB	62 MB
2 GB	82 MB	122 MB
4 GB	164 MB	245 MB
8 GB	328 MB	492 MB
16 GB	512 MB[a]	1,126 MB
32 GB	512 MB [a]	2,764 MB
64 GB	512 MB [a]	6,042 MB
128 GB	512 MB [a]	12,596 MB

[a] Directory Server は、***nsslapd-dbcachesize*** パラメーターに 512 MB の制限を適用します。

6.2. 必要なキャッシュサイズの決定

Dsconf monitor dbmon コマンドを使用すると、実行時にキャッシュ統計を監視できます。

統計を表示するには、次のコマンドを実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com monitor dbmon
DB Monitor Report: 2020-06-24 11:31:27
--------------------------------------------------------
Database Cache:
 - Cache Hit Ratio:     50%
 - Free Space:          397.31 KB
 - Free Percentage:     2.2%
 - RO Page Drops:       0
 - Pages In:            2934772
 - Pages Out:           219075

Normalized DN Cache:
 - Cache Hit Ratio:     60%
 - Free Space:          19.98 MB
 - Free Percentage:     99.9%
 - DN Count:            100000
```

```
 - Evictions:          9282348

Backends:
  - dc=example,dc=com (userroot):
    - Entry Cache Hit Ratio:       66%
    - Entry Cache Count:           50000
    - Entry Cache Free Space:      2.0 KB
    - Entry Cache Free Percentage: 0.8%
    - Entry Cache Average Size:    8.9 KB
    - DN Cache Hit Ratio:          21%
    - DN Cache Count:              100000
    - DN Cache Free Space:         4.29 MB
    - DN Cache Free Percentage:    69.8%
    - DN Cache Average Size:       130.0 B
```

必要に応じて、**-b *back_end*** または **-x** オプションをコマンドに渡して、特定のバックエンドまたはインデックスの統計を表示します。

キャッシュのサイズが十分である場合、**DN Cache Count** の数は **Cache Count** バックエンドエントリーの値と一致します。さらに、すべてのエントリーと DN がそれぞれのキャッシュ内に収まる場合、**Entry Cache Count** カウント値は **DN Cache Count** 値と一致します。

この例の出力は、以下のようになります。

- 2.2% の空きデータベースキャッシュのみが残っている場合:

  ```
  Database Cache:
   ...
   - Free Space:          397.31 KB
   - Free Percentage:     2.2%
  ```

  ただし、効率的に操作するには 15% 以上の空きデータベースキャッシュが必要です。データベースキャッシュの最適なサイズを決定するには、サブディレクトリーと changelog データベースを含む **/var/lib/dirsrv/slapd-*instance_name*/db/** ディレクトリーのすべての ***.db** および ***.db4** ファイルのサイズを計算し、オーバーヘッドとして 12% を追加します。

  データベースキャッシュを設定するには、「データベースキャッシュサイズの設定」 を参照してください。

- **userroot** データベースの DN キャッシュは適正です。

  ```
  Backends:
    - dc=example,dc=com (userroot):
      ...
      - DN Cache Count:              100000
      - DN Cache Free Space:         4.29 MB
      - DN Cache Free Percentage:    69.8%
      - DN Cache Average Size:       130.0 B
  ```

  データベースの DN キャッシュには 100000 のレコードが含まれ、キャッシュの空き領域の割合は 69.8 % で、メモリーの各 DN には平均 130 バイトが必要です。

  DN キャッシュを設定するには、「データベースキャッシュサイズの設定」 を参照してください。

- **userroot** データベースのエントリーキャッシュの統計は、パフォーマンスを向上させるためにエントリーキャッシュの値を大きくする必要があることを示しています。

```
Backends:
 - dc=example,dc=com (userroot):
 ...
  - Entry Cache Count:           50000
  - Entry Cache Free Space:      2.0 KB
  - Entry Cache Free Percentage: 0.8%
  - Entry Cache Average Size:    8.9 KB
```

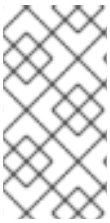
エントリーキャッシュのこのデータベースには 50000 のレコードが含まれており、残りの空き領域は 2 キロバイトのみです。Directory Server がすべての 100000 DN をキャッシュできるようにするには、キャッシュを最小でも 890 MB（100000 DN * 8,9 KB の平均エントリーサイズ）に増やす必要があります。ただし、Red Hat は、必要な最小サイズを次に大きい GB に切り上げし、その結果を 2 倍にすることを推奨します。この例では、エントリーキャッシュを 2 ギガバイトに設定する必要があります。

エントリーキャッシュを設定するには、「エントリーキャッシュサイズの手動設定」 を参照してください。

6.3. エントリーキャッシュサイズの手動設定

エントリーキャッシュは、検索操作および読み取り操作中に使用されるディレクトリーエントリーを保存するために使用されます。すべてのレコードを格納できるサイズにエントリーキャッシュを設定すると、検索操作のパフォーマンスに最も大きな影響を与えます。

エントリーのキャッシュが設定されていない場合、Directory Server は **id2entry.db** データベースファイルからエントリーを読み取り、DN をディスク上のフォーマットからメモリ内のフォーマットに変換します。キャッシュに保存されているエントリーにより、サーバーはディスク I/O および変換の手順をスキップできます。



注記

エントリーキャッシュサイズを手動で設定する代わりに、Red Hat は、ハードウェアリソースに基づいて最適化された設定の自動サイズ設定機能を推奨します。詳細は、「データベースおよびエントリーキャッシュの自動サイズ調整を手動で再有効化」を参照してください。

6.3.1. コマンドラインを使用したエントリーキャッシュサイズの手動設定

コマンドラインを使用してエントリーキャッシュサイズを手動で設定するには、以下を実行します。

1. 自動キャッシュチューニングを無効にします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --cache-autosize=0
```

2. 接尾辞とそれに対応するバックエンドを表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com suffix list
dc=example,dc=com (userroot)
```

このコマンドにより、各接尾辞の横にバックエンドデータベースが表示されます。次の手順で、接尾辞のデータベース名が必要です。

3. データベースのエントリーキャッシュサイズを設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix set --cache-memsize=2147483648 userRoot
```

このコマンドは、エントリーキャッシュを 2 ギガバイトに設定します。

4. Directory Service インスタンスを再起動します。

```
# dsctl instance_name restart
```

6.3.2. Web コンソールを使用したエントリーキャッシュサイズの手動設定

Web コンソールを使用してエントリーキャッシュサイズを手動で設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. Database タブで、**Global Database Configuration** を選択します。
4. **Automatic Cache Tuning** を無効にします。
5. **Save Configuration** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。
7. **Entry Cache Size (bytes)** フィールドにデータベースキャッシュのサイズを設定します。
8. **Save Configuration** をクリックします。
9. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

6.4. DN キャッシュのサイズ設定

entryrdn インデックスは、DN と RDN をエントリーに関連付けるために使用されます。これにより、サーバーは サブツリーの **rename**、エントリーの **move**、および **moddn** 操作を効率的に実行できます。DN キャッシュは、コストの高いファイル I/O および変換操作を回避するために、**entryrdn** インデックスのメモリー内表現をキャッシュするために使用されます。特にエントリーの **rename** および **move** 操作で最高のパフォーマンスを得るには、Directory Server がデータベース内のすべての DN をキャッシュできるサイズに DN キャッシュを設定します。

DN がキャッシュに保存されていない場合、Directory Server は **entryrdn.db** インデックスデータベースファイルから DN を読み取り、オンディスクフォーマットからインメモリーフォーマットに DN を変換します。キャッシュに保存されている DN により、サーバーはディスク I/O および変換の手順をスキップできます。

6.4.1. コマンドラインを使用した DN キャッシュのサイズ設定

コマンドラインを使用してデータベースの DN キャッシュサイズを設定するには、以下を実行します。

1. 接尾辞とそれに対応するバックエンドを表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com suffix list
dc=example,dc=com (userroot)
```

このコマンドにより、各接尾辞の横にバックエンドデータベースが表示されます。次の手順で、接尾辞のデータベース名が必要です。

2. 以下のコマンドを使用して DN キャッシュサイズを設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix set --
dncache-memsize=20971520 userRoot
```

このコマンドは、**userRoot** データベースの DN キャッシュを 20 メガバイトに設定します。

3. Directory Service インスタンスを再起動します。

```
# dsctl instance_name restart
```

6.4.2. Web コンソールを使用した DN キャッシュのサイズ設定

Web コンソールを使用してデータベースの DN キャッシュサイズを設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** タブで、DN キャッシュサイズを設定する接尾辞を選択します。
4. **DN Cache Size (bytes)** フィールドにサイズ（バイト単位）を入力します。
5. **Save Configuration** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

6.5. データベースキャッシュサイズの設定

データベースキャッシュには、データベースの Berkeley データベースインデックスファイルが含まれます。つまり、データベースによる属性のインデックス化に使用されるすべての ***.db** およびその他のファイルになります。この値は、Berkeley DB API 関数 **set_cachesize()** に渡されます。

このキャッシュサイズは、エントリーキャッシュサイズに比べて Directory Server のパフォーマンスへの影響は少ないですが、エントリーキャッシュサイズを設定した後に RAM に余裕がある場合は、データベースキャッシュに割り当てるメモリ量を増やします。

オペレーティングシステムにはファイルシステムのキャッシュもあり、これがデータベースのキャッシュと RAM 使用で競合することがあります。ファイルシステムキャッシュの設定やファイルシステムキャッシュの監視についての詳細は、オペレーティングシステムのドキュメントを参照してください。

注記

エントリーキャッシュサイズを手動で設定する代わりに、Red Hat は、ハードウェアリソースに基づいて最適化された設定の自動サイズ設定機能を推奨します。詳細は、「データベースおよびエントリーキャッシュの自動サイズ調整を手動で再有効化」を参照してください。

6.5.1. コマンドラインを使用したデータベースキャッシュサイズの手動設定

コマンドラインを使用してデータベースキャッシュサイズを手動で設定するには、以下を実行します。

1. 自動キャッシュチューニングを無効にし、データベースのキャッシュサイズを設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend config set --cache-autosize=0 --dbcachesize=268435456
```

このコマンドは、データベースキャッシュを 256 メガバイトに設定します。

2. Directory Service インスタンスを再起動します。

```
# dsctl instance_name restart
```

6.5.2. Web コンソールを使用したデータベースキャッシュサイズの手動設定

Web コンソールを使用してデータベースキャッシュサイズを手動で設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** タブで、**Global Database Configuration** を選択します。
4. **Automatic Cache Tuning** を無効にします。
5. **Save Configuration** をクリックします。
6. **Database Cache Size (bytes)** フィールドをデータベースキャッシュサイズに設定します。
7. **Save Configuration** をクリックします。
8. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

6.5.3. RAM ディスクでのデータベースキャッシュの保存

Directory Server インスタンスを実行しているシステムに十分な空き RAM がある場合は、必要に応じてデータベースキャッシュを RAM ディスクに保存し、パフォーマンスをさらに向上させることができます。

1. データベースキャッシュやメタデータのディレクトリーを RAM ディスク上に作成します。

```
# mkdir -p /dev/shm/slapd-instance_name/
```

2. ディレクトリーに以下の権限を設定します。

```
# chown dirsrv:dirsrv /dev/shm/slapd-instance_name/
# chmod 770 /dev/shm/slapd-instance_name/
```

3. Directory Server インスタンスを停止します。

```
# systemctl stop dirsrv@instance_name
```

4. **/etc/dirsrv/slapd-*instance_name*/dse.ldif** ファイルを編集し、**cn=bdb,cn=config,cn=ldbm database,cn=plugins,cn=config** エントリーの ***nsslapd-db-home-directory*** パラメーターに新しいパスを設定します。

```
dn: cn=bdb,cn=config,cn=ldbm database,cn=plugins,cn=config
...
nsslapd-db-home-directory: /dev/shm/slapd-instance_name/
```

nsslapd-db-home-directory 属性が存在しない場合は、新しい値で **cn=bdb,cn=config,cn=ldbm database,cn=plugins,cn=config** エントリーに追加します。

5. Directory Server インスタンスを停止します。

```
# systemctl start dirsrv@instance_name
```

注記

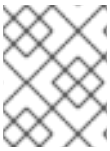
データベースキャッシュが RAM ディスクに保存されている場合は、再起動するたびに Directory Server がこれを再作成する必要があります。そのため、キャッシュが再作成されるまで、サービスの起動と初期操作は遅くなります。

第7章 DIRECTORY SERVER スレッドの数の設定

同時接続を処理するために Directory Server が使用するスレッドの数は、サーバーのパフォーマンスに影響します。たとえば、すべてのスレッドが時間のかかるタスク（**add** 操作など）の処理に追われている場合、新しい受信接続は、空いているスレッドがリクエストを処理できるようになるまでキューに置かれます。

サーバーが提供する CPU スレッド数が少ない場合、スレッド数を多く設定することでパフォーマンスが向上します。ただし、CPU スレッドが多数あるサーバーでは、設定値が高すぎてもパフォーマンスは向上しません。

デフォルトでは、Directory Server はスレッド数を自動的に計算します。この数値は、インスタンス起動時のサーバーのハードウェアリソースに基づいています。



注記

Red Hat は、自動チューニング設定の使用を推奨します。スレッド数は手動で設定しないでください。

7.1. 自動スレッドチューニング

自動スレッドチューニングを有効にすると、Directory Server は以下の最適化されたスレッド数を使用します。

CPU スレッド数	Directory Server スレッド数
1	16
2	16
4	24
8	32
16	48
32	64
64	96
128	192
256	384
512	512 [a]
1024	512 [a]

CPU スレッド数	Directory Server スレッド数
2048	512 [a]
[a] 推奨される最大スレッド数が適用されます。	

7.1.1. コマンドラインを使用した自動スレッドチューニングの有効化

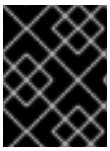
Directory Server は、利用可能なハードウェアスレッドに基づいてスレッド数を自動的に設定できます。この機能を有効にするには、以下を実行します。

1. スレッド数の自動設定を有効にします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-threadnumber="-1"
```

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```



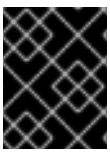
重要

スレッド数の自動設定を有効にした場合、***nsslapd-threadnumber*** パラメーターには、Directory Server の実行中に計算されたスレッド数が表示されます。

7.1.2. Web コンソールを使用した自動スレッドチューニングの有効化

Directory Server は、利用可能なハードウェアスレッドに基づいてスレッド数を自動的に設定できます。この機能を有効にするには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Server Settings** メニューを開き、**Tuning & Limits** を選択します。
4. **Number Of Worker Threads** フィールドを **-1** に設定します。
5. **Save** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。



重要

自動設定を有効にした場合、**Number Of Worker Threads** フィールドには、Directory Server の実行中に計算されたスレッド数が表示されます。

7.2. スレッド数の手動設定

特定の状況では、自動スレッドチューニングを使用する代わりに、一定数の Directory Server スレッドを手動で設定する必要がある場合があります。

注記

たとえば、Directory Server インスタンスを実行する仮想マシンの CPU コアを増やすため、ハードウェアスレッドの数が変更された場合は、スレッド数を手動で更新する必要があります。最適化および自動設定の使用に関する詳細は、「自動スレッドチューニング」 を参照してください。

7.2.1. コマンドラインを使用したスレッド数の手動設定

コマンドラインを使用してスレッド数を手動で設定するには、以下を実行します。

1. スレッドの数を設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-threadnumber="64"
```

このコマンドは、スレッド数を **64** に設定します。

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

7.2.2. Web コンソールを使用したスレッド数の手動設定

Web コンソールを使用してスレッド数を手動で設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Server Settings** メニューを開き、**Tuning & Limits** を選択します。
4. **Number Of Worker Threads** フィールドをスレッド数に設定します。
5. **Save** をクリックします。
6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

第8章 レプリケーションパフォーマンスのチューニング

8.1. マルチサプライヤーレプリケーションの効率性の向上

特にサーバーが WAN (Wide Area Network) を使用して接続している場合、マルチサプライヤーレプリケーション環境のレプリケーションレイテンシーは、複数のサプライヤーが同時に更新を受信すると高くなる可能性があります。これは、1 つのサプライヤーがレプリカを長期間解放せずに排他的にアクセスした場合に発生します。このような状況では、他のサプライヤーはこのコンシューマーに更新を送信できず、レプリケーションレイテンシーが増加します。

一定期間の経過後にレプリカを解放するには、レプリケーションサプライヤーとハブに ***nsds5ReplicaReleaseTimeout*** パラメーターを設定します。

注記

ほとんどの環境には、**60** 秒のデフォルト値が適しています。設定した値が高すぎるまたは低すぎると、レプリケーションのパフォーマンスに影響を及ぼす可能性があります。値を低く設定しすぎると、レプリケーションサーバーは常に相互に取得され、サーバーは多くの更新を送信できません。トラフィックの多いレプリケーション環境では、タイムアウトが長いと 1 つのサプライヤーのみがレプリカにアクセスする状況を改善できます。ただし、ほとんどの場合、**120** 秒よりも高い値を設定するとレプリケーションの速度が低下します。

8.1.1. コマンドラインを使用したレプリケーション解放タイムアウトの設定

コマンドラインを使用してレプリケーション解放のタイムアウトを設定するには、以下を実行します。

1. タイムアウト値を設定します。

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com replication set --
suffix="dc=example,dc=com" --repl-release-timeout=70
```

このコマンドは、**dc=example,dc=com** 接尾辞のレプリケーションリリースのタイムアウト値を **70** 秒に設定します。

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

8.1.2. Web コンソールを使用したレプリケーション解放タイムアウトの設定

Web コンソールを使用してレプリケーション解放のタイムアウトを設定するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Replication** メニューを開き、**Configuration** を選択します。
4. **Show Advanced Settings** をクリックします。

5. **Replication Release Timeout** フィールドにタイムアウト値を設定します。
6. **Save** をクリックします。
7. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

第9章 データベースリンクのパフォーマンスの調整

Directory Server の接続およびスレッド管理への変更により、データベースリンクのパフォーマンスを改善できます。

9.1. リモートサーバーへの接続の管理

各データベースリンクは、リモートサーバーへの接続のプールを維持します。本セクションでは、それらを最適化する方法を説明します。

9.1.1. コマンドラインを使用したリモートサーバーへの接続の管理

本セクションでは、特定のデータベースの設定とデフォルト設定を更新する方法を説明します。

9.1.1.1. 特定のデータベースのデータベースリンク接続管理設定の更新

特定のデータベースのデータベースリンク接続管理設定を更新するには、以下を実行します。

1. 以下のコマンドを使用して、データベースリンクの設定を更新します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-set
parameter=value link_name
```

設定可能なパラメーターのリストは、以下を入力します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-set --help
```

2. Directory Server インスタンスを再起動します。

```
# dsctl instance_name restart
```

9.1.1.2. デフォルトのデータベースリンク接続管理設定の更新

デフォルトのデータベースリンク接続管理設定を更新するには、以下のコマンドを使用します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set-def
parameter=value
```

設定可能なパラメーターのリストは、以下を入力します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set-def --help
```

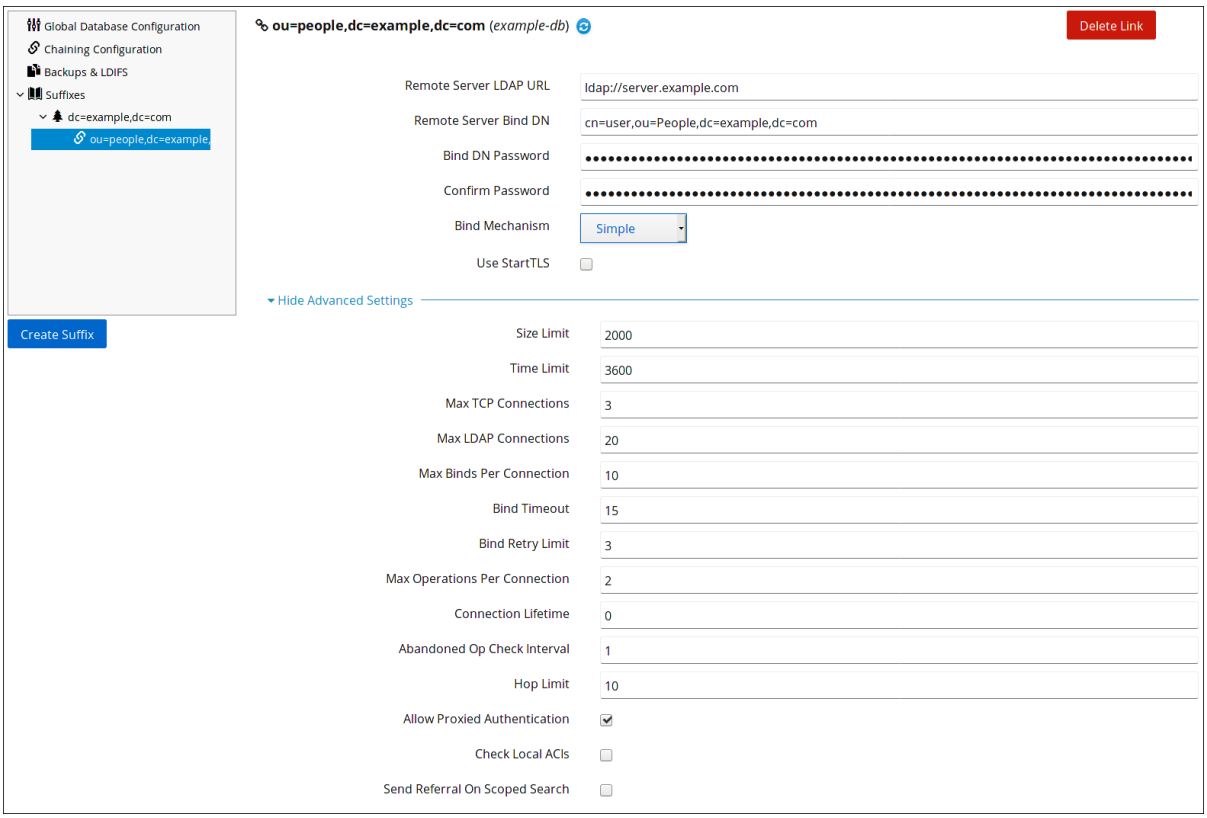
9.1.2. Web コンソールを使用したリモートサーバーへの接続の管理

本セクションでは、特定のデータベースの設定とデフォルト設定を更新する方法を説明します。

9.1.2.1. 特定のデータベースのデータベースリンク接続管理設定の更新

特定のデータベースのデータベースリンク接続管理設定を更新するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** タブで、更新するデータベースリンク設定を選択します。
4. **Show Advanced Settings** をクリックします。
5. 詳細設定エリアでフィールドを更新します。



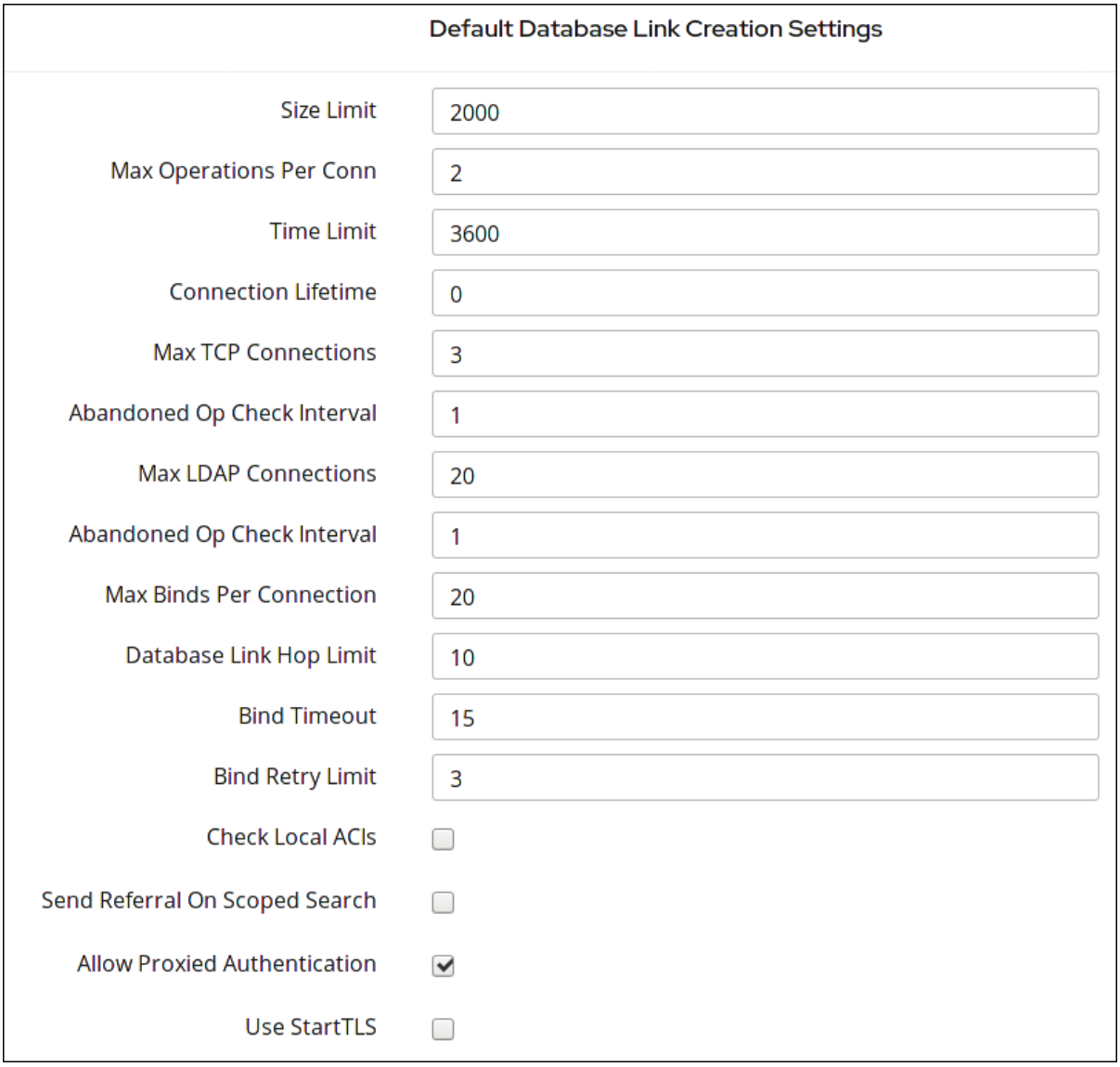
ツールヒントとパラメーターの **cn=config** エントリーで対応する属性名を表示するには、設定の上にマウスのカーソルを合わせます。詳細は、「『Red Hat Directory Server Configuration, Command, and File Reference』」のパラメーターの説明を参照してください。

6. **Save Configuration** をクリックします。
7. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

9.1.2.2. デフォルトのデータベースリンク接続管理設定の更新

デフォルトのデータベースリンク接続設定を更新するには、以下を実行します。

1. Web コンソールで Directory Server ユーザーインターフェースを開きます。詳細は、『『Red Hat Directory Server Administration Guide』』の「Logging Into Directory Server Using the Web Console」セクションを参照してください。
2. インスタンスを選択します。
3. **Database** タブで **Chaining Configuration** を選択します。
4. **Default Database Link Creation Settings** エリアのフィールドを更新します。



ツールヒントとパラメーターの **cn=config** エントリーで対応する属性名を表示するには、設定の上にマウスのカーソルを合わせます。詳細は、「『Red Hat Directory Server Configuration, Command, and File Reference』」のパラメーターの説明を参照してください。

5. **Save Default Settings** をクリックします。

6. **Actions** ボタンをクリックして、**Restart Instance** を選択します。

9.2. 通常処理中のエラーの検出

データベースリンクとリモートサーバーとの間の通常のチェーン操作中にエラーを検出することで、サーバーのパフォーマンスを保護します。データベースリンクには ***nsMaxResponseDelay*** と ***nsMaxTestResponseDelay*** の 2 つの属性があります。これは、リモートサーバーが応答しないかどうかを判断するために連携します。

最初の属性 ***nsMaxResponseDelay*** は、LDAP 操作が完了するまで最大期間を設定します。この属性に指定された時間よりも長い時間が操作にかかる場合、データベースリンクのサーバーはリモートサーバーがオンラインでないことを想定します。

nsMaxResponseDelay の期間が満たされると、データベースリンクがリモートサーバーに ping します。ping 時に、データベースリンクは、リモートサーバーに存在しないオブジェクトの単純な検索要求である別の LDAP 要求を発行します。Ping の期間は ***nsMaxTestResponseDelay*** を使用して設定されます。

nsMaxTestResponseDelay 期間が経過する前にリモートサーバーが応答しない場合は、エラーが返され、接続に down フラグが付けられます。データベースリンクとリモートサーバー間の接続はすべて 30 秒間ブロックされ、パフォーマンス低下からサーバーを保護します。30 秒後に、データベースリンクがリモートサーバーに対して行った操作リクエストは、通常どおりに続行されます。

どちらの属性も **cn=config,cn=chaining database,cn=plugins,cn=config** エントリーに保存されます。以下の表は、属性の詳細を示しています。

表9.1 データベースリンク処理エラー検出パラメーター

属性名	説明
nsMaxResponseDelay	エラーが疑われる前に、データベースリンクによる LDAP 操作要求にリモートサーバーで応答する最大時間。この期間は秒単位で指定されます。デフォルトの遅延期間は **60** 秒です。この遅延期間が満たされると、データベースリンクはリモートサーバーとの接続をテストします。
nsMaxTestResponseDelay	リモートサーバーが応答しないかどうかを確認するためにデータベースリンクによって実行されるテストの期間。この期間の経過前にリモートサーバーからの応答が返されなかった場合、データベースリンクはリモートサーバーが停止していることを想定し、後続の操作で接続は使用されません。この期間は秒単位で指定されます。デフォルトのテスト応答の遅延期間は **15** 秒です。

第10章 インポートパフォーマンスの改善

非常に大きなエントリーサイズまたは多数のエントリーの場合、インポート操作中にサーバーのパフォーマンスに悪影響を及ぼす可能性があります。本セクションでは、Directory Server 設定とオペレーティングシステム設定の両方を調整して、インポートパフォーマンスを向上する方法を説明します。

10.1. 大きなデータベースのインポートおよび大きな属性を持つインポートの DIRECTORY SERVER のチューニング

以下の状況でエントリーキャッシュを更新します。

- 非常に大きなデータベースをインポートする場合。
- 証明書チェーンまたはイメージを格納するバイナリー属性など、大きな属性を持つデータベースをインポートする場合。

エントリーキャッシュのサイズの設定に関する詳細は、「データベースおよびエントリーキャッシュの自動サイズ設定機能」 および 「エントリーキャッシュサイズの手動設定」 を参照してください。

10.2. DIRECTORY SERVER のチューニング： 多数のエントリーのインポート

多数のエントリーをインポートする場合、ユーザープロセスの最大数にオペレーティングシステムを設定すると、Directory Server のパフォーマンスを制限できます。

- プロセスの最大数を一時的に増やすには、次のコマンドを実行します。

  ```
  # ulimit -u 32000
  ```

  ユーザーがログオフすると、変更はデフォルト設定に戻ります。

- プロセスの最大数を永続的に増やすには、「 「How to set ulimit values 」 」を参照してください。

付録A 改訂履歴

改訂番号は本ガイドに関するものであり、Red Hat Directory Server のバージョン番号ではありません。

改訂 11.4-1 **Tue Nov 09 2021** **Marc Muehlfeld**
Red Hat Directory Server 11.4 版のガイドをリリース

改訂 11.3-1 **Tue May 11 2021** **Marc Muehlfeld**
Red Hat Directory Server 11.3 版のガイドをリリース

改訂 11.2-1 **Tue Nov 03 2020** **Marc Muehlfeld**
Red Hat Directory Server 11.2 版のガイドをリリース

改訂 11.1-1 **Tue Apr 28 2020** **Marc Muehlfeld**
Red Hat Directory Server 11.1 版のガイドをリリース

改訂 11.0-1 **Tue Nov 05 2019** **Marc Muehlfeld**
Red Hat Directory Server 11.0 版のガイドをリリース