Red Hat Directory Server 12

Red Hat Directory Server のパフォーマンスチューニング

サーバーおよびデータベースのパフォーマンスを改善するためのヒント

Last Updated: 2022-12-20

Red Hat Directory Server 12 Red Hat Directory Server のパフォーマンスチューニング

サーバーおよびデータベースのパフォーマンスを改善するためのヒント

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

本書では、Directory Server を手動で調整する方法を説明します。デフォルト設定を使用する際にパフォーマンスが最適でない場合は、この手順を適用します。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

弊社ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点はございませんか。改善点を報告する場合は、以下のように行います。

- 特定の文章に簡単なコメントを記入する場合は、以下の手順を行います。
  1. ドキュメントの表示が **Multi-page HTML** 形式になっていていることを確認してください。ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
  2. マウスカーソルを使用して、コメントを追加するテキストの部分を強調表示します。
  3. 強調表示されたテキストの下に表示される **Add Feedback** ポップアップをクリックします。
  4. 表示される指示に従ってください。
- より詳細なフィードバックをお寄せいただく場合は、Bugzilla のチケットを作成してください。
  1. Bugzilla の Web サイトに移動します。
  2. Component (コンポーネント) として **Documentation** を使用します。
  3. **Description** フィールドに、ドキュメントの改善に向けたご提案を記入してください。ドキュメントの該当部分へのリンクも追加してください。
  4. **Submit Bug** をクリックします。

第1章 データベースアクティビティーの監視

管理者は、データベースアクティビティーを監視して、キャッシュなどのチューニング設定が適切に設定されていることを確認する必要があります。

1.1. 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

コマンドラインを使用して監視アクティビティーを表示するには、**cn=monitor,cn=database_name,cn=ldbm database,cn=plugins,cn=config** に格納されている動的に更新された読み取り専用属性を表示します。

手順

- データベースの現在のアクティビティーを表示するには、次のように入力します。

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

このコマンドは、**userRoot** データベースのアクティビティーを表示します。

関連情報

- データベース監視属性

1.2. 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 コンソールでは、Directory Server は、監視タブの **cn=monitor,cn=database_name,cn=ldbm database,cn=plugins,cn=config** から動的に更新された読み取り専用監視属性の値を表示します。

手順

1. **Monitoring → Database → database name** に移動します。
2. **Entry Cache** タブと **DN Cache** タブにキャッシュ値を表示します。

関連情報

- データベース監視属性

1.3. データベース監視属性

表1.1 継承の設定

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、サーバーがデータベースからエントリーをリロードせずにエントリーキャッシュからエントリーを取得できた合計回数です。

属性	説明
entrycachetries	インスタンスを開始してからのエントリーキャッシュルックアップの総数。値は総数です。インスタンスが開始されたため、Directory Server 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の数は、エントリーキャッシュのルックアップを成功させようとします。この数は、インスタンスを最後に開始してからのルックアップとヒットの合計に基づいています。エントリーキャッシュのヒット率が 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を見つけようとするときはいつでも、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、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。
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、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** 属性の値を増やします。
maxentrycachesize	Directory Server がエントリーキャッシュに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、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。 この値を調整するには、**cn=database_name,cn=ldbm database,cn=plugins,cn=config** エントリーの **nsslapd-cachesize** 属性の値を増やします。
dncachehits	サーバーが、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) を取得することにより、要求を処理できた回数。

属性	説明
dncachetries	インスタンスを開始してからの DN キャッシュアクセスの総数。
dncachehitratio	キャッシュの比率は、DN キャッシュヒットの成功を試みます。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、データベースの **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	Directory Server が DN キャッシュに保持できる 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、データベースの **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。

第2章 ビューのパフォーマンスの改善

ビューベースの階層のパフォーマンスは、階層自体の構造とディレクトリーツリー (DIT) のエントリー数に依存します。

一般的には、仮想 DIT ビューを使用する場合に、(標準の DIT での同等の検索に対して数パーセントポイント以内で) パフォーマンスに関して若干の変化が発生する可能性があります。検索でビューを呼び出さない場合は、パフォーマンスへの影響はありません。導入の前に、予想される検索パターンおよび負荷に対して仮想 DIT ビューをテストします。

組織内でビューを汎用ナビゲーションツールとして使用する場合は、ビューフィルターで使用される属性をインデックス化することが推奨されます。

さらに、ビューでサブフィルターの評価に使用する仮想リストビュー (VLV) インデックスを設定できます。

特にビュー用に、ディレクトリーの他の部分をチューニングする必要はありません。

2.1. コマンドラインを使用してビューのパフォーマンスを改善するためのインデックスの作成

ビューは指定のフィルターに基づいて検索結果から派生します。フィルターの一部は **nsViewFilter** で明示的に指定される属性です。フィルターの残りの部分はエントリー階層に基づいており、ビューに含まれる実際のエントリーの **entryid** および **parentid** 操作属性を検索します。

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

検索された属性 (**entryid**、**parentid**、または **nsViewFilter** の属性) のいずれかにインデックスが付けられていない場合、検索は部分的にインデックスが解除され、Directory Server はディレクトリーツリー全体で一致するエントリーを検索します。

ビューのパフォーマンスを改善するには、以下のようにインデックスを作成します。

- **entryid** の**等式インデックス** (**eq**) を作成します。**parentid** 属性は、デフォルトでシステムインデックスでインデックス化されます。
- **nsViewFilter** テストにフィルターがある場合 (**attribute=***) は、テスト中の属性について、**存在インデックス**(**pres**) を作成します。このインデックスタイプは、少数のディレクトリーエントリーに表示される属性でのみ使用する必要があります。
- **nsViewFilter** テスト等価 (**attribute=value**) のフィルターの場合、テストする属性に対して **等価インデックス** (**eq**) を作成します。
- **nsViewFilter** のフィルターがサブ文字列をテストする場合 (**attribute=value***) は、テストする属性の substring index (**sub**) を作成します。
- **nsViewFilter** でフィルターが近似値をテストする場合 (**attribute~=value**)、テストされている属性のapproximate index (**approximate**) を作成します。

たとえば、以下の view フィルターを使用する場合は、以下を実行します。

```
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=*66))
```

デフォルトで行われる**等価インデックス**で **objectClass** にインデックスを付け、**部分文字列インデックス**で **roomNumber** にインデックスを付ける必要があります。

前提条件

- ビューフィルターで使用する属性に注意してください。

手順

1. オプション: バックエンドを一覧表示し、データベースをインデックス化します。

```
# dsconf -D "cn=Directory Manager" instance_name backend suffix list
dc=example,dc=com (userroot)
```

   (括弧で) 選択したデータベース名を書き留めておきます。

2. 選択したバックエンドのデータベースの **dsconfig** ユーティリティーを使用してインデックス設定を作成します。
   特に国際化されたインスタンスの場合、属性名、インデックスタイプと、任意で照合順序 (OID) を設定するためのマッチングルールを指定します。

```
# dsconf -D "cn=Directory Manager" instance_name backend index add --attr
roomNumber --index-type sub userroot
```

   view フィルターで使用される属性ごとに、このステップを繰り返します。

3. 新規インデックスを適用するためにデータベースを再インデックスします。

```
# dsconf -D "cn=Directory Manager" instance_name backend index reindex userroot
```

検証

1. ビューで使用するフィルターが同じ標準ディレクトリーツリーに基づく検索を実行します。

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
dc=example,dc=com (&(objectClass=inetOrgPerson)(roomNumber=*66))
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
dc=example,dc=com "(&(objectClass=inetOrgPerson)(roomNumber=*66))"
```

2. **/var/log/dirsrv/slapd-instance_name/access**.の**アクセス**ログを見る。
   検索の **RESULT** には **note=U** または **Partially Unindexed Filter** が詳細に含まれていないはずです。

関連情報

- インデックスの管理

2.2. WEB コンソールを使用してビューのパフォーマンスを改善するためのインデックスの作成

ビューは指定のフィルターに基づいて検索結果から派生します。フィルターの一部は **nsViewFilter** で明示的に指定される属性です。フィルターの残りの部分はエントリー階層に基づいており、ビューに含まれる実際のエントリーの **entryid** および **parentid** 操作属性を検索します。

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

検索された属性 (**entryid**、**parentid**、または **nsViewFilter** の属性) のいずれかにインデックスが付けられていない場合、検索は部分的にインデックスが解除され、Directory Server はディレクトリーツリー全体で一致するエントリーを検索します。

ビューのパフォーマンスを改善するには、以下のようにインデックスを作成します。

- **entryid** の**等式インデックス** (**eq**) を作成します。**parentid** 属性は、デフォルトでシステムインデックスでインデックス化されます。
- **nsViewFilter** テストにフィルターがある場合 (**attribute=***) は、テスト中の属性について、**存在インデックス**(**pres**) を作成します。このインデックスタイプは、少数のディレクトリーエントリーに表示される属性でのみ使用する必要があります。
- **nsViewFilter** テスト等価 (**attribute=value**) のフィルターの場合、テストする属性に対して **等価インデックス** (**eq**) を作成します。
- **nsViewFilter** のフィルターがサブ文字列をテストする場合 (**attribute=value***) は、テストする属性の substring index (**sub**) を作成します。
- **nsViewFilter** でフィルターが近似値をテストする場合 (**attribute~=value**)、テストされている属性のapproximate index (**approximate**) を作成します。

たとえば、以下の view フィルターを使用する場合は、以下を実行します。

```
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=*66))
```

デフォルトで行われる**等価インデックス**で **objectClass** にインデックスを付け、**部分文字列インデックス**で **roomNumber** にインデックスを付ける必要があります。

前提条件

- Web コンソールでインスタンスにログインしている。
- ビューフィルターで使用する属性に注意してください。

手順

1. **Database** で、インデックスを作成する設定ツリーから接尾辞を選択します。
2. **Indexes** および **Database Indexes** に移動します。
3. **Add Index** ボタンをクリックします。
4. 属性の名前を入力し、属性を選択します。
5. この属性に対して作成する必要のある **Index Types** を選択します。
6. 必要に応じて、特に国際化されたインスタンスの場合に、照合順序 (OID) を指定するための **Matching Rules** を追加します。
7. **Index attribute after creation** を選択し、後でインデックスを再ビルドします。
8. **Create Index** をクリックします。
9. インデックス化される各属性に対して手順を繰り返します。

検証

- 追加された属性の名前を入力して、**Filter Indexes** します。
- 新たにインデックス化された属性が結果に表示されるはずです。

関連情報

- インデックスの管理

第3章 ID の長いリストをロードするときのパフォーマンスを向上させるためにインデックススキャン制限を設定する

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。たとえば、**inetorgperson** 属性を備えた 100 万のエントリーを持つディレクトリーは、**(objectclass=inetorgperson)** などのフィルターを使用した検索でこれらすべてのエントリーを返します。

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。ID リストのスキャン制限は、キーがプライマリーインデックス全体と一致すると見なされる前に、ディレクトリーサーバーが読み取る ID の数に制限を設定します。これは、ディレクトリーサーバーが検索を、異なるリソース制限のセットを持つインデックスなしの検索として扱うことを意味します。

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、ディレクトリーとディレクトリー自体のサイズに近いサイズのインデックスを検索するのではなく、結果を処理するためにディレクトリー全体を 1 か所だけ検索する必要があります。

インデックススキャンの制限は、グローバルに、または特定のデータベースに対して設定できます。

3.1. コマンドラインを使用したグローバルインデックススキャン制限の設定

デフォルトでは、ディレクトリーサーバーの ID リストのスキャン制限は **4000** です。ほとんどのシナリオでは、この値は一般的な範囲のデータベースサイズとアクセスパターンに対して良好なパフォーマンスを提供するため、デフォルト値を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 エントリーよりわずかに多くても、ディレクトリー全体よりもかなり小さい場合は、ID リストのスキャン制限を上げると検索が改善されます。

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。

手順

1. ID リストのスキャン制限を更新します。

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

   このコマンドは、制限を **8000** エントリーに設定します。

2. インスタンスを再起動します。

   ```
   # dsctl instance_name restart
   ```

3.2. WEB コンソールを使用したグローバルインデックススキャン制限の設定

デフォルトでは、ディレクトリーサーバーの ID リストのスキャン制限は **4000** です。ほとんどのシナリオでは、この値は一般的な範囲のデータベースサイズとアクセスパターンに対して良好なパフォーマンスを提供するため、デフォルト値を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 エントリーよりわずかに多くても、ディレクトリー全体よりもかなり小さい場合は、ID リストのスキャン制限を上げると検索が改善されます。

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。

手順

1. **Database → Global Database Configuration** に移動します。
2. **ID List Scan Limit** フィールドを更新します。
3. **Save Config** をクリックします。
4. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

3.3. コマンドラインを使用してデータベースにインデックススキャン制限を設定する

場合によっては、特定のインデックスに制限を定義したり、ID リストをまったく使用しない方が便利な場合があります。さまざまなタイプの検索フィルターの ID リストスキャン制限を個別に設定できます。

たとえば、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))** フィルターはまず **objectClass=inetOrgPerson** に一致する 1,000 万の ID すべてを含む ID リストを作成します。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に制限を定義するか、ID リストをまったく使用しないようにすると便利です。

この手順では、**AND** 句で **objectClass=inetOrgPerson** 条件の ID リストを作成しないようにディレクトリーサーバーを設定する方法を示します。

手順

- **nsIndexIDListScanLimit** パラメーターを設定します。

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://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
```

これらの設定では、ディレクトリーサーバーは **AND** 句の **objectClass=inetOrgPerson** 条件の ID リストを作成しません。その他のすべての状況では、ディレクトリーサーバーはグローバル ID リストのスキャン制限値を適用します。

nsIndexIDListScanLimit パラメーターは、次の構文を使用します。

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

  - **limit**: ID リストの最大サイズを設定します。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 32 ビット整数の最大値 (**2147483647**): ID の最大数

- **type**: オプション: スキャン制限の動作を変更するフラグを設定します。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。
- **values**: オプション: 制限を適用するために検索フィルターに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してください。値は、インデックスタイプと、インデックスが適用される属性の構文に対応している必要があります。たとえば、整数ベースの属性 **uidNumber** を指定し、それが **eq** タイプに対してインデックス付けされている場合、**type=eq values=abc** は使用できません。

  値にスペース、コンマ、NULL、またはその他のエスケープが必要な値が含まれる場合、LDAP フィルタエスケープ構文を使用します。バックスラッシュ (\) の後に、文字の 2 桁の 16 進数コードを続けます。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
  ```

第4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。

サーバーが使用可能なロックを使い果たした場合、ディレクトリーサーバーは次のエラーを **/var/log/dirsrv/slapd-instance_name/errors** ファイルに記録します。

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

ただし、ディレクトリーサーバーのデフォルト設定では、サーバーがロックを使い果たしないようにして、データの破損を回避しようとします。詳細は、Avoiding data corruption by monitoring free database locks を参照してください。

4.1. 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。

この手順により、間隔が **600** ミリ秒に変更され、しきい値が **85** % に変更されます。

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。

手順

1. 間隔としきい値を設定します。

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

2. インスタンスを再起動します。

```
# dsctl instance_name restart
```

検証

- ロックモニタリング設定を表示します。

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com backend config get | grep "nsslapd-db-locks-monitoring"
nsslapd-db-locks-monitoring-enabled: on
nsslapd-db-locks-monitoring-threshold: 85
nsslapd-db-locks-monitoring-pause: 600
```

4.2. ロック数の手動監視

ディレクトリーサーバーは、**cn=database,cn=monitor,cn=ldbm database,cn=plugins,cn=config** の **nsslapd-db-current-locks** および **nsslapd-db-max-locks** 属性で現在のロック数を追跡します。

手順

- ロックの数を表示するには、次のように入力します。

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://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
...
nsslapd-db-current-locks: 37
nsslapd-db-max-locks: 39
```

4.3. コマンドラインを使用したロック数の設定

dsconf backend config set コマンドを使用して、ディレクトリーサーバーが使用できるロックの数を更新します。

手順

1. ロックの数を設定します。

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

   このコマンドは、ロック数を **20000** に設定します。

2. インスタンスを再起動します。

```
# dsctl instance_name restart
```

検証

- **nsslapd-db-locks** パラメーターの値を表示します。

```
# dsconf -D "cn=Directory Manager" ldap://supplier.example.com backend config get |
grep "nsslapd-db-locks:"
nsslapd-db-locks: 20000
```

4.4. WEB コンソールを使用したロック数の設定

ディレクトリーサーバーが使用するロックの数は、Web コンソールのグローバルデータベース設定で設定できます。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. Database → Global Database Configuration に移動します。
2. **Show Advanced Settings** をクリックします。
3. **Enable Monitoring** を選択し、しきい値のパーセンテージと一時停止ミリ秒を入力します。
4. **Save Config** をクリックします。
5. Actions → Restart Instance をクリックします。

検証

1. Database → Global Database Configuration に移動します。
2. **Show Advanced Settings** をクリックします。
3. ロックモニタリングの設定を確認してください。

第5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、ディレクトリーサーバーは自動チューニング設定を使用します。この設定では、サーバーは使用可能な CPU スレッドと同じ数のディレクトリーサーバースレッドを最大 512 のディレクトリーサーバースレッドまで使用します。

注記

Red Hat は、自動チューニング設定を使用することを推奨しています。スレッド数は手動で設定しないでください。

5.1. コマンドラインを使用したスレッド数の設定

特定の状況では、ディレクトリーサーバースレッドの固定数を手動で設定する必要があります。たとえば、自動チューニング設定を使用せずに仮想マシンの CPU コア数を変更した場合、ディレクトリーサーバースレッドの数を調整するとパフォーマンスが向上する可能性があります。

以前に特定の数のスレッドを設定した場合は、この手順を使用して自動チューニング設定を再度有効にすることもできます。

手順

1. ディレクトリーサーバーが使用するスレッド数を設定します。

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

   自動チューニング設定を有効にするには、**nsslapd-threadnumber** パラメーターを **-1** に設定します。

2. インスタンスを再起動します。

```
# dsctl instance_name restart
```

5.2. WEB コンソールを使用したスレッド数の設定

特定の状況では、ディレクトリーサーバースレッドの固定数を手動で設定する必要があります。たとえば、自動チューニング設定を使用せずに仮想マシンの CPU コア数を変更した場合、ディレクトリーサーバースレッドの数を調整するとパフォーマンスが向上する可能性があります。

以前に特定の数のスレッドを設定した場合、Web コンソールを使用して自動チューニング設定を再度有効にすることはできないことに注意してください。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Server → Tuning & Limits** に移動します。
2. **Number Of Worker Threads** フィールドにスレッド数を設定します。
3. **Save settings** をクリックします。
4. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

第6章 リソース制限のチューニング

ディレクトリーサーバーには、インスタンスが使用するリソースの量をチューニングするための設定がいくつか用意されています。コマンドラインまたは Web コンソールを使用して変更できます。

6.1. コマンドラインを使用したリソース制限設定の更新

このセクションでは、リソース制限の設定を変更する一般的な手順について説明します。環境に合わせて設定を調整してください。

手順

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=value
```

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**: 検索の時間制限を設定します。
- **nsslapd-sizelimit**: 検索サイズの制限を設定します。
- **nsslapd-pagedsizelimit**: ページ検索のサイズ制限を設定します。
- **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、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**: Basic Encoding Rules (BER) の最大サイズを設定します。
- **nsslapd-maxsasliosize**: Simple Authentication and Security Layer (SASL) の最大 I/O サイズを設定します。
- **nsslapd-listen-backlog-size**: 受信接続の受信に使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**: 仮想属性ルックアップ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。
  詳細は、Configuration and schema reference のパラメーターの説明を参照してください。

2. インスタンスを再起動します。

```
# dsctl instance_name restart
```

6.2. WEB コンソールを使用したリソース制限設定の更新

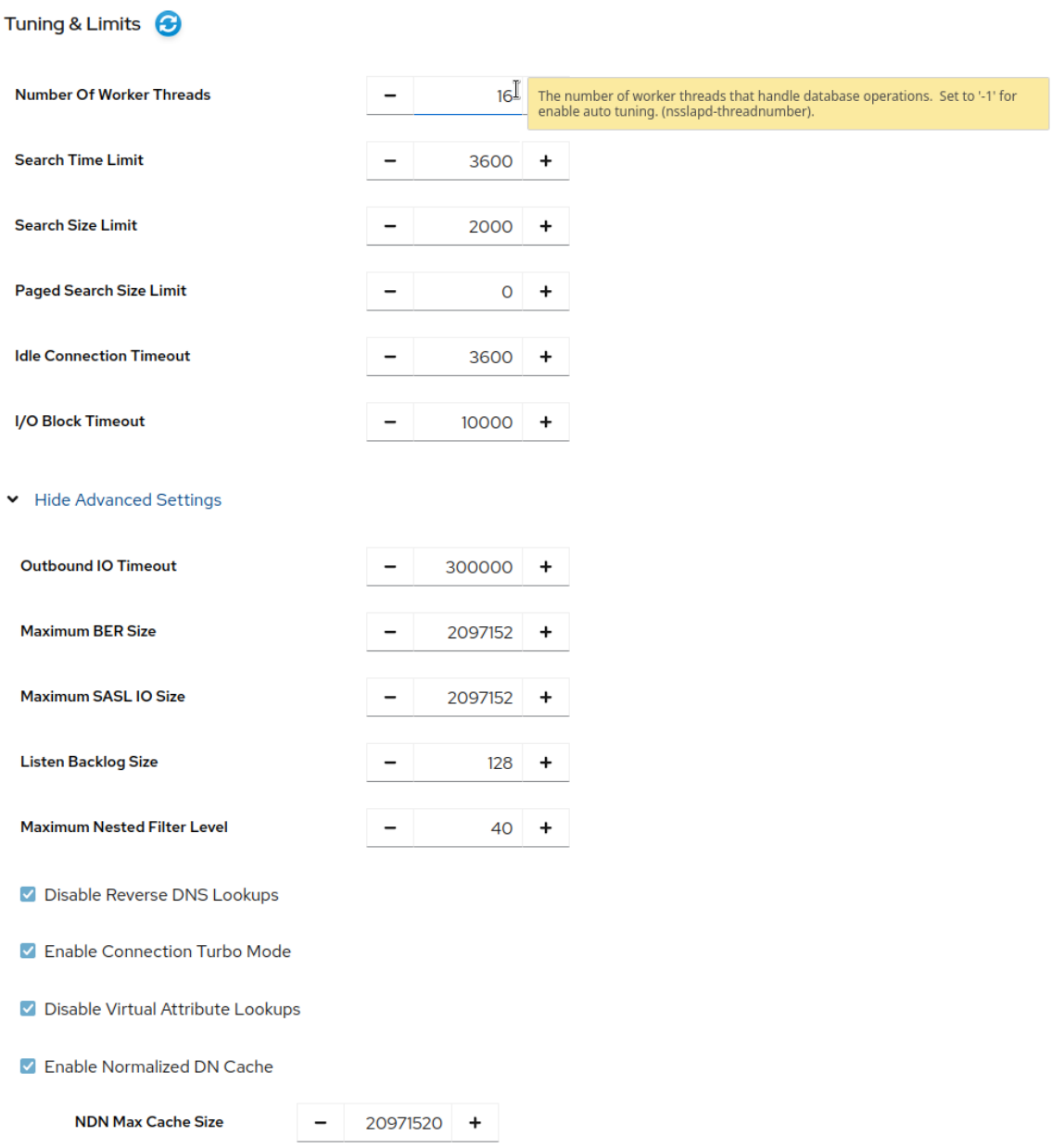
このセクションでは、リソース制限の設定を変更する一般的な手順について説明します。環境に合わせて設定を調整してください。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Server → Tuning & Limits** に移動します。
2. 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。



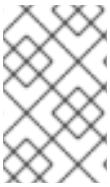
3. **Save settings** をクリックします。

4. **Actions → Restart Instance** をクリックします。

6.3. TRANSPARENT HUGE PAGES 機能の無効化

Transparent Huge Pages (THP) は Linux のメモリー管理機能で、より大きなメモリーページを使用することで、大量のメモリーを搭載したマシンでの Translation Lookaside Buffer (TLB) チェックのオーバーヘッドを削減します。THP 機能は、RHEL システムではデフォルトで有効になっており、2 MB のメモリーページをサポートしています。

ただし、THP 機能は、大規模で連続した割り当てパターンで有効にすると最適に機能し、Red Hatディレクトリーサーバーに典型的な小規模でまばらな割り当てパターンではパフォーマンスが低下する可能性があります。プロセスの常駐メモリーサイズは、最終的に制限を超えてパフォーマンスに影響を与えるか、メモリー不足 (OOM) キラーによって終了する可能性があります。



注記

パフォーマンスとメモリー消費の問題を回避するために、Red Hat ディレクトリーサーバーがインストールされている RHEL システムでは THP を無効にすることをお勧めします。

手順

1. 次のコマンドを実行して、Transparent huge page の現在のステータスを確認します。

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
```

2. 透過的なヒュージページ機能が有効になっている場合は、起動時または実行時に無効にします。

   - **grub.conf** ファイルのカーネルコマンドラインに以下を追加して、ブート時に透過的な Huge Page を無効にします。

```
transparent_hugepage=never
```

   - 次のコマンドを実行して、実行時に透過的な Huge Page を無効にします。

```
# echo never > /sys/kernel/mm/transparent_hugepage/enabled
# echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

関連情報

- The negative effects of Transparent Huge Pages (THP) on RHDS 記事
- Configuring Transparent Huge Pages.

第7章 ロギングのパフォーマンスを向上させるためにアクセスログのバッファーリングを無効にする

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。

注記

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。

7.1. 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

アクセスログのバッファーリングを無効にすると、ディレクトリーサーバーはログエントリーをディスクに直接書き込みます。これにより、大規模なデプロイメントでのパフォーマンスが向上します。

手順

1. アクセスログのバッファーリングを無効にするには、次のように入力します。

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

2. インスタンスを再起動します。

```
# dsctl instance_name restart
```

検証

1. 連続モードでアクセスログを表示します。

```
# tail -f /var/log/dirsrv/slapd-instance_name/access
```

2. 検索など、ディレクトリー内でアクションを実行します。
3. アクセスログをモニターします。ユーザーがディレクトリーでアクションを実行すると、ログエントリーが遅延なく表示されます。

7.2. 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

アクセスログのバッファーリングを無効にすると、ディレクトリーサーバーはログエントリーをディスクに直接書き込みます。これにより、大規模なデプロイメントでのパフォーマンスが向上します。

手順

1. **Server → Logging → Access Log → Settings** に移動します。
2. **Access Log Buffering Enabled** の選択を解除します。
3. **Save Log Settings** をクリックします。

4. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

検証

1. Monitoring → Logging → Access Log に移動します。
2. **Continuously Refresh** を選択します。
3. 検索など、ディレクトリー内でアクションを実行します。
4. アクセスログをモニターします。ユーザーがディレクトリーでアクションを実行すると、ログエントリーが遅延なく表示されます。

第8章 ローカルディスクをモニターして、ディスク容量が少ないときにディレクトリーサーバーをシャットダウンする

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、ディレクトリーサーバーを設定して、設定、トランザクションログ、および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ディスク領域をモニターします。空き容量が設定されたしきい値に達すると、ディレクトリーサーバーはインスタンスをシャットダウンします。

8.1. 空きディスク容量に応じたディレクトリーサーバーの動作

監視を設定するときのディレクトリーサーバーの動作は、残りの空き領域の量によって異なります。

- 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。

8.2. 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

ディレクトリーサーバーは、設定、トランザクションログ、および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ディスク容量をモニターできます。残りの空き容量に応じて、ディレクトリーサーバーは特定のログ機能を無効にするかシャットダウンします。

手順

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=3221225472 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 (3,221,225,472 バイト) に設定し、猶予期間を 60 秒に設定します。

2. オプション: アクセスロギングを無効にしたり、アーカイブログを削除したりしないようにディレクトリーサーバーを設定します。

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

3. インスタンスを再起動します。

```
# dsctl instance_name restart
```

8.3. WEB コンソールを使用したローカルディスク監視の設定

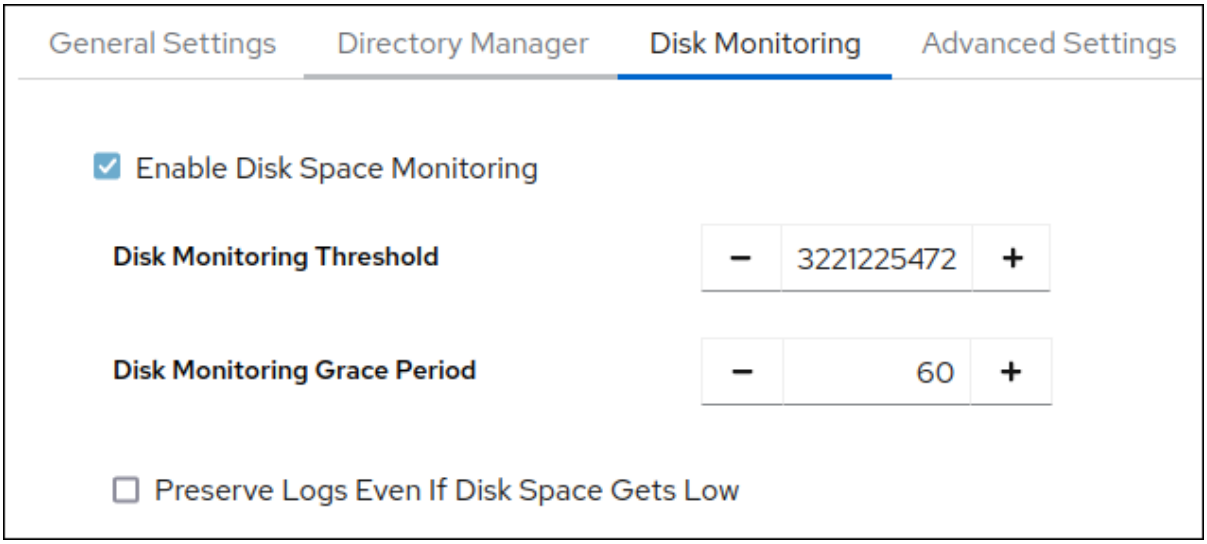
ディレクトリーサーバーは、設定、トランザクションログ、およびデータベースディレクトリーを含むファイルシステムの空きディスク容量をモニターできます。残りの空き容量に応じて、ディレクトリーサーバーは特定のログ機能を無効にするかシャットダウンします。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Server → Server Settings → Disk Monitoring** に移動します。
2. **Enable Disk Space Monitoring** を有効にするを選択します。しきい値をバイト単位で、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。

3. オプション: **Preserve Logs Even If Disk Space Gets Low** を選択します
4. **Save settings** をクリックします。
5. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

第9章 トランザクションログのチューニング

すべてのディレクトリーサーバーインスタンスには、管理するデータベースの更新を記録するトランザクションログが含まれています。変更操作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、エントリーを含むデータベースファイル内のエントリーレコードの更新と、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。この all-or-nothing アプローチにより、更新操作が atomic であることが保証されます。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。

ディレクトリーサーバーは定期的に、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のトリミングが必要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、サーバ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ロギング、データベースのフラッシュ、トリミング、およびデータベースリカバリーは介入を必要としない自動プロセスですが、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してパフォーマンスを最適化することをお勧めします。

9.1. 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

ディレクトリーサーバーは定期的に、トランザクションログに記録されたトランザクションをデータベー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にチェックポイントエントリーを記録します。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、ディレクトリーサーバーは 60 秒ごとにデータベーストランザクションログにチェックポイントエントリーを送信します。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、異常シャットダウン後にディレクトリーデータベースを回復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可能性もあります。

手順

- たとえば、チェックポイント間隔を 120 秒に変更します。

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

9.2. 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

ディレクトリーサーバーは定期的に、トランザクションログに記録されたトランザクションをデータベー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にチェックポイントエントリーを記録します。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、ディレクトリーサーバーは 60 秒ごとにデータベーストランザクションログにチェックポイントエントリーを送信します。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、異常シャットダウン後にディレクト

リーデータベースを回復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可能性もあります。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database → Global Database Configuration** に移動します。
2. **Show Advanced Settings** をクリックします。
3. **Database Checkpoint Interval** フィールドの値を更新します。
4. **Save Configuration** をクリックします。

9.3. 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設定される各 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更新で設定できますが、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オフにすると、ディレクトリーサーバーの書き込みパフォーマンスが向上する可能性がありますが、データ損失のリスクがあります。

永続トランザクションログを無効にすると、ディレクトリーサーバーはすべてのディレクトリーデータベース操作をデータベーストランザクションログファイルに書き込みますが、物理的にすぐにディスクに書き込まれない場合があります。ディレクトリー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。

ディレクトリーサーバーが実行中の場合は、**nsslapd-db-durable-transaction** パラメーターを変更できないことに注意してください。

手順

1. 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. インスタンスを起動します。

```
# dsctl instance_name start
```

第10章 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。
- 識別名 (DN) キャッシュは、DN と相対識別名 (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** ファイルを含む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

10.1. CACHE-AUTOSIZE および CACHE-AUTOSIZE-SPLIT パラメーターがデータベースとエントリーのキャッシュサイズに与える影響

デフォルトでは、ディレクトリーサーバーは自動サイズ変更機能を使用して、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上のデータベースとエントリーキャッシュの両方のサイズを最適化します。

重要

Red Hat は、自動サイズ設定機能を使用し、キャッシュサイズを手動で設定しないことをお勧めします。

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

これらの設定では、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。自動サイズ設定が有効である:

- データベースとエントリーキャッシュの両方で、**nsslapd-cache-autosize parameter** が **0** より大きい値に設定されている場合。
- データベースキャッシュで、**nsslapd-cache-autosize** 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で、**nsslapd-cache-autosize** 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、ディレクトリーサーバーがデータベースキャッシュに使用する RAM の割合を設定します。サーバーは残りのパーセンテージをエントリーキャッシュに使用します。
- 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。

デフォルトでは、ディレクトリーサーバーは次の設定を使用します。

- **nsslapd-cache-autosize: 10**

- **nsslapd-cache-autosize-split: 40**

これらの設定を使用すると、システムの空き RAM の 10% が使用されます (**nsslapd-cache-autosize**)。このメモリーのうち、サーバーは 40% をデータベースキャッシュ (**nsslapd-cache-autosize-split**) に使用し、残りの 60% をエントリーキャッシュに使用します。

空き RAM に応じて、以下のキャッシュサイズになります。

表10.1 nsslapd-cache-autosize と nsslapd-cache-autosize-split の両方がデフォルト値を使用する場合のキャッシュサイズ

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	1,126 MB
32 GB	512 MB	2,764 MB
64 GB	512 MB	6,042 MB
128 GB	512 MB	12,596 MB

RAM の空き容量が 16 GB 以上の場合、Directory Server は **nsslapd-dbcachesize** パラメーターに 512 MB の制限を適用します。

10.2. 必要なキャッシュサイズ

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: 2022-02-24 10:25:16
--------------------------------------------------------
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変更ログデータベースを含む **/var/lib/dirsrv/slapd-instance_name/db/** ディレクトリー内のすべての ***.db** 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、Setting the database cache size using the command line 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、Setting the DN cache size using the command line 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、Setting the entry cache size using the command line を参照してください。

10.3. 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ー量を増やします。

手順

1. Automatic Cache Tuning を無効にします。

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

2. データベースのキャッシュサイズを手動で設定します。

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

データベースのキャッシュサイズをバイト単位で指定します。この例では、コマンドはデータベースキャッシュを 256 MB に設定します。

3. インスタンスを再起動します。

```
# dsctl instance_name restart
```

10.4. 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ー量を増やします。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database → Global Database Configuration** に移動します。
2. **Automatic Cache Tuning** の選択を解除します。
3. **Save Config** をクリックします。
4. 256 MB の場合は **268435456** のように、データベースキャッシュサイズをバイト単位で **Database Cache Size** フィールドに入力します。
5. **Save Config** をクリックします。
6. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

10.5. コマンドラインを使用した DN キャッシュサイズの設定

ディレクトリーサーバーは **entryrdn** インデックスを使用して、識別名 (DN) および相対識別名 (RDN) をエントリーに関連付けます。これにより、サーバーは効率的にサブツリーの名前を変更し、エントリーを移動し、**moddn** 操作を実行できます。サーバーは DN キャッシュを使用して **entryrdn** インデックスのメモリー内表現をキャッシュし、コストのかかるファイル I/O および変換操作を回避します。

自動チューニング機能を使用しない場合、特にエントリーの名前変更と移動操作で最高のパフォーマンスを得るには、DN キャッシュを、ディレクトリーサーバー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。

手順

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. 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. インスタンスを再起動します。

```
# dsctl instance_name restart
```

10.6. WEB コンソールを使用した DN キャッシュサイズの設定

ディレクトリーサーバーは **entryrdn** インデックスを使用して、識別名 (DN) および相対識別名 (RDN) をエントリーに関連付けます。これにより、サーバーは効率的にサブツリーの名前を変更し、エントリーを移動し、**moddn** 操作を実行できます。サーバーは DN キャッシュを使用して **entryrdn** インデックスのメモリー内表現をキャッシュし、コストのかかるファイル I/O および変換操作を回避します。

自動チューニング機能を使用しない場合、特にエントリーの名前変更と移動操作で最高のパフォーマンスを得るには、DN キャッシュを、ディレクトリーサーバー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。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database → Suffixes → suffix_name** に移動します。
2. **DN Cache Size** フィールドに DN キャッシュサイズをバイト単位で入力します。
3. **Save Configuration** をクリックします。
4. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

10.7. 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設定

ディレクトリーサーバーはエントリーキャッシュを使用して、検索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格納します。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、識別名 (DN) をディスク上のフォーマットからメモリー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。

手順

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バイト単位で設定します。

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

このコマンドは、**userRoot** データベースのエントリーキャッシュを 2 GB に設定します。

4. インスタンスを再起動します。

```
# dsctl instance_name restart
```

10.8. 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設定

ディレクトリーサーバーはエントリーキャッシュを使用して、検索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格納します。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、識別名 (DN) をディスク上のフォーマットからメモリー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。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database → Suffixes → suffix_name → Settings** に移動します。
2. **Automatic Cache Tuning** 設定を無効にします。
3. **Save Configuration** をクリックします。
4. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。
5. **Database → Suffixes → suffix_name → Settings** に移動します。
6. Entry Cache Size 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。
7. **Save Configuration** をクリックします。
8. 右上隅の **Actions** をクリックし、**Restart Instance** を選択します。

第11章 インポートパフォーマンスの向上

非常に大きな属性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。このセクションでは、Directory Server の設定を調整してインポートパフォーマンスを向上させる方法について説明します。

11.1. 大きなデータベースのインポートおよび大きな属性値を持つインポートに対する DIRECTORY SERVER のチューニング

次のような操作には、インポートキャッシュの自動サイズ設定機能を使用します。

- 非常に大きなデータベースのインポート
- 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の大きな属性を持つデータベースのインポート

注記

オフラインインポートは、オンラインインポートよりも高速です。可能であれば、オフラインインポートの使用を検討してください。

nsslapd-import-cache-autosize 属性を使用して、インポートキャッシュの自動サイズ設定機能を設定できます。デフォルトでは、Directory Server は **ldif2db** 操作に対してのみインポートキャッシュの自動サイズ設定を有効にし、空き物理メモリーの 50% をインポートキャッシュに自動的に割り当てます。

詳細については、「設定、コマンド、およびファイルリファレンス」ドキュメントで nsslapd-import-cache-autosize 属性の説明を参照してください。