NetIQ® eDirectory™
チューニングガイド

2019年9月

保証と著作権

保証と著作権、商標、免責事項、保証、輸出およびその他の使用制限、米国政府の規制による権利、特許ポリシー、およびFIPSコンプライアンスの詳細については、https://www.netiq.com/company/legal/を参照してください。

本書およびライブラリについて

展開全体で優れたパフォーマンスを引き出すために、NetIQ eDirectory (eDirectory)製品を分析し、チューニングする方法について説明します。

『*NetIQ eDirectory 9.1 チューニングガイド*』の最新版については、NetIQ eDirectoryオンラインヘルプのWebサイトを参照してください。

本書の読者

このガイドはネットワーク管理者を対象としています。

ライブラリに含まれているその他の情報

ライブラリには次の情報リソースが含まれています。

XDASv2管理ガイド

eDirectoryとNetIQ Identity Managerを監査するためのXDASv2の設定と使用方法について説明します。

インストールガイド

eDirectoryのインストール方法について説明します。ネットワーク管理者を対象としています。

管理ガイド

eDirectoryの管理および設定方法について説明します。

トラブルシューティングガイド

eDirectoryの問題を解決する方法について説明します。

新機能ガイド

eDirectoryの新機能について説明します。

これらのガイドは、NetIQ eDirectory documentationのWebサイトで入手できます。

eDirectory管理ユーティリティの詳細については、『*NetIQ iManager 3.1 Administration Guide*』を参照してください。

NetIQ社について

当社はグローバルなエンタープライズソフトウェア企業であり、お客様の環境において絶えず挑戦となる変化、複雑さ、リスクという3つの要素に焦点を当て、それらをお客様が制御するためにどのようにサポートできるかを常に検討しています。

当社の観点

変化に適応すること、複雑さとリスクを管理することは普遍の課題

実際、直面するあらゆる課題の中で、これらは、物理環境、仮想環境、およびクラウドコンピューティング環境の安全な評価、監視、および管理を行うために必要な制御を脅かす最大の要因かもしれません。

重要なビジネスサービスの改善と高速化を可能にする

当社は、IT組織に可能な限りの制御能力を付与することが、よりタイムリーでコスト効率の高いサービス提供を実現する唯一の方法だと信じています。組織が継続的な変化を遂げ、組織を管理するために必要なテクノロジが実質的に複雑さを増していくにつれ、変化と複雑さという圧力はこれからも増え続けていくことでしょう。

当社の理念

単なるソフトウェアではなく、インテリジェントなソリューションを販売する

確かな制御手段を提供するために、まずお客様のIT組織が日々従事している現実のシナリオを把握することに努めます。そのようにしてのみ、実証済みで測定可能な結果を成功裏に生み出す、現実的でインテリジェントなITソリューションを開発することができます。これは単にソフトウェアを販売するよりもはるかにやりがいのあることです。

当社の情熱はお客様の成功を推し進めること

お客様が成功するためにわたしたちには何ができるかということが、わたしたちのビジネスの核心にあります。製品の着想から展開まで、当社は次のことを念頭に置いています。お客様は既存資産とシームレスに連動して動作するITソリューションを必要としており、展開後も継続的なサポートとトレーニングを必要とし、変化を遂げるときにも共に働きやすいパートナーを必要としています。究極的に、お客様の成功こそがわたしたちの成功なのです。

当社のソリューション

- IDおよびアクセスのガバナンス
- アクセス管理
- セキュリティ管理
- システムおよびアプリケーション管理

- ワークロード管理
- サービス管理

セールスサポートへのお問い合わせ

製品、価格、および機能についてのご質問は、各地域のパートナーへお問い合わせください。パートナーに連絡できない場合は、弊社のセールスサポートチームへお問い合わせください。

各国共通:	www.netiq.com/about_netiq/officelocations.asp
米国およびカナダ:	1-888-323-6768
電子メール:	info@netiq.com
Webサイト:	www.netiq.com

テクニカルサポートへのお問い合わせ

特定の製品に関する問題については、弊社のテクニカルサポートチームへお問い合わせください。

各国共通:	www.netiq.com/support/contactinfo.asp
北米および南米:	1-713-418-5555
ヨーロッパ、中東、アフリカ:	+353 (0) 91-782 677
電子メール:	support@netiq.com
Webサイト:	www.netiq.com/support

マニュアルサポートへのお問い合わせ

弊社の目標は、お客様のニーズを満たすマニュアルの提供です。改善のためのご提案は、www.netiq.com/documentationに掲載されている本マニュアルのHTML版で、各ページの下にある［**コメントを追加**］をクリックしてください。Documentation-Feedback@netiq.com宛てに電子メールを送信することもできます。貴重なご意見をぜひお寄せください。

オンラインユーザコミュニティへのお問い合わせ

NetIQのオンラインコミュニティであるQmunityは、他のユーザやNetIQのエキスパートとやり取りできるコラボレーションネットワークです。より迅速な情報、有益なリソースへの役立つリンク、NetIQエキスパートとのやり取りを提供するQmunityは、頼みにしているIT投資が持つ可能性を余すことなく実現するために必要な知識の習得に役立ちます。詳細については、http://community.netiq.comを参照してください。

1 概要

NetIQ eDirectory 9.2はディレクトリサービスソリューションで、標準準拠、クロスプラットフォーム、ハイスケーラビリティ、耐故障性、およびハイパフォーマンスを特徴としています。このガイドは、お客様のeDirectory環境をチューニングし、パフォーマンスを向上させるための情報を提供します。

パフォーマンスのチューニングは複雑です。eDirectoryおよびオペレーティングシステムのサブシステムに関する理解が求められます。システムを監視しボトルネックを特定することやそれを一つずつ修正することも含まれています。多くの場合リソースは限られており、チューニングはeDirectoryとオペレーティングシステムに限定されています。

このガイドでは、チューニングを始める前にセクション「前提条件」読み、それから他のセクションへ進んでください。「eDirectoryのサブシステム」は、eDirectoryのパフォーマンスに影響を及ぼすプライマリサブシステムについて説明しています。「システムボトルネックの分析」は、システムリソースおよびeDirectoryのパフォーマンスに対するその影響について説明しています。「eDirectoryサブシステムのチューニング」は、様々な条件および環境下でeDirectoryを分析し、チューニングする方法について説明しています。最後に「eDirectory設定」では、チューニング可能パラメータの設定方法について説明しています。

前提条件

パフォーマンス向上のためのシステムチューニングを行う前に、次の一般的な前提条件が満たされていることを確認してください。

- eDirectoryのツリー設計が適切なら、eDirectoryのパフォーマンスを向上させることができます。次の点を考慮してください:
  - アプリケーションは要求チェーンを渡す必要なく、すべての情報をサーバ上でローカルに読み込みます。
  - eDirectoryは自動的に、オブジェクト参照を効率的に処理します。可能なら、サーバ上のオブジェクトはそのサーバ上のローカルではないオブジェクトを参照すべきではありません。非ローカルオブジェクト参照の維持にはより時間がかかるからです。そのような参照がある場合、バックリンクを維持する必要があります。これは、大規模な展開で煩雑になります。
  - 10,000人以上のメンバーを持つグループが必要な場合は、動的グループをお勧めします。こうすることで、多くの人のために参照を維持することによるオーバーヘッドを避けることができます。動的グループの設定は注意深く行ってください。不適切な検索条件で複数の動的グループを使用すると、サーバの過負荷やサーバパフォーマンス全体の低下につながりかねないからです。検索操作が完了するまでに長い時間がかかる場合は、選択したインデックスが適切でない可能性があります。通常の（静的）グループの使用はできるだけ控えてください。ログイン時のツリーウォーキングが増大する可能性があるからです。
  - ACLを効果的に使用してください。たとえば、権利を自身に割り当てるACLテンプレートを使用するのではなく、［This］トラスティを使用し、コンテナレベルで割り当ててください。ACLが少なければ少ないほど、パフォーマンスは向上します。ACLの詳細については、『*NetIQ eDirectory管理ガイド*』の「「eDirectoryでの権利」」を参照してください。
  - 複数のレプリカサーバに負荷を分散してください。

- 適切なツリー設計はツリーウォーキングを必要最小限に抑えますが、ときに必要なこともあります。『*NetIQ eDirectory管理ガイド*』の「「詳細参照コスト」」を考慮してください。
- ログインに時間がかかる場合、ログイン更新を無効にできます。NDSおよびNetIQModular Authentication Service (NMAS)ログインにはそれぞれ、ログイン更新を無効にする異なる方法があります。とはいえ、セキュリティへの影響 (http://www.novell.com/documentation/nmas33/admin/data/bg8dphs.html)を理解することが重要です。

- iMonitorによりヘルスチェックを実行してください。詳細については、『*NetIQ eDirectory管理ガイド*』の「「eDirectoryサーバヘルスの表示」」を参照してください。次のことを確認します。
  - 時刻がすべてのレプリカサーバ間で同期されていること。
  - レプリカ同期とバックグラウンドプロセスが正常であること。

2 eDirectoryのサブシステム

このセクションは、eDirectoryのサブシステムについて説明します。

FLAIMデータベース

eDirectoryはデータベースとしてFLAIMを使用しています。FLAIM (Flexible Adaptable Information Manager)は、従来型の情報、揮発性情報および複雑な情報に対して使用されます。FLAIMは拡張性に優れたデータベースエンジンで、マルチリーダ/シングルライタ並列実行モデルをサポートしています。リーダはライタをブロックせず、ライタもリーダをブロックしません。

物理的に、FLAIMはデータをブロックに体系化します。一部のブロックは、一般的にメモリ内に保持されます。これらがブロックキャッシュです。エントリキャッシュ(レコードキャッシュと呼ばれることもあります)は、データベースからの論理エントリをキャッシュします。エントリは、ブロックキャッシュ内のアイテムで構成されています。FLAIMは両方のキャッシュに対してハッシュテーブルを保持します。ハッシュバケツのサイズは、アイテムの数に応じて定期的に調整されます。

デフォルトで、eDirectoryは4KBのブロックサイズを使用します。DIB全体をキャッシュするためのブロックキャッシュサイズはDIBサイズに等しく、エントリキャッシュに必要なサイズはDIBサイズの約2倍から4倍です。

エントリを読み出す際に、FLAIMはまずエントリキャッシュ内のエントリを確認します。エントリが存在する場合、ブロックキャッシュからの読出しは必要ありません。ディスクからブロックを取得する際に、FLAIMはまずキャッシュ内のブロックを確認します。ブロックが存在する場合、ディスク読み込み操作は必要ありません。

エントリが追加されたかまたは変更された場合、そのエントリに対応するブロックはディスクに直接コミットされないので、ディスクとメモリは同期されていない可能性があります。ただし、エントリに対する更新はロールフォワードログ(RFL)に記録されます。RFLは、システム障害後にトランザクションを回復するために使用されます。

Least Recentrly Used (LRU)は、キャッシュ内のアイテムを置換するために使用される置換アルゴリズムです。

チェックポイント

チェックポイントは、オンディスクバージョンのデータベースとインメモリ(キャッシュされた)データベースの整合性を保ちます。FLAIMは、データベースへの最小限の更新アクティビティ中にチェックポイントを実行することができます。これは毎秒実行され、ダーティブロック(ダーティキャッシュ)をディスクに書き込みます。キャッシュ内で変更されたものの、まだディスクに書き込まれていないブロックを「ダーティブロック」と呼びます。FLAIMはデータベースのロックを取得し、チェックポイントが完了するか他のスレッドがデータベースの更新を待機するまで、可能な限り最大量の仕事をします。オンディスクデータベースが同期状態からかけ離れてしまわないため、ある条件下では、データベースの更新を待機しているスレッドがある場合でもチェックポイントが強制的に実行されます:

- チェックポイントスレッドが既定の時間(デフォルトで3分)以内にチェックポイントを完了できない場合、強制的に実行され、ダーティキャッシュは排除されます。
- ダーティキャッシュのサイズがmaxdirtycache(設定されている場合)よりも大きい場合、チェックポイントはダーティキャッシュのサイズをmindirtycache(設定されている場合)または0まで強制的に減らします。

インデックス

インデックスは、インデックス内で特定のキーを探すというタスクのスピードを大幅に向上できるように配置されたキーの集合です。インデックスキーは、エントリから一つ以上のフィールド(属性)の内容を抽出することで構築されます。インデックスはブロックキャッシュ内に保持されます。インデック付き属性が変更された場合、インデックスブロック内での変更が必要です。

eDirectoryは、システム属性(フィールド)に対してデフォルトのインデックス集合を定義しています。parentIDおよびancestorIDなどのシステム属性は、1レベル検索およびサブツリー検索に使用されます。これらのインデックスは、中断や削除ができません。ディレクトリは内部でこれらを使用します。デフォルトインデックスは、CN、Surname、GivenNameなどの属性に対して定義されています。インデックスには、実在インデックス、値インデックスおよび部分文字列インデックスの種類があります。これらのインデックスは中断できます。削除すると、自動的に再作成されます。

iManagerまたはndsindex Lightweight Directory Access Protocol (LDAP)ユーティリティを使用してインデックスを作成できます。インデックス (http://www.novell.com/documentation/edir88/edir88/data/a5tuuu5.html)はサーバに固有のものです。

DSTrace (ndstrace)内でStorage Manager(StrMan)タグを有効にすることで、検索クエリに選んだインデックスを確認できます。

次に示すのは、"cn=admin"、CNを使用したサブツリー検索に対するDSTraceログの例です。

```
3019918240 StrMan: Iter #b239c18 query ((Flags&1)==1) &&
((CN$217A$.Flags&8=="admin") && (AncestorID==32821))

3019918240 StrMan: Iter #b239c18 index = CN$IX$220
```

次に示すのは、"Description=This is for testing"、AncestorIDを使用したサブツリー検索に対するDSTraceログの例です。

```
2902035360 StrMan: Iter #83075b0 query ((Flags&1)==1) &&
((Description$225A$.Flags&8=="This is for testing") && (AncestorID==32821))

2902035360 StrMan: Iter #83075b0 index = AncestorID_IX
```

サーバ側のソートによって検索のパフォーマンスを向上させるには、-aオプションを使用して、新しいインデックスの作成時に渡される属性のリストにAncestorID属性でプレフィックスを付けるようにします。

ロールフォワードログ

FLAIMは、各更新トランザクションに対する操作をロールフォワードログ(RFL)ファイルに記録します。RFLは、システム障害からトランザクションを回復する時や、バックアップから復元する時に使用されます。RFLファイルはチェックポイントが完了するたびに切り詰められます。ただしhot continuous backupを使用して(rflkeepfiles (http://www.novell.com/documentation/edir88/edir88/data/a2n4mb7.html))がオンになっている場合を除きます。

FLAIMの属性コンテナリゼーション

エントリキャッシュの使用率を最適化し、属性検索操作のパフォーマンスを向上させるため、FLAIMは、値のサイズが大きい属性または値の個数が多い属性を、属性コンテナと呼ばれる別の場所に格納します。デフォルトでは、次の条件を満たす属性がコンテナに自動的に移されます:

- 値の数が25個を超えている
- サイズが2048バイトを超える値を持っている

属性の自動コンテナリゼーションを無効にするには、_ndsdb.iniファイルにdisablemovetoattrcontainer=1を追加し、eDirectoryを再起動します。

eDirectoryでは、属性の移動を柔軟にスケジューリングできます。まず、移動する準備ができている属性を表示し、自分の都合に合わせて移動をスケジュールします。

属性コンテナに移動する準備ができている属性の数を表示するには、ndscheckコマンドを実行します。属性の詳細を表示するには、［**エージェント環境設定**］で、iMonitorの疑似サーバオブジェクトの［**dsContainerReadyAttrs**］属性を使用します。ユーザは、インデックス作成のためにマークされている属性を［エージェントヘルス］で確認することもできます。

属性コンテナリゼーションを開始するには、擬似サーバオブジェクトのndsrepairの単一オブジェクト修復オプションを使用します。属性をコンテナに格納するには、ndsrepairコマンドを実行します。その際次のように、新しいアドバンススイッチ-amと、その後に属性名を指定します。

```
ndsrepair –J <Pseudo server object ID> –Ad –AM/–am <attribute name>
```

属性コンテナに属性を移動すると、eDirectoryによって、その属性の名前でシステムインデックスが作成されます。コンテナに格納した属性を、元のコンテナに戻すことはできません。

注: 属性に2048バイトよりも大きい値がある場合は、コンテナリゼーションは依然として実行されますが、eDirectoryでシステムインデックスは作成されません。

スレッドプール

eDirectoryはパフォーマンスの理由でマルチスレッドになっています。マルチスレッドでは、システムがビジーの場合、負荷に対応するためより多くのスレッドが作成され、余分なオーバーヘッドを避けるためにいくつかのスレッドが終了されます。スレッドを頻繁に作成したり破棄したりする

のは非効率で高くつきます。タスクごとに新しいスレッドを生成し破棄するのではなく、多くのスレッドが開始されプールに置かれます。システムは、必要に応じてスレッドプールからスレッドをタスクに対して割り当てます。タスクは2種類のキューに保持されています:

- すぐにスケジューリングが必要なタスクは、Readyキューに保持されます。
- 後でスケジューリングが必要なタスクは、Waitingキューに保持されます。

すべてのモジュールがスレッドキューを使用するわけではありません。プロセスに対するスレッドの実際の数は、スレッドプールに存在する数よりも多くなります。たとえば、FLAIMは自身のバックグラウンドスレッドを単独で管理します。

ndstrace -c threadsコマンドを実行すると、次のスレッドプールに関する統計が返されます。

- 生成されたスレッド、終了されたスレッド、およびアイドル状態にあるスレッドの合計数。
- 現時点でのワーカスレッドの合計数およびワーカスレッドのピーク数。
- Readyキュー内のタスクの数と、タスクのピーク数。
- Readyキューで費やされる最大、最小および平均のマイクロ秒数。
- Waitingキュー内の現在および最大のタスク数。

サンプルスレッドプールの例:

```
Thread Pool Information
Summary      : Spawned 42, Died 5
Pool Workers : Idle 8, Total 37, Peak 37
Ready Work   : Current 0, Peak 10, maxWait 67436 us
Sched delay  : Min 14 us, Max 1052004 us, Avg: 792 us
Waiting Work : Current 17, Peak 21
```

特定のスレッドプールパラメータがあります。

- **n4u.server.max-threads:** プール内の利用可能なスレッドの最大数。
- **n4u.server.idle-threads:** プール内の利用可能なアイドルスレッドの最大数。
- **n4u.server.start-threads:** 開始されたスレッドの数。

ndsconfig getおよびndsconfig setコマンドを実行し、スレッドプールのサイズを取得し設定します。

3 システムボトルネックの分析

システムリソースはeDirectoryのパフォーマンスに影響を及ぼします。最新バージョンのオペレーティングシステムに更新することで、パフォーマンスを改善できます。

ディスクI/Oサブシステム

ディスクサブシステムは最も一般的なボトルネックです。I/Oは長いキューに対して比較的長い時間がかかり、ディスクの高使用率やCPUサイクルのアイドリングにつながります。予想されるピーク負荷時にiostatツールを使用し、平均応答時間の指標を測定してください。

ディスクの読み込み、書き込みおよび更新操作は連続的であるか、またはランダムです。ランダムな読み込みおよび更新は、eDirectoryの展開のなかで最も一般的なアクセスパターンです。

ランダムワークロードに対するソリューション:

- RAMを増やす。これにより頻繁に使用するデータや先読みデータをファイルシステム層にキャッシュすることができるようになります。また、DIBをFLAIMサブシステム内にキャッシュすることもできるようになります。
- DIBに専用のボリュームを使用する。スピンドル付近に生成されたボリュームに対して、ファイルシステムパフォーマンスが向上します。RFLおよび他のログに専用のボリュームを使用する。
- 時間が経過するにつれ、フラグメンテーションによりディスクのレイテンシが増加するので、デフラグメントを行う必要があります。
- FLAIMRFLのために別個のディスクドライブを追加する。このタイプのログ記録は高速ディスク上で実行できます。
- より多くのディスクドライブで、RAID 10(1+0)を使用する。

eDirectoryが作成するファイルは4GBまで増大することがあります。大きなファイルに対処できるよう最適化されたファイルシステムは、eDirectoryと効率的に連携できます。

- Solaris™のVeritas* VxFSファイルシステムはエクステントベースのファイルシステムで、ファイルシステムメタデータは大きなファイルに対して最適化されています。UFSファイルシステムは間接的なブロックベースで、ファイルシステムメタデータは多数のブロック内に保存されます。大きなファイルに散乱させられることさえあり、より大きなファイルに対してUFSはさらに遅くなります。
- Linux™のReiserファイルシステムは高速なジャーナリングファイルシステムで、大きなDIB集合に対してext3ファイルシステムよりもパフォーマンスが優れています。とはいえ、ext3のライトバックジャーナリングモードはReiserファイルシステムのパフォーマンスに匹敵することが知られています。ただデフォルトのオーダードモードはより優れたデータの整合性を実

現します。XFSはハイパフォーマンスのジャーナリングファイルシステムで、大きなファイルを処理し、円滑なファイル転送を実現できます。eDirectory 9.1は、XFSファイルシステムを持つSLES 11 32ビットおよび64ビットプラットフォームをサポートしています。

FLAIMは、4KBおよび8KBのブロックサイズをサポートしています。デフォルトは4KBです。これはLinuxでのデフォルトのブロックサイズと同じです(tune2fs -1 device)。Solarisでは、UFSファイルシステムは8KBのデフォルトブロックサイズで生成されます(df -g mountpoint)。FLAIMのブロックサイズがファイルシステムのブロックサイズよりも小さい場合、ブロックへの部分的な書き込みが発生することがあります。データベースのブロックサイズがファイルシステムのブロックサイズよりも大きい場合、各ブロックの読み込みおよび書き込みは、連続した別個の物理的I/O操作に分割されてしまいます。そのため、FLAIMのブロックサイズは必ずファイルシステムのブロックサイズと同じにするべきです。

ブロックサイズは、DIBの作成時にのみ調整することができます。「blocksize=8192」という行を_ndsdb.iniに追加し、8KのブロックサイズでDIBを作成してください。

適切なブロックサイズは、展開でのFLAIMレコードの平均サイズによって決まります。展開に適切なブロックサイズを判断するために、テストデータの適切な集合に対する実験的テストが必要です。

CPUサブシステム

eDirectoryは、高度にスケーラブルなアーキテクチャ上に構築されます。プロセッサの数が増すにつれ、パフォーマンスも向上します。高負荷下で、少なくとも12番目のプロセッサまでスループットの増加が観察されています。ただし、システムへの負荷が増大する際、他のリソースのパフォーマンスがスループット増加に影響を与えます。サーバのディスクとメモリの構成が十分でないことがよくあります。プロセッサは次の状況下でのみ、追加すべきです。

- 現在使用しているプロセッサに対する平均の負荷が、使用率75%を超えている場合。現在のCPU使用率が75%未満の場合、CPUを増やしてもパフォーマンスは向上しないことがあります。
- パフォーマンスが十分に向上する場合。

eDirectoryに設定されているスレッドが多すぎる場合、CPU時間の大半がコンテキストスイッチに費やされます。この場合、スレッドを減らすことでスループットを改善できます。

メモリサブシステム

RAMを増やすことで、サーバアプリケーションのパフォーマンスが著しく改善することがあります。eDirectoryデータベースをファイルシステムまたはFLAIMキャッシュにキャッシュすることで、検索および変更操作のパフォーマンスが向上します。ただし、大規模な展開ではDIB全体をキャッシュすることはできません。FLAIMエントリおよびブロックキャッシュのサイズを減らせる場合でも、ページスワッピングは避けてください。メモリサブシステムについての詳細は、vmstatツールを使用してください。

eDirectoryがメモリを使用するように、スレッドプールからの各スレッドはスタックとして1MBのRAMを使用します。デフォルトでは、FLAIMのキャッシュサイズは200MBに設定されています。

eDirectoryが起動する際いくつかのローダブルモジュールも起動されますが、eDirectoryのローダブルモジュールアーキテクチャでは、使用しないモジュールをロードしないことでプロセスのメモリフットプリントを減少させることができます(たとえば、SecretStore、LDAPまたはeMBox)。さらに、IDMのような製品には、eDirectory内で実行されるモジュールがいくつかあります。

eDirectoryが使用するメモリが増加しているように思えるかもしれません。eDirectoryプロセスがメモリを解放しても、システム空きプールに対して解放されない場合があります。eDirectoryが内部的に使うメモリ管理は、将来のためにメモリ割り当てを最適化しようとするからです。これはFLAIMダイナミックコンフィギュレーションをお勧めしない理由の一つです。Topツールを使用し、展開でのndsdプロセスの仮想メモリサイズを確かめてください。

プロセスに割り当てられる最大メモリは、いくつかの方法で制限されています。RAMの一部は、オペレーティングシステムおよびシステムの他のプロセスによって使用されます。オペレーティングシステムは、プロセスが使用する物理RAMに制限を課すことがあります。

ネットワークサブシステム

一般的な展開では、ピーク時のネットワーク負荷に対処する十分な帯域幅があります。十分な帯域幅があれば、エラー、コリジョンおよびパケットドロップを減らすことができます。netstatツールを使用して、ネットワークの統計を確認してください。

いくつかのオペレーティングシステムは、ネットワークインテンシブサーバをチューニングするためのTCP/IPチューナブルパラメータを提供しています。詳細は、オペレーティングシステムのドキュメンテーションを参照してください。

ネットワークがボトルネックなら、帯域幅を増やすべきです。アプリケーションサーバとeDirectoryサーバ間に専用のプライベートネットワークを配置することで、ネットワークの輻輳を減少させることができる場合があります。

4 eDirectoryサブシステムのチューニング

このセクションでは、次の情報を紹介します。

FLAIMデータベース

キャッシュのサイズ設定はおそらく、eDirectoryの全体的なパフォーマンスに影響を及ぼす最も重要な要素です。キャッシュ可能なアイテム(ブロックおよびエントリ)の数が増すにつれ、全体のパフォーマンスも向上します。キャッシュの中でブロックやエントリが見つかる回数の割合を、ヒット率と呼びます。比率が高くなると、パフォーマンスが良くなります。iMonitorを使用することで、ヒット率を確かめることができます。

ブロックキャッシュは更新操作にとって最も効果的です。エントリキャッシュは、エントリのベーススコープ検索を行う操作にとって最も効果的です。とはいえ、1レベルおよびサブツリースコープ検索のどちらも、ブロックキャッシュだけでなくエントリキャッシュも使用します。ブロックキャッシュは、インデックスを取得する際に使用されます。必要に応じて適切な型のインデックスを作成してください。詳細は「20 ページの 「インデックスの選択」」を参照してください。

ブロックキャッシュに失敗すると、ディスク読み込み操作が行われます。ディスク読み込みは常に高くつきますが、ファイルシステムキャッシュからブロックを取得できる場合は避けることができます。

データベース全体をブロックキャッシュにキャッシュするために必要なメモリの量は、ディスク上のデータベースのサイズとほぼ同じで、データベース全体をエントリキャッシュにキャッシュするために必要なメモリの量は、ディスク上のデータベースのサイズのほぼ2倍から4倍です。システム上のメモリが少ない場合は、エントリキャッシュを減らし、ブロックまたはファイルシステムキャッシュを増やしてください。

読み込みがディレクトリ内のエントリ集合に集中している場合、エントリキャッシュのヒット率が向上する限り、エントリキャッシュを増やしてください。

読み込みパターンが完全にランダムで、DIBが利用可能なRAMよりもはるかに大きい場合、エントリキャッシュよりもブロックキャッシュおよびファイルシステムキャッシュが大きくなるようにしてください。

どんな方法でeDirectoryをチューニングし、パフォーマンスを改善するとしても、実験的なテストをする必要があります。検索が集中する環境でのエントリキャッシュとブロックキャッシュの適切な比率は、2:1です。他のプロセスのために十分なメモリが残っていることを確認してください。FLAIMキャッシュサイズを減少させることができるとしても、ページスワッピングは避けてください。

FLAIMはあらかじめ割り当てられたキャッシングを提供するので、eDirectoryキャッシュに割り当てられるメモリがネイティブのオペレーティングシステムメモリマネージャに断片化させられることは絶対にありません。

インデックスの選択

インデックスは、1レベル検索またはサブツリースコープ検索のパフォーマンスを向上させるためのものです。動的グループも、1レベルまたはサブツリースコープ検索を使用します。インデックスは、ベーススコープ検索では使用されません。

Presenceインデックスはpresent値およびnot present (deleted)値の間で違いがないので、主に内部的な目的のために使用されます。アプリケーションがPresence型の検索クエリを実行する場合、このインデックスは決して使用されないので、アプリケーションは自身のために作成されたPresenceインデックスを持つべきではありません。

アプリケーションは属性に対してValueインデックスを作成することができ、これはほとんどの検索に対して十分です。FLAIMは、属性に対してPresense検索およびSubstring検索の両方を行うために、Valueインデックスを使用できます。

Substringインデックスは、属性に対する更新の速度を著しく落とします。Substringインデックスをサポートするために必要なインデックスブロックの数は、Valueインデックスと比較してかなり大きくなります。それらをキャッシュするためにより多くのブロックキャッシュが必要になるということです。必要なときにだけ、Substringインデックスを作成してください。ほとんどの検索にとって、Valueインデックスは十分なはずです。とはいえ、ValueインデックスによるSubstring検索では許容できるパフォーマンスが得られない場合、属性に対してSubstringインデックスを作成することができます。

選択したインデックスにかかわらず検索操作が完了するのに長い時間がかかる場合、検索フィルタの属性の一つに対して、新しい値インデックスを導入することもできます。インデックス付けしたときに最高の結果を生み出す属性を選んでください。

更新のチューニング

ブロックキャッシュは、更新操作にとって最も効果的です。インデックスもブロックキャッシュにあります。インデックスは検索を高速化するのに役立ちますが、インデックスが多すぎるとサーバがそれらを維持するのに忙しくなります。属性値が変更、追加または削除されると、インデックスは変更されます。大規模なアップロード操作中は、アップロード高速化のためインデックスを無効にできます。

RFLディレクトリをDIBディレクトリとは異なるディスクに置くことで、パフォーマンスが向上します。

更新操作に対する応答時間の許容限度は、maxdirtycaheを使って調整することができます。たとえば、サーバ応答の許容限度が5秒で、ランダムディスク書き込み速度が毎秒20Mの場合、mazdirtycacheを20x5 = 100MBに設定します。ブロックキャッシュがこの量のダーティブロックをメモリ内に保持できることを確認して下さい。詳細については、30 ページの 「_ndsdb.iniによるFLAIMキャッシュ設定の変更」を参照してください。

スレッドプール

デフォルトでは、スレッドプールで利用可能なスレッドの最大数は256です。ほとんどの展開にとってこの数は十分なはずです。大規模な展開では、512スレッドまで増やすことができます。次のような状況では、プール内のスレッドの数を増やす必要があります。

- アイドルスレッド数が0であることがよくある。
- Readyキュー内のタスクに費やされる平均時間が長く、増加している。
- Readyキュー内のタスクの数が多く、増加している。

サーバのパフォーマンスが向上しているなら、最大スレッド数を増やしてください。CPU使用率も増加するはずです。

スレッドプールの統計を確認するための詳細については、『*NetIQ eDirectory管理ガイド*』の「「スレッドプール統計の表示」」を参照してください。

ACL

- 21 ページの 「eDirectoryの検索および読み込みの改善」
- 22 ページの 「ACLテンプレートを無効にする」

eDirectoryの検索および読み込みの改善

eDirectory内のLDAP検索は、ユーザに返される属性の数に応じて結果を返します(inetOrgPerson)。

eDirectory内でオブジェクトが作成されると、デフォルトのACLがオブジェクトに追加されることがあります。これは、オブジェクトが属するobjectClassのための、スキーマ定義内のACLテンプレートによって決まります。たとえば、inetOrgPersonのデフォルト設定では、ユーザオブジェクトにACLを6つまで追加することができます。すべての属性と共にユーザオブジェクトを返すようにLDAP検索リクエストがなされた場合、ACL属性を含まずにユーザオブジェクトを返す場合よりも、クライアントにオブジェクトが返されるまでに少し長い時間がかかります。

デフォルトのACLはオフにすることができますが、ACLはより適切なアクセス制御のために必要なので、管理者はオフにしたくないと考えることがあります。しかし、それらをリクエストしないか読み込みフィルタ済み属性としてマーク付けすることにより、検索のパフォーマンスを向上することができます。ほとんどのアプリケーションは有効な特権を使用しており、特定のACLに依存していないので、こうした変更が何らかのアプリケーションを壊してしまうことはありません。

ACLをリクエストしない アプリケーションによってはACL属性を必要としないので、それらのアプリケーションは関係のある特定の属性をリクエストするように変更できます。これによりLDAP検索のパフォーマンスが改善されます。

ACLを読み込みフィルタ済みとしてマーク付けする アプリケーションの変更ができない場合、管理者はarf_acl.ldifを使用してACL属性を読み込みフィルタ済み属性としてマーク付けすることができます。ACLが読み込みフィルタ済み属性としてマーク付けされている場合、すべての属性がリクエストされても、サーバはエントリのこの属性を返しません。とはいえ、もし運用属性を返すようにLDAP検索がなされるか、リクエストがACL属性を明示的に要求する場合、マーク付けされたこの属性は返されます。ACL属性への読み込みフィルタ済みフラグは、rrf_acl.ldifを使用することでオフにすることができます。これらのLDIFはスキーマ上のACL属性に影響するので、ツリールートに対してスーパバイザ権を持っているユーザだけが拡張できます。

デフォルトでは、ACLは読み込みフィルタ済みとしてマーク付けされていないので、すべての属性を返すリクエストに対するパフォーマンスの効果は見られません。

次の表は、異なるプラットフォーム上でのarf_acl.ldifおよびrrf_acl.ldifファイルの位置を示しています。

プラットフォーム	ディレクトリ
Linux	◆ /opt/novell/eDirectory/lib64/nds-schema/
Windows	◆ <unzipped_location>\eDirectory\windows\x64\NDSonNT\ndsnt\nds

ACLテンプレートを無効にする

バルクロードのパフォーマンスを向上させるために、ACL(アクセス制御リスト)テンプレートを無効にすることができます。これによりいくつかのACLが見つからなくなりますが、必要なACLをLDIFファイルに追加するか、それらのACLを後から適用することで、この問題は解決できます。

1 次のコマンドを実行します。

```
ldapsearch -D cn_of_admin -w password -b cn=schema -s base
objectclasses=inetorgperson
```

このコマンドの出力は次のようになります。

```
dn: cn=schema

objectClasses: (2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson' SUP
organizationalPerson STRUCTURAL MAY (groupMembership $ ndsHomeDirectory
$ loginAllowedTimeMap $ loginDisabled $ loginExpirationTime $
loginGraceLimit $ loginGraceRemaining $ loginIntruderAddress $
loginIntruderAttempts $ loginIntruderResetTime $
loginMaximumSimultaneous $ loginScript $ loginTime $
networkAddressRestriction $ networkAddress $ passwordsUsed $
passwordAllowChange $ passwordExpirationInterval $
passwordExpirationTime $ passwordMinimumLength $ passwordRequired $
passwordUniqueRequired $ printJobConfiguration $ privateKey $ Profile $
publicKey $ securityEquals $ accountBalance $ allowUnlimitedCredit $
minimumAccountBalance $ messageServer $ Language $ UID $
lockedByIntruder $ serverHolds $ lastLoginTime $ typeCreatorMap $
higherPrivileges $ printerControl $ securityFlags $ profileMembership $
Timezone $ sASServiceDN $ sASSecretStore $ sASSecretStoreKey $
sASSecretStoreData $ sASPKIStoreKeys $ userCertificate
$ nDSPKIUserCertificateInfo $ nDSPKIKeystore $ rADIUSActiveConnections $
rADIUSAttributeLists $ rADIUSConcurrentLimit $ rADIUSConnectionHistory
$ rADIUSDefaultProfile $ rADIUSDialAccessGroup $ rADIUSEnableDialAccess
$ rADIUSPassword $ rADIUSServiceList $ audio $ businessCategory $
carLicense $ departmentNumber $ employeeNumber $ employeeType $
givenName $ homePhone $ homePostalAddress  $ initials $ jpegPhoto $
labeledUri $ mail $ manager $ mobile $ pager $ ldapPhoto $
preferredLanguage $ roomNumber $ secretary $ uid $ userSMIMECertificate
$ x500UniqueIdentifier $ displayName $ userPKCS12) X-NDS_NAME 'User' X
-NDS_NOT_CONTAINER '1' X-NDS_NONREMOVABLE '1' X-NDS_ACL_TEMPLATES
('2#subtree#[Self]#[All Attributes Rights]' '6#entry#[Self]#loginScript'
'1#subtree#[Root Template]#[Entry Rights]' '2#entry#[Public]#messageServer'
'2#entry#[Root Template]#groupMembership' '6#entry#[Self]#printJobConfiguration'
'2#entry#[Root  Template]#networkAddress'))
```

2 この出力から、太字で示されている情報を削除します。

3 変更を加えた出力をLDIFファイルとして保存します。

4 新しく保存したLDIFファイルに次の情報を追加します。

```
dn: cn=schema

changetype: modify

delete: objectclasses

objectclasses: (2.16.840.1.113730.3.2.2)

-

add:objectclasses
```

これにより、新しいLDIFは次のようになります。

```
dn: cn=schema

changetype: modify

delete: objectclasses

objectclasses: (2.16.840.1.113730.3.2.2)

-

add:objectclasses

objectClasses: (2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson' SUP
organizationalPerson STRUCTURAL MAY (groupMembership $ ndsHomeDirectory
$ loginAllowedTimeMap $ loginDisabled $ loginExpirationTime $
loginGraceLimit $ loginGraceRemaining $ loginIntruderAddress $
loginIntruderAttempts $ loginIntruderResetTime $
loginMaximumSimultaneous $ loginScript $ loginTime $
networkAddressRestriction $ networkAddress $ passwordsUsed $
passwordAllowChange $ passwordExpirationInterval $
passwordExpirationTime $ passwordMinimumLength $ passwordRequired
$ passwordUniqueRequired $ printJobConfiguration $ privateKey $ Profile $
publicKey $ securityEquals $ accountBalance $ allowUnlimitedCredit $
minimumAccountBalance $ messageServer $ Language $ UID $
lockedByIntruder $ serverHolds $ lastLoginTime $ typeCreatorMap $
higherPrivileges $ printerControl $ securityFlags $ profileMembership $
Timezone $ sASServiceDN $ sASSecretStore $ sASSecretStoreKey $
sASSecretStoreData $ sASPKIStoreKeys $ userCertificate $
nDSPKIUserCertificateInfo $ nDSPKIKeystore $ rADIUSActiveConnections $
rADIUSAttributeLists $ rADIUSConcurrentLimit $ rADIUSConnectionHistory $
rADIUSDefaultProfile $ rADIUSDialAccessGroup $ rADIUSEnableDialAccess
$ rADIUSPassword $ rADIUSServiceList $ audio $ businessCategory $
carLicense $ departmentNumber $ employeeNumber $ employeeType $ givenName $
homePhone $ homePostalAddress  $ initials $ jpegPhoto $ labeledUri $ mail
$ manager $ mobile $ pager $ ldapPhoto $ preferredLanguage $ roomNumber
$ secretary $ uid $ userSMIMECertificate $ x500UniqueIdentifier $
displayName $ userPKCS12) X-NDS_NAME 'User' X-ND S_NOT_CONTAINER '1' X
-NDS_NONREMOVABLE '1')
```

5 次のコマンドを入力します。

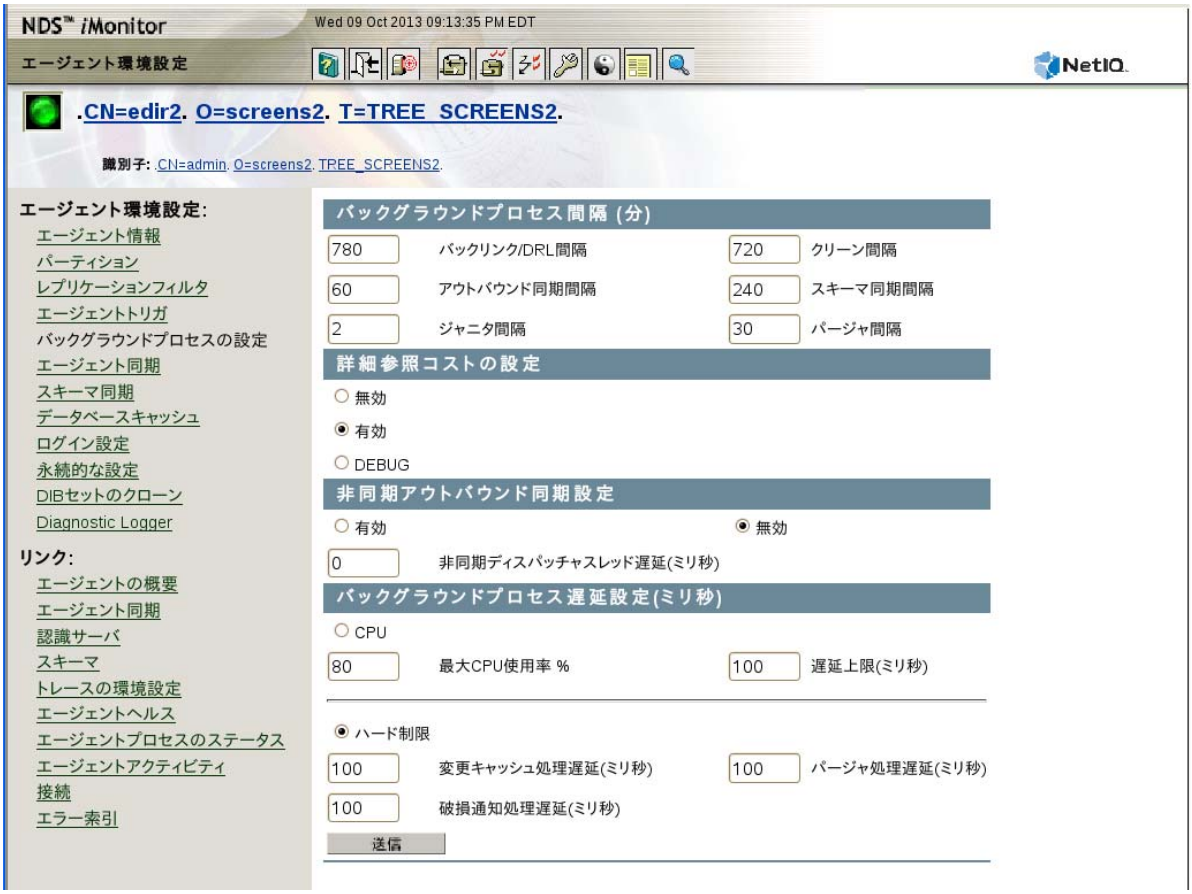
```
ldapmodify -D cn_of_admin -w password -f LDIF_file_name
```

複製

このリリースでは、大規模で動的な環境に対応できるように、いくつかのバックグラウンドプロセスが再設計されました。詳細は、『*NetIQ eDirectory管理者ガイド*』の「「バックグラウンドプロセスの管理」」を参照してください。

ハード制限を*5ms*に、非同期アウトバウンド同期を*有効*に設定することをお勧めします。ただし、CPU使用率が高い場合はスリープの時間を増やしてください。図 4-1は、**バックグラウンドプロセスの遅延設定**に設定された値を示しています。

***図4-1** バックグラウンドプロセスの設定*



10台のサーバを設置し、次の設定で社内実験を実施しました:ハード制限-*0ms*、非同期アウトバウンド同期-*有効*、非同期ディスパッチャスレッド遅延-*0ms*。このテストは、デフォルト設定に比べてレプリケーションが7倍速いことを示しています。このテスト期間中、他のクライアント操作は何も行われませんでした。

注: これらのスケーラビリティの向上によるシステムパフォーマンスの改善を最大限に生かすために、必ずすべてのサーバでeDirectory 9.1以上を使用してください。レプリカリングで古いバージョンが使用されている場合でも、パフォーマンスは向上します。

ソリッドステートディスク(SSD)

より良いIO操作のために、このリリースはエンタープライズSSDをサポートしています。25 ページの 表 4-1は、テスト環境でのSSD修復パフォーマンスの向上を示しています:

***表 4-1** 修復パフォーマンス*

DIBサイズ(GB)	HDD(分)	SSD(分)	向上(%)
11	80	53	33.75
24	277	169	38.98
34	542	296	45.38
75	1383	618	55.31
98	3171	1023	67.73

NMASログイン更新間隔

詳細は、『*NetIQ eDirectory管理ガイド*』の「「sasUpdateLoginInfoおよびsasUpdateLoginTimeInterval属性の使用」」を参照してください。

SSLオーバーヘッド

SSL上のLDAPは、SSLの暗号化要件のためにCPUに付加的な負荷を与えます。実験室でのパフォーマンス研究は、暗号化のオーバーヘッドによる10%を超えるパフォーマンスヒットを示しています。

インポート/エクスポート変換ユーティリティ(ICE)

NetIQインポート/エクスポート変換(ICE)ユーティリティは、データをeDirectoryにアップロードするために、LBURPと呼ばれる最適化されたバルク更新プロトコルを使用します。このプロトコルは、ldapmodifyコマンドを単独で使用してデータをアップロードするよりも大幅に高速です。詳細については、『*NetIQ eDirectory管理ガイド*』の「*オフラインバルクロードユーティリティ*」*を参照してください。*

ldif2dib

ldif2dibユーティリティを使用した、オフラインバルクアップロード中のeDirectoryパフォーマンスのチューニングについては、『*NetIQ eDirectory管理ガイド*』の「*ldif2dibのチューニング*」*を参照してください。*

拡張NCPパケットサイズ

さまざまなサーバ間で通信を行うため、eDirectoryは、通信プロトコルとしてNetware Core Protocol (NCP)を使用します。従来のリリースでは、NCPの最大パケットサイズが64 KBであったため、NCP経由のデータ転送時の最大スループットに制約がありました。今回のリリースでNCPの能力が向上し、最大1 MBのパケットを処理できるようになったため、eDirectoryは1つのパケットで最大1 MBのデータを同期できるようになりました。eDirectoryは、パケットサイズ64 KBで同期を開始し、同期する残りのデータに基づいてパケットサイズを大きくしていきます。これにより、レプリケーションのパフォーマンスが大幅に向上します。両方のサーバが9.2である場合は、この機能拡張を活用するために追加設定を行う必要はありません。

5 eDirectory設定

このセクションでは、次の情報を紹介します。

FLAIMサブシステムの設定

広範な展開と設定に対応するため、eDirectoryにはキャッシュメモリ消費を制御する2つの方法が用意されています。この2つの方法は相互に排他的です。

ハードキャッシュ制限

次のいずれかの方法により、ハードメモリ制限を指定することができます:

- バイトの定数を用いる。
- 物理メモリの割合を用いる。
- 利用可能な物理メモリの割合を用いる。

2番目または3番目の方法でハード制限を指定した場合、バイトの定数に変換されます。2番目の方法を使用した場合、バイト数はeDirectoryが起動する際に検出される物理メモリの割合になります。3番目の方法を使用した場合、バイト数はeDirectoryが起動する際に検出される利用可能な物理メモリの割合になります。

上限の動的な調整

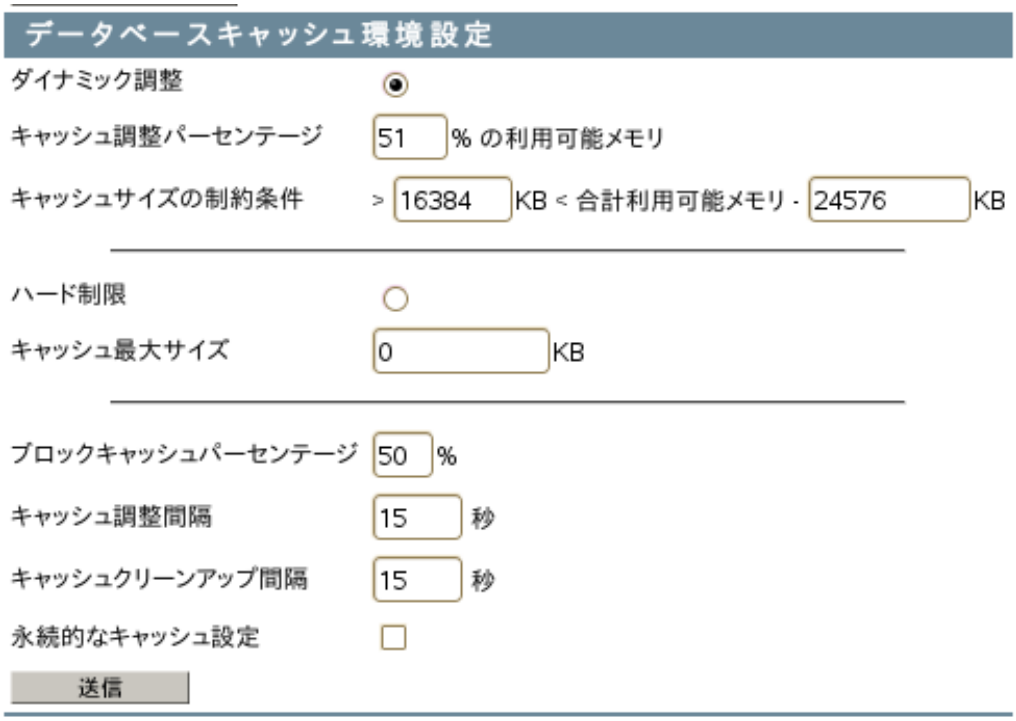
動的調整により、eDirectoryは他のプロセスによる可変メモリ消費に応じて、自身のメモリ消費を周期的に調整できるようになります。一般的な状況下では動的なメモリ調整はうまくいきますが、Linuxプラットフォームでのeディレクトリの最適なパフォーマンスのためにはお勧めしません。Linuxプラットフォームではメモリ使用形態およびメモリアロケータが大きく違うからです。

FLAIMキャッシュ設定の変更

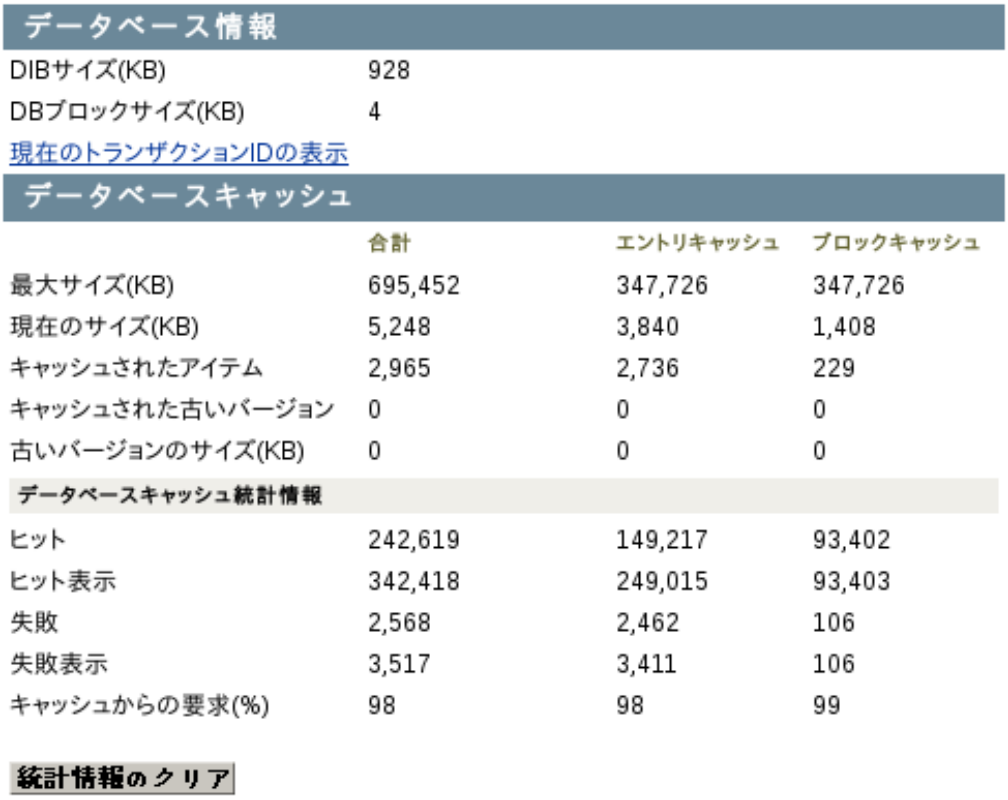
iMonitorによるFLAIMキャッシュ設定の変更

iMonitorを使用して次のことができます。

- キャッシュ設定を確かめる、または変更する。



- キャッシュ統計を監視する。



データベース情報

DIBサイズ(KB)	928
DBブロックサイズ(KB)	4

現在のトランザクションIDの表示

データベースキャッシュ

	合計	エントリキャッシュ	ブロックキャッシュ
最大サイズ(KB)	695,452	347,726	347,726
現在のサイズ(KB)	5,248	3,840	1,408
キャッシュされたアイテム	2,965	2,736	229
キャッシュされた古いバージョン	0	0	0
古いバージョンのサイズ(KB)	0	0	0
データベースキャッシュ統計情報			
ヒット	242,619	149,217	93,402
ヒット表示	342,418	249,015	93,403
失敗	2,568	2,462	106
失敗表示	3,517	3,411	106
キャッシュからの要求(%)	98	98	99

統計情報のクリア

詳細については、iMonitorのエージェント設定にある、データベースキャッシュを参照してください。

データベースキャッシュ情報	説明
最大サイズ	指定したキャッシュが拡張できる最大サイズ(KB)です。
現在のサイズ	指定したキャッシュの現在のサイズ(KB)です。
キャッシュされたアイテム	指定したキャッシュ内のアイテム数です。
キャッシュされた古いバージョン	指定したキャッシュ内の古いバージョンの数です。古いバージョンのキャッシュアイテムは、データベースの読み込みトランザクションの整合性を維持するために保持されます。つまり、あるスレッドが読み込みトランザクションを実行しており、別のスレッドが書き込みトランザクションを実行している場合、書き込みによって変更されるブロックの古いバージョンが読み込み操作のために保持されます。これは、読み込みのトランザクションを実行している間に変更処理が発生したとしても、読み込みの表示結果に整合性があるようにするためです。
古いバージョンのサイズ	キャッシュされる古いバージョンのアイテムサイズ(KB)です。
ヒット	指定したキャッシュ内のアイテムに正常にアクセスできた回数です。

データベースキャッシュ情報	説明
ヒット表示	指定したキャッシュ内で、アイテムに正常にアクセスする前に参照されたアイテム数です。ヒット表示とヒットの比率が、キャッシュ検索効率の目安となります。通常、この比率はほぼ1:1になるようにします。
失敗	アイテムが指定したキャッシュ内に見つからず、低いレベルのキャッシュまたはディスクから取得されなければならなかった回数です。
失敗表示	必要なアイテムが指定したキャッシュ内にないと判断されるまでに、キャッシュ内で参照されたアイテムの数です。失敗表示と失敗の比率が、キャッシュ検索効率の目安となります。通常、この比率はほぼ1:1になるようにします。

_ndsdb.iniによるFLAIMキャッシュ設定の変更

FLAIMキャッシュ設定およびFLAIM設定は、DIBディレクトリにある_ndsdb.iniファイルを変更することにより行えます。_ndsdb.iniファイルを変更した場合、eDirectoryを再起動してください。

動的調整の上限またはハードキャッシュ制限を設定できます。キャッシュオプションを以下にリストします。複数のオプションをコンマで区切り、任意の順序で指定できます。これらはいずれも必須ではありません。

- **DYNまたはHARD** -上限またはハード制限を動的に調整します。
- **% : percentage** -使用する利用可能メモリまたは物理メモリの割合です。
- **AVAILまたはTOTAL** -割合は利用可能なメモリまたは合計物理メモリを指定します。ハード制限にのみ適用でき、動的調整の上限に対しては無視されます。動的調整の上限は、常に利用可能な物理メモリに基づいて計算されるからです。デフォルトでは、AVAILです。
- **MIN:bytes** -最小バイト数です。
- **MAX:bytes** -最大バイト数です。
- **LEAVE:bytes** -残しておく最小バイト数です。

  次に例を示します。

  ```
  cache=HARD,%:75, MIN:200000000
  ```

  ```
  cache=500000000
  ```

- **preallocatecache: true/false** -この設定により、eDirectoryはハードキャッシュ制限に指定されるメモリ量をあらかじめ割り当てます。
- **rfldirectory -** RFLファイルに別のパスを指定できます。
- **cpinterval** -FLAIMがチェックポイントを強制的に実行するまでの秒数です。デフォルトは3分です。
- **maxdirtycache** -ダーティキャッシュバイトの最大数です。
- **lowdirtycache** -ダーティキャッシュバイトの最小数です。
- **blockcachepercent** -ブロックキャッシュに使用されるFLAIMキャッシュの割合です。
- **cacheadjustinterval** -キャッシュを動的に調整する秒間隔です。
- **cachecleanupinterval** -古いバージョンのエントリとブロックをキャッシュから削除する秒間隔です。