Oracle Content Server

パフォーマンス・チューニング・ガイド

10*g* リリース 3（10.1.3.3.0）

部品番号 : B54799-01

2009 年 5 月

ORACLE®

Oracle Content Server パフォーマンス・チューニング・ガイド , 10*g* リリース 3（10.1.3.3.0）

部品番号 : B54799-01

原本名 : Content Server - Performance Tuning Guide, 10*g* Release 3 (10.1.3.3.0)

原本協力者 : Jean Wilson

目次

第 4 章 : Content Server のコア・チューニング

第 5 章 : アドオン・コンポーネントのチューニング

第 6 章 : システム・アーキテクチャのチューニング

索引

1

はじめに

概要

この章では、パフォーマンス・チューニングの概要、およびこのガイドで使用する表記規則について説明します。内容は次のとおりです。

このガイドについて

このガイドでは、Content Server のパフォーマンスの最適化に使用できる手法の概要を説明します。トラブルシューティングと同様に、パフォーマンス・チューニングでは、目的の達成のために明確に定義された一連の手順による体系的な手法を使用します。ただし、トラブルシューティングと異なり、対処が必要な問題領域が明確に定義されていない場合が多くあります。このドキュメントは、チューニングで最大限の効果を上げるための最適な方法を判断する際に役立ちます。

実行できるチューニングには、主に次の 3 つのタイプがあります。

- 外部システムのチューニング : これには、ネットワーク上のメモリーの使用状況、ファイアウォールまたはキャッシュのチェックが含まれます。
- Content Server のチューニング : これには、Content Server のコア・パフォーマンスに影響を及ぼす、デフォルト・パラメータやソフトウェアの設定の変更が含まれます。

- Content Server の付加的要素のチューニング : これには、アドオン・コンポーネント、データベースおよび索引など、Content Server でリソースとして使用される要素のチューニングが含まれます。

一部のチューニングおよびベンチマーキングは、Content Server を主にコンテンツの消費と投稿のどちらに使用しているかによって異なります。このドキュメントでは、主にコンテンツの消費に使用されるシステムに焦点を当てます。

このドキュメントは、次の章で構成されています。

- 第 1 章「はじめに」では、ドキュメントの構成および補足事項について説明します。
- 第 2 章「最適化について」では、最適化プランの作成の個々の手順について説明します。
- 第 3 章「ベンチマーキング」では、最適化の進行状況を判定するためのベンチマークの作成方法および使用方法について説明します。
- 第 4 章「Content Server のコア・チューニング」では、パフォーマンスを変更するために Content Server、索引およびデータベースに行える変更について説明します。
- 第 5 章「アドオン・コンポーネントのチューニング」では、Content Server で使用できる様々なアドオンに実行可能なチューニングについて説明します。
- 第 6 章「システム・アーキテクチャのチューニング」では、システム・アーキテクチャの様々な側面をチューニングし、パフォーマンスを向上させる方法について説明します。

関連ドキュメント

次のドキュメントも、システム・パフォーマンスの最適化に役立ちます。

- 『Troubleshooting Guide』
- 『システム設定およびプロセスの管理』
- 『Planning and Implementation Guide』
- 『Choosing a Search Solution』
- 『Reverse Proxy Server Resource Guide』

対象読者

このガイドは、Content Server およびその関連製品のパフォーマンスの向上を目指すシステム開発者およびシステム管理者を対象としています。

表記規則

このガイドでは次の表記規則を使用します。

- *<Install_Dir>/* という表記は、コンテンツ・サーバー・インスタンスがインストールされているシステム上の場所を参照するために使用されます。
- スラッシュ（/）は、パス名のディレクトリ・レベルの区切りとして使用されます。ディレクトリ名の末尾には常にスラッシュが付きます。
- 注意、技術ヒント、重要な通知および警告では、次の表記規則を使用します。

記号	説明
	これは注意です。特別な注意事項を示すために使用されます。
	これは技術ヒントです。タスクをより簡単に実行するために役立つ情報を示します。
	これは重要な通知です。必要な手順または情報を示すために使用されます。
	これは警告です。データの損失または重大なシステム問題の原因となる可能性がある情報を示すために使用されます。

2

最適化について

概要

パフォーマンス・チューニング（多くの場合、最適化と呼ばれる）では、改善が必要なシステム領域を定義し、そのチューニング方法を特定します。最適化は反復的に行うプロセスであり、アーキテクチャ、インフラストラクチャおよび使用方法によってその内容は大きく異なります。

この章では、最適化に関する次の項目について説明します。

システム・パフォーマンスの低下の検出

一般的に、次の Content Server タスクを実行すると、パフォーマンス低下の明らかな兆候が表れます。

- 検索
- 大量のコンテンツのインポートなどのバッチロードまたは投稿
- ページのレンダリング
- 他の場所へのコンテンツの公開またはプッシュ

このドキュメントでは、パフォーマンスの低下要因の特定、およびパフォーマンス低下の発生箇所の検出に役立つ情報を提供します。

Content Server のチューニングは、次に示す Web アプリケーションのチューニングと同じ基本ルールに従います。

- ユーザー固有のニーズに合せて一般構成を調整します。
- サード・パーティ製のソフトウェアを最適化します。
- ハードウェアの使用を最大化します。
- キャッシュの使用を最適化します。

これらの項目の詳細は、このドキュメントの後半で説明します。

パフォーマンス低下の代表的な要因

一般的に、Content Server のパフォーマンス低下の要因となるのは、不適切なシステム・ハードウェアと不適切なシステム構成の 2 つです。システムのハードウェアと構成を検証し、これらがパフォーマンスの問題の原因ではないことがわかったら、次は Content Server チューニングの調査に進みます。

システム・ハードウェア

パフォーマンス評価は、システム・ハードウェアから開始するのが適しています。システム・パフォーマンスに問題がある場合、次の項目をチェックして、タスクに対して適切であるかを確認してください。

- ファイル・システム・サイズ : 使用しているファイル・システムが不十分であるか、またはその速度が遅すぎる場合、検索パフォーマンスが低下します。サイトで使用しているサーバー数に対して、十分な領域があることを確認してください。

- メモリー・サイズ : メモリー・サイズが小さすぎる場合、スワップの時間によって大幅にパフォーマンスが低下します。十分なメモリーがあること、および過不足なく適切に使用されていることを確認してください。
- プロセッサ・タイプ : 十分な速度を備えたプロセッサを使用していること、および過度に使用していないことを確認してください。

システム構成

システム構成においては、キャッシュとアカウントがパフォーマンス・チューニングのキーとなります。

- 外部接続を必要とするアプリケーション（ネットワーク間通信など）では、キャッシュが必要です。キャッシュのタイプは、使用するアプリケーションによって異なります。検索キャッシュ、リバース・プロキシ Web キャッシュおよびページ・キャッシュなどがあります。また、`cacheInclude` Idoc 機能を使用して、リソース・インクルードをキャッシュできます。キャッシュ・オプションの詳細は、『システム設定およびプロセスの管理』、『Idoc スクリプト・リファレンス・ガイド』、『Reverse Proxy Resource Guide』を参照してください。

  キャッシュが適切に設定されていないと、Web サーバーは絶えず元のサーバーに戻ってコンテンツにアクセスすることになります。これは、コンテンツが変更されていない場合、不要な処理です。キャッシュが小さすぎる、十分に使用されていない、または無視されている場合、コンテンツの取得速度が大幅に低下します。パフォーマンスを最適化する際、まず最初に検討する項目の 1 つは、システムのキャッシュ構成です。

- Content Server に多くのセキュリティ・グループが構成されている場合、システム・パフォーマンスが低下する場合があります。一般的に、15 を超えるセキュリティ・グループを必要とする場合、アカウントベースのセキュリティ・モデルに切り替えてください。アカウントが有効な場合、必要なセキュリティ・グループは最大 15 ～ 20 のみです。アカウントは階層構造で設定できるため、一部のユーザーには階層構造のブランチ全体へのアクセスを許可し、他のユーザーにはアクセス権限を制限することができます。この方法では、特に検索時のシステムのレスポンス時間を大幅に高速化できます。アカウントとユーザー権限の設定の詳細は、『Planning and Implementation Guide』および『Oracle Universal Content Management セキュリティおよびユーザー・アクセスの管理』を参照してください。

その他のよくある質問

次の質問に対する回答は、パフォーマンス・チューニングの対象となる領域を絞り込む上で役に立ちます。

- Content Server のサイズが大きくなりすぎていませんか。コンテンツ・アイテムがシステムに追加されると、データベースおよび検索コレクションへの問合せに費やされる時間が長くなります。対処方法の 1 つは、コンテンツ・アイテムを複数のコンテンツ・サーバーに分散することです。パフォーマンスの低下を緩和するために、クラスタ構成内に別のサーバーを追加することが必要になる場合もあります。
- サイトのトラフィックは増大していますか。サイトの使用量は、Content Server へのリクエスト数で定義されますが、様々なタイプのリクエスト（静的か動的かなど）による負荷は大きく変動します。また、サイトの使用量には、サイトの訪問頻度や滞在時間を組み合せたサイト上のユーザー数も含まれます。トラフィック量が多い場合、サーバーの追加が必要になる場合もあります。これにより、パフォーマンスの低下が緩和されます。
- ネットワークの接続は適正ですか。キャッシュと同様に、ネットワーク・インフラストラクチャはコンテンツの配信速度に影響を及ぼします。
- 正しいバージョンの Content Server を使用していますか。より新しいバージョンには、古いバージョンとは重複できない、最適化された拡張機能がコアのソフトウェアに組み込まれています。必要に応じて、ソフトウェアのアップグレードを検討してください。
- アドオン・コンポーネントがパフォーマンスに影響を与えている可能性はありますか。古いバージョンの Content Server に多くのアドオンを追加している場合、バージョンをアップグレードした方が効果的です。コアの Content Server サーバー・ソフトウェアに追加されている多くのアドオンは、使用しやすいように最適化されています。

その他の質問を次に示します。

- Content Server に十分なメモリーが割り当てられていますか。
- データベースに十分なメモリーがありますか。
- データベースの表の索引は最適化されていますか。
- データベースは適切にチューニングされていますか。
- 検索データベースには、検索対象のフィールドに対する索引がありますか。
- 検索データベースは、最適に体系化されていますか。
- 非効率的な検索またはデータベース問合せはありませんか。

チューニング・プロセスの手順

最適化は、6 つの基本手順からなる反復プロセスです。

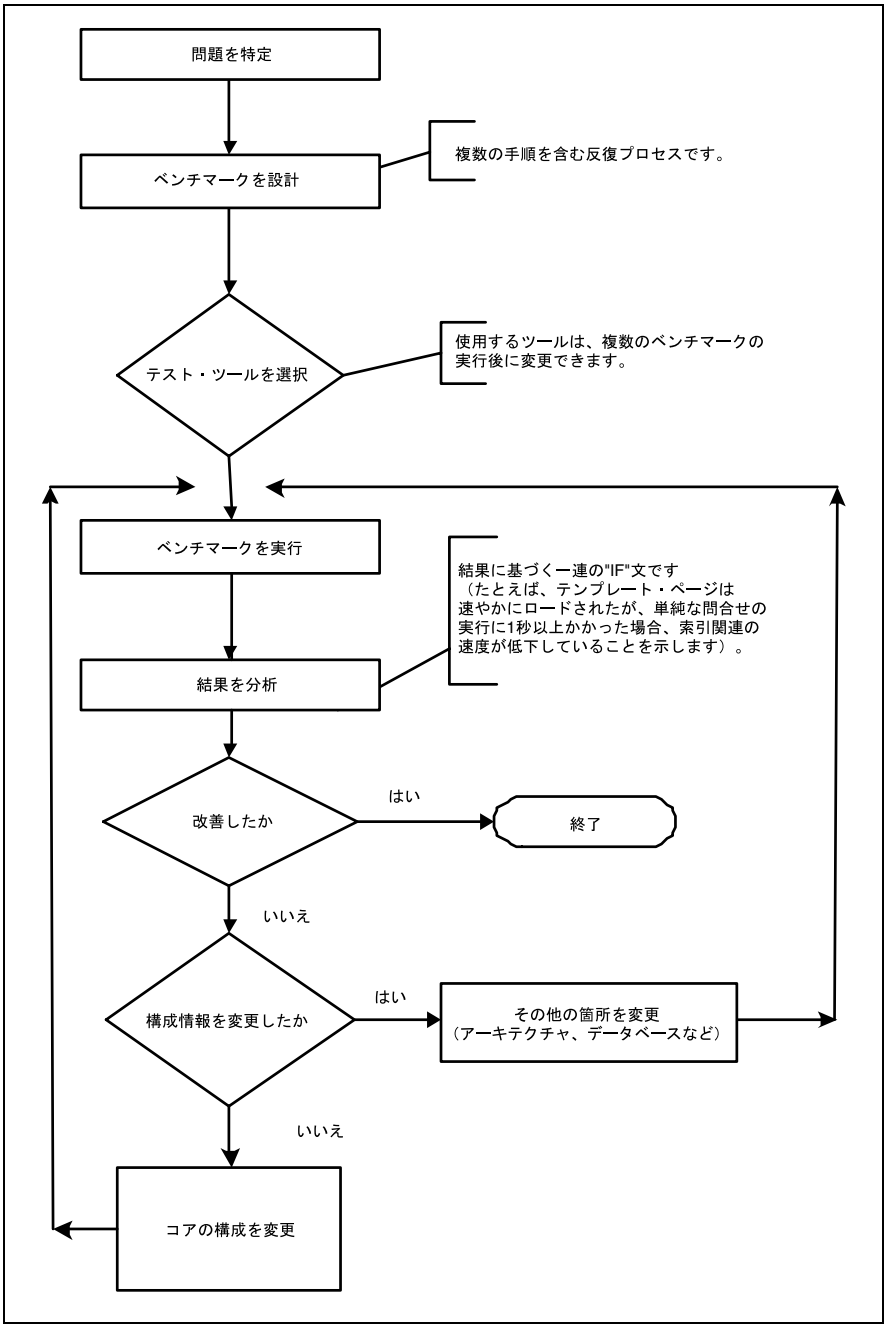
1. 問題を特定し、評価します。システムの使用方法をチェックし、サイトにおいて許容できる動作と許容できない動作を特定します。サイトのパフォーマンス評価に使用できる一般的なシステム統計については、3-7 ページの「結果の分析」を参照してください。
2. ベンチマークを設計します。ベンチマークは、テスト結果を判定する際の基準です。ベンチマークは、パフォーマンスの定量化、および正しい領域の特定に役立ちます。
3. ベンチマークを実行します。典型的なユーザー・セッション（特定のページへのアクセス、フォームの送信、コンテンツのチェックインなど）を再現するスクリプトを作成し、ベンチマーク・シナリオとして使用します。
4. 結果を分析します。テスト結果を検討し、パフォーマンスが低下している箇所を特定できないか確認します。
5. 最適化の手法を適用します。ボトルネックの発生箇所に応じて、様々な手法を使用できます。データベース、インデクサおよびシステム・ハードウェアなど、コンテンツ・サーバー以外の箇所についても最適化を実行できます。
6. 再テストします。最適化手法の効果を確認するために、テストを再度実行します。新しいパフォーマンス・データを記録し、副次的な悪影響がないか確認します。場合によっては、ある領域におけるパフォーマンスの改善が、別の領域においてパフォーマンスの低下を引き起こすことがあります。

注意 : ベンチマークが実環境のシナリオの完全なコピーではないかぎり、テスト・サーバーで得られた改善は実際と異なる場合があることに注意してください。実際の稼働システムでは、同じレベルのパフォーマンスの改善は期待できません。

この後の章で、各手順の詳細を説明します。

次のフローチャートは、最適化プロセスの手順の概要を表しています。

図 2-1　　最適化プロセス



3

ベンチマーキング

概要

ベンチマーキングとは、システムの動作を文書化するテストを設計し、問題の発生箇所を絞り込めるようにこのテストを調整するプロセスです。この章では、ベンチマーキングに関する次の項目について説明します。

ベンチマークの設計

ベンチマークとは、ハードウェアまたはソフトウェア、あるいはその両方のパフォーマンスの比較に使用するテストです。ベンチマークの結果を比較する際、ベンチマークが何をテストするために設計されているかを正確に把握することが重要です。たとえば、画像速度をテストするベンチマークは、テストで使用する画像アプリケーションが実際に使用している画像アプリケーションと異なる場合、意味を持ちません。

基本のベンチマークは短期間で設計可能ですが、より詳細なベンチマークは時間をかけて念入りに開発します。ベンチマークは、平均的なユーザーのパフォーマンスを定量化するために設計することが重要です。平均には常に例外がありますが、平均的ユーザーに焦点を当てることが最も有益です。

設計プロセスを開始する前に、次の点に注意してください。

- 可能なかぎり実際の状況に則したベンチマークを作成します。本番サーバーを使用するか、本番サーバーの正確なレプリカを使用してください。これにより、ユーザーに表示される内容を再現できます。
- ベンチマーク・サーバーを分離します。テストの担当者のみがこのサーバーにアクセスできるようにします。パフォーマンスの変化を定量化するにはこの分離は必須ですが、分離することにより実際の状況とは異なるため結果は歪曲されたものになることに注意してください。
- プロセスを文書化します。実行中のすべてのソフトウェアおよびハードウェアを確認し、パフォーマンスに関するドキュメントを継続的に更新するようにしてください。

ベンチマークの設計のポイント

ベンチマークの設計時には、次の点を考慮します。

現行システムの確認と文書化

ベンチマーク・プロセスを開始する前に、システムの詳細についてすべて文書化します。この作業のポイントは次のとおりです。

- システムに追加されたカスタマイズ内容にエラーがないことを確認します。ロギングにエラーがあると、システムが停止することはなくても、パフォーマンスが低下します。カスタマイズ内容に含まれる IdocScript エラーは、ベンチマークを実行する前に修正してください。
- オペレーティング・システム、ハードウェア、データベース、セキュリティ・システムおよび Content Serrver など、使用しているシステムのすべての部分について、現行のパッチをすべて適用します。

- ❖ ベンチマーク開始時の設定情報を適切な場所に文書化します。
  - ハードウェア : CPU のサイズと構成、メモリー、製造メーカーおよびその他の関連詳細。
  - ソフトウェア : データベース、Web サーバーおよび Content Server で実行しているソフトウェアとバージョン。カスタマイズ内容についても記録します。
  - ネットワーク : ベンチマーク・プロセスに関連するネットワーク・ハードウェアについて文書化します。対象は、ハブ、スイッチ、ファイアウォール、プロキシ・サーバーおよびその他のネットワーク・デバイスなどです。ネットワーク・ハードウェアの帯域幅の制限に注意してください。
  - セキュリティ : LDAP、Active Directory、および Content Server のセキュリティ・グループやアカウントなど、使用しているセキュリティのすべての側面を文書化します。頻繁に発生するセキュリティ・チェックは、パフォーマンスを低下させる場合があります。
- ❖ システムの使用情報を文書化します。次の情報を記録してください。
  - バッチロードの実行頻度
  - 公開の実行頻度（日次、週次、または必要に応じて）
  - 主なユーザーのタイプ（コントリビュータまたはコンシューマ）
  - アーカイブの頻度

文書化することにより、ベンチマークの実行前に、問題のある箇所が明確になる場合があります。文書化した内容を検討して、修正の必要な箇所を特定してください。

- ❖ Web ページの配信速度の向上を図りたいですか。
- ❖ サイトに投稿できるユーザー数を最大化したいですか。
- ❖ 配信可能な Web ページ数を増やしたいですか。
- ❖ 検索およびチェックイン時間を最小化する必要がありますか。

代表的なセッションの再現

すでに述べたように、ベンチマークは、標準のユーザー・セッションの結果をテストするように設定する必要があります。次のような代表的なユーザー・セッションのスクリプトを設計してください。

- ❖ 一般的なページへのアクセス（Site Studio、Publisher または標準テンプレート）
- ❖ 検索の実行
- ❖ 各コンテンツの更新の実行
- ❖ ワークフローの開始またはワークフローへの参加
- ❖ コンテンツのチェックイン

このスクリプトに使用する、複雑性が異なる 20 ～ 30 のコンテンツ・ページを選択します。次に、テストを 1000 回繰り返すプランを作成し、その実行のタイミングを決定してください。

バッチローダまたはアーカイバをテストする場合、バッチ・テスト用の代表的なサンプル項目を入手します。これは、データベース、インデクサおよびリファイナリのテストに役立ちます。Publishing Utility をテストする場合、Site Studio、Publisher および Connection Server から代表的な例を入手してください。可能であれば、有効なテスト結果を得るために代表的な項目をそれぞれ 1000 項目入手してください。

ベンチマーク・ツール

ベンチマーク・スクリプトを設計し、重点を置く領域を決定したら、ベンチマーキングに使用するツールを選択します。

様々なツールを使用できます。簡単なツールの例として、Windows の `taskmgr` コマンド、UNIX システムの `top` コマンドがあります。その他に、Mercury LoadRunner、Microsoft Application Center Test（.NET）、および様々な Java プロファイラなどのツールがあります。

また、多くのデータベース分析ツールも使用できます。ほとんどのツールは、各データベースに固有のものです。サイトのデータベース・アナリストとともに、最適なツールを検討し決定してください。

システムを監視するツールの他に、複数の Content Server ツールを使用して、タイミングとパフォーマンスを監視できます。「Admin Applets」ページで、Refinery、Archiver および Content Server の管理ログ・ファイルを確認します。

また、「Administration」トレイからアクセスする「System Audit Info」ページで、様々なトレース操作を選択することによりトレースを有効にできます。インデクサ、ソケット・リクエスト、システム・データベースなどそれぞれに対する様々なトレース・レポートを生成できます。「System Audit Info」ページの**「Edit Active Console Output Tracing」**セクションで使用可能なトレース・レポートを有効にする方法の詳細は、『Troubleshooting Guide』を参照してください。

次に、使用可能なすべてのトレース・オプションを示します。パフォーマンス・チューニングとは関連のないオプションも含まれています。

- **applet:** Configuration Manager または User Admin などの初期化されたアプレットからの結果セットが含まれます。
- **archiver:** アーカイバ・データ・ファイルの読取り / 書込み、およびアーカイブ・アクティビティの開始時刻 / 終了時刻など、アーカイブ・アクティビティに関する情報を提供します。

- **archiverlocks:** 開始時刻など、アーカイブ時にファイルに適用されたロックに関する情報を提供します。
- **chunkedrequest:** 多数のリクエストが少数のリクエストにグループ化される際に作成されたメッセージおよびヘッダーを表示します。
- **docprofile:** ラベルであるフィールド、非表示のフィールドなどを判別するルールの評価など、コンテンツ・プロファイルの計算結果を表示します。
- **encoding:** 発生したエンコーディング変換、およびエンコーディングが発生したアクティビティに関する情報を表示します。
- **filelock:** アーカイブなどのアクティビティ中にディレクトリに適用された、発生する競合やタイムアウトを中心とした短時間のシステム・ロックに関する情報を表示します。
- **filelonglock:** システムによって適用された長時間ロックの作成、削除およびメンテナンスに関する情報を表示します。
- **filequeue:** ファイル・キューへのアクセス情報を表示します。
- **indexer:** データベースの更新時に発生する索引機能に関する情報を表示します。情報には、索引の更新手順、および各手順の経過時間などが含まれます。
- **indexermonitor:** 開始時刻および終了時刻など、自動索引アクティビティの概要を提供します。
- **indexerprocess:** 手動で起動された索引プロセスに関する情報を表示し、プロセスが適切に終了したかを示します。
- **localization:** ローカリゼーションの使用状況およびアクティビティに関する情報を表示します。
- **mail:** コンテンツ・サーバーによって送信された電子メールについての説明を提供します。
- **pagecreation:** サーバー・スレッド、およびページの生成に費やされた時間など、表示されるページの作成に関する情報を表示します。
- **requestaudit:** リクエストの経過時間、および作成されたリクエスト数など、サービス・リクエストについてのサマリー・レポートを提供します。
- **scheduledevents:** バックグラウンドで毎時または日次で実行するようにスケジュールされているイベントのリストを提供します。
- **schema:** スキーマの公開（.js ファイルとして公開される表およびビュー）およびキャッシュ（コンテンツ・サーバー・メモリーにキャッシュされる表）に関する情報を提供します。
- **searchquery:** 検索に使用されたフィールド、および結果のソート順など、最近実行した検索に関する情報を提供します。

- **socketrequests:** ソケット・リクエストの日付、時刻およびスレッド番号の他、リクエスト時の操作も表示します。
- **system:** システムのソケット・リクエストおよびレスポンスなど、内部システム・メッセージを表示します。
- **systemdatabase:** 実行された問合せ、索引の更新、使用されたスレッドおよび開始時刻など、データベース・アクティビティに関する情報を提供します。
- **transfermonitor:** アーカイバおよびバッチ・ファイル転送アクティビティに関する情報を表示します。
- **userstorage:** アクセス時に実行された操作など、外部ユーザー・リポジトリへのアクセスに関する情報を表示します。
- **workflow:** ドキュメント・タイトルおよびリビジョン番号など、ワークフロー内のコンテンツ・アイテムのメタデータのリストを表示します。

注意 : トレース・レポートは、Content Server 7.0 以降の「System Audit Info」ページで使用できます。それより前のバージョンの場合、構成フラグを使用してトレースを有効にする必要があります。構成の詳細は、『Idoc スクリプト・リファレンス・ガイド』を参照してください。

ベンチマークの実行

作成したスクリプトを実行する際は、すべてのサーバーのプロセスを監視する必要があります。CPU および RAM の使用状況の他、I/O 読取りも確認します。また、`htmlexport` および Java プロセスなど、Content Server サーバー・プロセスもチェックします。さらに、データベースで使用されるプロセスはすべてチェックします。

また、テスト実行時のリソースの使用状況もチェックします。このタイプの調査は、使用するオペレーティング・システムに依存しますが、パフォーマンス・テストの実行時には、CPU に負荷をかけるプロセス、または Content Server リソースにアクセスするプロセスを停止することをお薦めします。この調査では、ウィルス・スキャンが実行中であるか、または他のプロセスがサーバーへのアクセスを試行していないかなどをチェックします。また、Content Server が使用するリソース（ファイル・サーバー、データベースまたは Web サーバー）を他のプロセスが使用していないか確認します。これは、パフォーマンスの結果に影響を及ぼします。

1 回目のベンチマークの実行後、より幅の広いパフォーマンスの数値を得られるように、次のようにシナリオを調整します。

- 最高の条件のシナリオでは、すべてのロギングを無効にし、投稿を許可しません。検索またはチェックインを実行し、所要時間を測定します。このシナリオでは、可能な範囲で最速の時間を得られます。
- 悪い条件のシナリオでは、検索キャッシュを無効にして、テスト・スクリプトを再実行します。
- 最低条件のシナリオでは、バッチ・テストと Web テストを同時に実行します。このシナリオでは、可能な範囲で最低の結果が得られます。

このようなシナリオの状況は、実際にはほとんど発生しませんが、予期される最高の結果と最低の結果を把握しておくことは重要です。たとえば、テストの実行がオフピーク時にスケジュールされていたとしても、緊急のバッチ処理によって、システムのパフォーマンスが著しく低下する場合もあります。起こる可能性のある状況を知っておくことが重要です。

ベンチマーク・テストの実行時には、可能なかぎり多くのタイミング・データを収集します。Content Server ログのタイムスタンプを分析し、Web サイト・テスト用に 1 ユーザーごとの 1 秒間当たりのリクエスト数を算出します。タイミングの数値を入手したら、結果の分析を開始できます。

結果の分析

結果分析の最初の手順は、その結果の値が遅いかどうかの判定です。ビジネス・コンテンツの操作（平均サイズの平均的なコンテンツ）の一般的な目標値は、次のとおりです。

- 読取り専用リクエスト : 20 リクエスト / 秒 /GHz/CPU。これは、不変の規則ではありません。レンダリングに 10 秒近くかかる読取り専用ページもあります。これは通常、ページで多様なリソースのコンテンツを組み立てているためです。読取り専用リクエストの数値は、ページの作成に必要な処理によって異なります。
- 生（RAW）の問合せ : 単純な検索の場合、4/ 秒 /GHz/CPU。これは、検索キャッシュを使用しない場合を想定しています。パフォーマンスは、使用しているセキュリティ・モデル、CPU、ファイル・システム、問合せの複雑性、およびデータの構成によって影響を受けます。このため、この数値は状況によって大きく異なる場合があります。
- 投稿 : 4 チェックイン / 秒 /GHz/CPU。この数値も平均的なビジネス・コンテンツを想定したものであり、より時間を要する複雑なコンテンツを想定したものではありません。投稿のパフォーマンスは、ファイル・サイズに加えて、主に、データベースおよびネットワーク・パフォーマンスによって制限を受けます。

得られた結果が、ここに示した数値よりも遅かった場合、パフォーマンス・チューニングを検討してください。問題の箇所を絞り込むには、「System Audit Info」ページでより多くのトレース・レポートを有効にする必要があります。

初回のベンチマークの実行結果により、問題の可能性のある領域が示されます。各領域について、この後の項で説明します。

検索および索引の問題領域

検索と索引は、別々の機能ですが関連しています。多くの場合、索引の最適化は、索引作成の速度を上げるものではなく、検索速度の向上につながる索引を作成するためのものです。

通常、索引固有のパフォーマンスの問題は、サイズの大きいドキュメント、あるいは暗号化またはサポートされていない形式などに起因します。これを確認するには、ログ・ファイル内のエラーをチェックし、次にデバッグを使用して問題解決を試行します。

次のベンチマーク結果は、検索またはインデクサ機能について考えられる問題を示しています。

- テンプレート・ページは速やかにロードされたが、単純な問合せの実行時間が 1 秒を超えている場合。
- ナビゲーション・ページが非常に複雑な問合せを実行している場合。`searchquery` トレースを分析して、実行内容を確認してください。動的なナビゲーションは、検索コレクションまたはデータベースのいずれかを使用できるため、これは明らかな問題ではありません。
- システムで複雑なセキュリティ・モデルが使用されている場合。複雑なセキュリティ・モデルでは、ユーザー・リクエストおよびセキュリティ・チェックの診断が不必要に強制実行されるため、検索に多くの時間がかかります。
- ユーザーのチェックイン、外部ポータル・アプリケーションのチェックイン、自動チェックイン、アーカイブのインポート、およびパブリッシャのチェックインなど、一般的な投稿が大量に行われた場合。検索のキャッシュおよびその他のパフォーマンスの問題は、システムが受信する投稿の量に影響を受けます。
- また、バッチローダを頻繁に使用した場合も、大量のコンテンツの入替え時に検索キャッシュでユーティリティの管理に問題が発生するため、パフォーマンスの低下が発生します。バッチロードは、使用するタイミングの制御は可能ですが、作成する投稿の量および要する時間の長さに特徴があります。バッチローダを使用する多くのシステムでは、Content Server 内のコンテンツのほとんどはバッチローダによってロードされたものです。

- 検索キャッシュの使用量が少ない場合。「System Audit Information」ページのキャッシュ統計をチェックしてください。キャッシュ・サイズが小さいまたはヒット数が少ない場合、検索キャッシュの使用量は少なくなります。ヒット率が 50% より低い場合、最適化の検討が有効です。単純なセキュリティ・モデルが使用されていて、かつ問合せの数が限定されている場合を除き、ヒット率 75% を継続的に超えるのは困難です。
- 検索コレクションがローカルにない場合。ネットワーク・ファイル・システムの速度が遅い場合、検索結果の出力も遅くなります。
- 同じサーバー上で大量のコンテンツの投稿および消費が発生した場合。

索引の最適化の詳細は、4-3 ページの「索引のチューニング」を参照してください。

データベースの問題領域

次のベンチマーク結果は、データベースの使用状況について考えられる問題を示しています。

- 長いキューに多くの接続が存在する場合、データベースへの問合せに時間がかかります。「System Audit Information」ページで、アクティブな接続および待機中の接続のリストを確認してください。接続待機中のスレッド、または処理に長い時間がかかっている問合せがある場合、問題が存在することを示します。通常どおり発行された問合せの実行時間が 2 秒を超えている場合、これは一般的にパフォーマンスの問題が原因です。

  さらに長い時間がかかる問合せもあります。たとえば、索引作成の問合せ、ワークフローの問合せ、またはレポートは、実行に比較的長い時間がかかります。この処理は、通常バックグラウンド・スレッドで実行されますが、頻繁に実行されるものではありません。その場合でも、1 分を超える実行時間は適切ではありません。
- 接続の切断エラー、または接続が多すぎることによるエラーがある場合、問題が存在する可能性があります。Web ログの接続エラーを確認してください。
- リクエスト時間のほとんどがデータベース内で費やされている場合、またはデータベースの CPU に負荷がかかりすぎている場合、データベースのレスポンスが遅くなる場合があります。`systemdatabase` トレース・レポートを確認して、問合せのタイムスタンプをチェックしてください。

データベース・チューニングの詳細は、4-6 ページの「データベースのチューニング」を参照してください。

ネットワークの問題領域

次のベンチマーク結果は、ネットワーク・インフラストラクチャのボトルネックを示します。

- ❖ リモート接続（特にデータベースへの接続）が遅いが、リモート・サービスの使用負荷が低い場合。`trafshow` ユーティリティまたは `Ethereal` を使用すると、ネットワークが飽和状態にあるかを判断できます。
- ❖ 問合せをデータベース上で直接実行する場合と、ネットワークを介して実行する場合を比較します。所要時間に著しい差異がある場合、ネットワーク接続に問題がある可能性があります。また、`systemdatabase` トレースを使用して、問合せのタイムスタンプを確認することもできます。

ネットワーク・チューニングの詳細は、第 6 章「システム・アーキテクチャのチューニング」を参照してください。

コア・エンジン /JVM の問題領域

公開またはページのレンダリングが遅い場合、コアの Content Server 構成に問題がある可能性があります。Content Server のデフォルト構成をチェックして、変更する箇所がないか確認してください。詳細は、4-2 ページの「構成フラグの調整」を参照してください。

次の状況が存在する場合、Java JVM に問題がある可能性があります。

- ❖ システムが大量の HCSP または JSP ファイルをレンダリングしている場合。
- ❖ システムが消費サーバーである場合。
- ❖ 大量のワークフローまたはサブスクリプションを実行している場合。
- ❖ 検索が実行されていないときに、Java VM ですべての CPU またはメモリーが使用されている場合。複数の CPU が完全に使用されていない可能性があります。Java VM をチューニングして、その他の CPU の存在を認識させる必要があります。

JVM の最適化の詳細は、4-9 ページの「Java のチューニング」を参照してください。

注意 : JVM の場合、小規模なベンチマーク・テストでは信頼性の高い結果は得られません。より大規模なベンチマークを実行することにより、ボトルネックの原因が JVM であるかどうかを判定しやすくなります。

ファイル・システム / メモリーの問題領域

次のベンチマーク結果は、ファイル・システムの問題を示しています。

- 予想よりも大きいファイル・アクティビティがある場合。I/O 読取りを監視して、発生しているスワップの量を確認してください。大量のスワップが発生している場合、RAM が不足しています。I/O スワップの量を確認する方法は、使用しているオペレーティング・システムによって異なりますが、一般的に、通常の RAM の使用量は物理 RAM より大きいため、スワップは頻繁に発生します。
- ファイル・システムが容量に近づいている場合、速度の低下が起こります。
- ファイル・システムがリモートにあるが、高パフォーマンスのネットワークが使用されていない場合。一般的に、SAN は非常に高速かつ高価で、設定に長い時間を要します。一方、一部の NAS は低速で安価であり、インストールも簡単です。通常、SAN では 2 つのシステムが同じファイルにアクセスできませんが、NAS ではアクセスできます。このため、クラスタリングなどのアプリケーションには、NAS やその他のネットワーク・ファイル・システムが必要になります。多くの場合、NAS は、複数の Windows マシンが同じ SAN を使用できるようにするために使用されます。

ファイル・システムの問題は、基盤であるネットワークの問題である場合があります。この 2 つは、多くの場合相互に関連しています。ファイル・システムのチューニングの詳細は、第 6 章「システム・アーキテクチャのチューニング」を参照してください。

Web セキュリティ・フィルタの問題領域

次の状況の場合、Web セキュリティ・フィルタ内にボトルネックがある可能性があります。

- ユーザーの最初のログインが完了するまで数秒かかったが、その後のページはすぐに表示された場合。
- 多くのユーザーがブラウザのホーム・ページにログインしたが、その後の操作が進まない場合。この場合、最初に高フィルタ・ロードが発生し、速度の低下が起こっています。`userstorage` トレース・レポートの詳細バージョンのタイムスタンプを確認してください。

内部および外部のセキュリティ、ならびにチューニングに使用できる構成オプションの詳細は、『Oracle Universal Content Management セキュリティおよびユーザー・アクセスの管理』を参照してください。

分析のヒント

ベンチマークの結果を分析する際は、次の点に注意してください。

- Java プロファイラ・レポートは、誤った解釈を生む場合があります。Content Server の幅広い知識と JVM の十分な理解がない場合は、Java プロファイラ・レポートの分析は行わないでください。
- ロギングとウィルス・スキャンは、システムの速度を低下させます。ベンチマークを実行する前に、すべてのログ出力およびウィルス・ソフトウェアを無効にしてください。
- 実際の状況に則したベンチマークを設定し、使用しているシステムが、消費主導なのか投稿主導なのかを特定します。検索キャッシュを無効にし、様々な操作のタイミングを注視してください。これにより、チューニングを行う際のポイントを判断しやすくなります。

4

CONTENT SERVER のコア・チューニング

概要

ベンチマークの実行、およびシステムの速度低下の分析を終えたら、Content Server のコアの動作を簡単にチューニングすることで、パフォーマンスの向上を図ることができます。この章では、Content Server のコア・チューニングに関する次の項目について説明します。

構成フラグの調整

パフォーマンス・チューニングを調整する最も簡単な方法は、Content Server 出荷時のデフォルト構成を調整することです。次に、変更可能な変数について説明します。詳細な説明およびその他の構成変数については、『Idoc スクリプト・リファレンス・ガイド』を参照してください。

- **インデクサ変数 :** いくつかの変数を使用して、インデクサの処理方法、および索引付けを行うレンディションの種類を変更できます。索引に関連する変数の完全なリストは、『Idoc スクリプト・リファレンス・ガイド』の第 5 章を参照してください。

  消費を最適化するには、索引は、可能なかぎり整理しサイズを小さくすることが重要です。ただし、これは投稿に対しては必ずしも最適であるとはかぎりません。実際のサイトの使用方法に基づいて、バランスの取れた方法を常に検討してください。

- **検索変数 :** 検索の接続数は、最小限にとどめることが重要です。通常、1CPU 当たり 1 つの接続が最適です。最終目的は、システムのスループットを最大限にすることです。各ハードウェア、ソフトウェアおよびネットワーク構成には、最大のスループットを可能にする検索およびデータベース接続の値があります。次の変数を使用して、接続数およびキャッシュの動作を設定できます。

  - `CachedResultRowCount`: 検索キャッシュのサイズを設定します。検索キャッシュでは、最新の検索エンジンの問合せをキャッシュすることにより検索のパフォーマンスが向上します。キャッシュのサイズを大きくすると検索時間は短縮されますが、メモリー使用量は増えます。
  - `MaxSearchConnections`: 同時に開ける検索接続数の最大値を設定します。

- **データベース変数 :** 可能なかぎり、データベース接続数は最小限にとどめてください。`NumConnections` 変数を使用して、データベースへのオープン接続数を設定します。デフォルトは 5 ですが、データベースが負荷に応じて過剰に使用される場合、より小さい値にすることも可能です。この設定は、コンテンツ・サーバー、およびスタンドアロンのアプリケーションやユーティリティに適用され、各アプリケーションでは、指定の接続数が使用されます。

- **その他の変数 :** 次の変数を使用して、キューおよび資格証明の情報を設定します。

  - `IdcServerSocketQueueDepth`: TCP/IP ソケット・キューの深度を指定します。
  - `UserCacheTimeout`: グローバルおよび外部ユーザー情報の一時キャッシュに対するタイムアウト値をミリ秒単位で設定します。

索引のチューニング

Content Server では、Verity、FAST または各種データベース索引ツールなど、多様な索引ツールが使用されます。各索引ツールは、フルテキスト索引（ファイル内のすべての単語に索引を付ける）およびメタデータのみの索引を提供します。フルテキスト索引は、メタデータ索引より処理に時間がかかりますが、より包括的な結果を得ることができます。使用する索引ツールは、インストールの前に実行対象のコンテンツ・サーバーおよび環境に基づいて選択します。

索引および検索機能は最適化できます。この項では、検索索引の使用方法の概要を説明します。詳細は、『システム設定およびプロセスの管理』を参照してください。

注意 : 索引のチューニングは、複数の異なる要因に大きく依存します。索引のパフォーマンス・チューニングに関しては、すべての状況に適合する全対応の推奨事項はありません。索引について大規模なパフォーマンス・チューニングを行う前にサポート担当者に連絡することをお薦めします。

検索索引の管理

管理者は、「Repository Manager」画面の「Indexer」タブを使用して、検索索引および検索コレクションの更新、再構築、構成または無効化を実行できます。

- 検索索引を更新するには、「Repository Manager」の「Indexer」タブを選択します。「Automatic Update Cycle」領域で**「Start」**をクリックします。
- コレクションを再構築するには、「Repository Manager」の「Indexer」タブを選択します。「Collection Rebuild Cycle」領域で**「Start」**をクリックします。
- 更新または再構築を構成するには、「Repository Manager」の「Indexer」タブを選択します。画面上の「Automatic Update Cycle」領域または「Collection Rebuild Cycle」領域のいずれかで**「Configure」**を選択してください。1 つのインデクサ・バッチ当たりのコンテンツ・アイテム数および 1 つのチェックポイント当たりのアイテム数を指定してください。1 つのバッチ当たりのアイテム数は、インデクサが一度に処理するファイルの最大数です。チェックポイント数は、すべての関連する索引状態を一度に通過するファイル数です。また、デバッグ・レベルも指定します。詳細は、『Managing Repository Content Guide』を参照してください。
- 特定のファイルでのフルテキスト索引を無効にするには、「Configuration Manager」で**「application/noindex」**という名前のファイル形式を定義します。「System Properties Options」タブで、**「Allow override format on check in」**を選択します。索引付けを行わないファイルをチェックインする際は、ファイル形式として**「application/noindex」**を選択するようにユーザーに指示してください。ただし、ファイル形式を変更すると、ブラウザでファイルを自動的に開けなくなります。

SelectivelyRefineAndIndex コンポーネントを使用すると、どのドキュメントを変換するか、およびどのドキュメントに索引付けを行うかを制御できます。このコンポーネントの使用方法の詳細は、コンサルティング・サービスに問い合せてください。

- `IndexerLargeFileSize` 変数を使用すると、インデクサがバッチにファイルを配置する際のファイル・サイズを MB 単位で設定できます。この設定よりサイズが大きいファイルは、別のバルクロードで索引付けされます。
- 再構築サイクルで新しいドキュメントをコレクションに追加できるようにするには、`AllowConcurrentUpdate` 構成変数を True に設定します。

フルテキスト索引について

フルテキスト索引を適用すると、メタデータだけではなくファイル内のすべての単語が索引付けされます。サポートされている形式の詳細、およびここで説明するフルテキスト索引ツールの詳細は、『システム設定およびプロセスの管理』を参照してください。

注意 : フルテキスト索引は、多くの場合、検索コレクションまたは検索コレクション索引と呼ばれます。コンテンツ・サーバー・インスタンスでは、1 つの索引コレクションが使用されます。ユーザーがドキュメントを追加すると、そのドキュメントは同じコレクションに追加されます。コレクションを再構築すると、コンテンツ・サーバーでは新しいコレクションが作成されます。アクティブにできるのは、常に 1 つのコレクションのみです。

アドオン統合を使用した索引作成

Verity Integration および FAST は、使用しているデータベースを問わず、Content Server で使用されるアドオン索引統合です。これらのツールでは、指定した形式のすべての変換ファイルにフルテキスト索引を適用するかどうかを選択できます。前に説明したように「System Properties」の形式の上書きオプションを有効にすることで、フルテキスト索引作成で索引付けを行うかどうかをコントリビュータが指定できるようになります。

ユーザーがメタデータまたはキーワードでコンテンツの検索を行うと、データベースではなく検索索引に対して問合せが発行されます。結果は、メタデータ・フィールドのいずれか、または検索エンジンに割り当てられている関連スコアに基づいてソートされます。

これらの索引アドオンの詳細は、『Choosing a Search Solution』を参照してください。

データベース・フルテキスト索引

データベース・フルテキスト索引を使用するには、次の行をコンテンツ・サーバーの *<install_dir>*/config/config.cfg ファイルに追加します。

```
SearchindexerEngineName=DatabaseFullText
```

メタデータのみの検索を使用する場合は、次のように指定します。

```
SearchindexerEngineName=Database
```

同じシステム内でこれら両方の設定を使用しないでください。

その他のツールについては、フルテキスト索引は、特定の形式に変換されたすべてのファイルにデフォルトで適用されます。使用可能な形式のリストは、『システム設定およびプロセスの管理』を参照してください。

IDOC スクリプト・キャッシュ

Idoc スクリプトは、クライアント側ではなくサーバー側で評価されます。このため、ページ要素の処理は、ブラウザがリクエストした後、リクエストされたページがクライアントに返される前に実行されます。

Idoc スクリプト・ページ内で実行されるアクティビティで最も時間のかかるアクティビティの 1 つは、ページ・コンテキスト内でのサービス・コールの実行です。このアクティビティには、検索およびデータベース問合せなどのアクティビティを実行する際の、サーバーに対する追加処理が含まれます。Content Server では、一部の問合せがキャッシュされますが、コンテンツ・アクティビティのためにキャッシュが意図したとおりに有効に動作しない場合があります。

Idoc スクリプト・キャッシュを使用すると、個々のインクルードをカプセル化し、ユーザーごとに、またはアプリケーション全体に対して一定時間、キャッシュを保存できます。

`cacheInclude` 構成変数を使用すると、キャッシュの Dynamic HTML フラグメントを含めることができます。これは、グローバル・キャッシュ、名前付きキャッシュまたはセッション・キャッシュなど、複数回使用されるフラグメントの場合に役立ちます。

`setMaxAge` 変数は、ページ全体のキャッシュに使用できます。ページのキャッシュには、キャッシュ制御 HTTP ヘッダーが使用されます。これは、頻繁に使用されるページの場合に役立ちます。

データベースのチューニング

重要 : データベースのチューニングは、データベースのフルテキスト索引のチューニングとは異なる問題です。データベースのチューニングでは、データベース・ソフトウェアの最適化を行います。フルテキスト索引のチューニングでは、検索コレクションの管理、および使用するデータベースに合せた検索コレクションの最適化を行います。

Content Server と連携するデータベースをチューニングするには、いくつかの方法があります。

- ファイルの I/O パフォーマンス（特に索引のパフォーマンス）を最適化します。これには、半導体ディスクが有効です。
- 検索頻度の高い列に索引を追加します。索引を追加する列を特定するには、「System Audit Information」ページの `systemdatabase` トレース・レポートを使用します。このトレース・レポートを使用すると、主にどの列が検索されているかがわかります。
- カスタム・メタデータ・フィールドが検索に使用されている場合、これらのフィールドに索引が作成されていることを確認します。キーワードを含むかどうかの検索にはフルテキスト索引を使用し、単純一致の検索には通常の索引を使用します。
- 大量のコンテンツを追加する前および後など、データベースの拡大時にはデータベース統計を更新します。これを行うことにより、頻繁に使用される項目を、データベースで認識することができます。このプロセスは、データベースごとに異なります。たとえば、SQL Server はデフォルトでこの処理を実行します。詳細は、各データベースのドキュメントを確認してください。
- 使用しているデータベースの行ロック・アルゴリズムを最適化します。たとえば、Oracle では、Content Server と適切に連携しない可能性があるデータベースの使用方法について、様々な想定を行っています。Sybase を使用する場合、<*install_dir*>/database/Sybase/admin ディレクトリに提供されている SQL スクリプトを使用して、行ロックを最適化できます。
- サイトのニーズに固有のカスタム・サービスおよびカスタム・コンポーネントの作成を検討します。たとえば、パフォーマンスの向上のためにデータベース固有の問合せを作成したり、大量のコンテンツ・バッチのチェックインを簡略化するサービスを作成します。詳細は、『Working with Components』を参照してください。
- Oracle Query Optimizer コンポーネントをサポート・サイトからダウンロードできます。このコンポーネントは、ユーザー問合せの非効率な箇所を削除することで Oracle データベースのパフォーマンスを改善します。このコンポーネントを使用すると、検索を効率よく行うためのヒントを問合せに追加できます。また、このコンポーネントにより、Oracle の索引機能を最大限に活用できます。

❖ Oracle の使用時には、次の方法も検討してください。

- 問合せのレスポンス時間の改善を図ります。次のコマンドを使用すると、問合せが完了する前に最初の行を可能なかぎり速く取得できるように問合せを最適化できます。

  `optimizer_mode=first_rows`
- 表のパーティショニングの使用を検討します。パーティションのサイズは、データ配分およびハードウェア構成によって異なります。7.5 より前のバージョンの Content Server を使用している場合は、メタデータ構造が非常に安定している場合を除き、DocMeta 表のパーティショニングは行わないでください。
- `Cursor_Sharing=SIMILAR` を設定します。デフォルトは、`Cursor_Sharing=EXACT` です。この変更を行うことをお薦めします。

注意 : ここで説明した内容は、Content Server に固有のものです。その他のデータベース・チューニング・オプションについては、それぞれのデータベースのドキュメントを参照してください。

データベースの一般的なメンテナンス

使用しているデータベースの種別にかかわらず、最適なパフォーマンスを維持するために実行できるメンテナンス・プランがあります。この項では、定期的に行う必要のある各メンテナンス・タイプの概要を説明します。詳細は、データベースのドキュメントを参照してください。

バックアップ・プランニング

適切なバックアップおよびリカバリ・プランは、データベースの状態を維持する上で非常に重要です。各データベースに必要なバックアップのタイプは様々です。実質的に静的なデータベースの場合、頻繁にバックアップを取る必要はありませんが、夜間または週に 1 回全体のバックアップを取る必要があります。ソース・トランザクション・データベースから簡単に再構築できるデータベースの場合、バックアップは必要ありません。バックアップは、パフォーマンスへの影響を最小限に抑えるために、可能なかぎり短時間で行うのが理想的です。ただし、バックアップの実行時にデータベースへの排他的アクセスが必要なバックアップもあります。つまり、この間ユーザーはデータベースに接続できず、作業を行うことができません。

一般的に、バックアップは、データベース作業への影響が最小限であるときに実行します。

システム・メンテナンス

データベースのパフォーマンスは、監視および調整（必要な場合）が必要な、いくつかのキーとなるメンテナンス操作に依存します。

- 領域 : データベースごとに、監視が必要なシステムレベルの領域構造（表領域、データベース領域など）は異なります。異常なデータベース領域または表領域が再生され再編成されると、パフォーマンスは改善されます。
- データベースの整合性 : データベースの整合性により、データベース内の基礎となるファイル構造の可用性、データベースの構造的な完全性、データ・ディクショナリ自体の一貫性、およびデータ・ディクショナリと付随するデータベース・オブジェクトとの一貫性が確保されます。データベースごとに、この検証の実行に使用するコマンドとツールは異なります。
- メモリー : 表または索引が使用中に断片化されるのと同様に、データベース・サーバーで使用されるメモリーも断片化されます。様々なキャッシュの要素はディスクから再ロードする必要があるため、多くのメモリー・メンテナンス機能にはよくない面もあります。このため、頻繁に使用されるオブジェクトおよびデータがメモリーにリストアされるまで、初めのうちはパフォーマンスに悪影響が出ます。

その他のデータベース・メンテナンス

データベースは、表、索引およびその他のオブジェクトで構成されます。メンテナンス・プランの一部として実行する必要のある、多様なオブジェクト形式のメンテナンス操作があります。

- 領域のフラグメンテーション : ほとんどすべてのデータベース・エンジンは、時間とともに拡大および縮小する表および索引の再編成機能を備えています。
- 表の再編成 : 表内の行を領域のブロックに展開する必要があるが、空き領域が不足しているために展開できない場合、表の再編成が必要になることがあります。行は、十分な空き領域のある別のブロックまたはページに再配置され、レコード・ポインタはそのまま残ります。データベースで行が必要になると、ポインタが読み取られ、次に行自体が読み取られます。このような表では、適切な行を読み取るためにデータベース・エンジンが余分な I/O 作業を実行することになるため、パフォーマンスに悪影響を及ぼします。
- 索引の再編成 : 大きいアクティビティのある索引は断片化される場合があり、その結果、スキャンおよび操作時間が長くなります。良好なパフォーマンス結果を得るには、定期的に索引の再構築を行う必要があります。

また、データベース・メンテナンスを実行する際、この他に 2 つの側面を検討する必要があります。1 つ目は、プラン依存性リストです。このリストは、データベース・メンテナンス・プランの一部が、プラン内の別の部分の実行が正常終了していることを前提にする場合などを考慮します。たとえば、整合性チェックによってデータベースが正確であることが確認されたときのみ、バックアップを実行するというような場合です。バックアップのスケジューリングは、有効な整合性チェックに依存します。

もう 1 つの側面は、通知の使用です。スケジュールされているプランの実行に失敗した場合、または予期しない結果が起こった場合、操作を確認できるように適切なユーザーに通知が送信されます。

JAVA のチューニング

Java 仮想マシン（JVM）は自己チューニング型のシステムですが、Content Server の intradoc.cfg ファイルにチューニング・パラメータをいくつか追加できます。たとえば、intradoc.cfg ファイルの次の設定を使用して、メモリーおよび Java コンパイラを最適化できます。

```
HEAP_OPTIONS=-Xmx512m -Xms512m -server
```

注意 : スタンドアロン・アプレットを使用しない場合、`-server` オプションのみを使用してください。

使用可能な RAM が十分にある場合、512m の値を 1000m または 2000m に増やすことができます。通常、これは、検索キャッシュが増加されている場合、またはフォルダのキャッシュを使用していて多くのフォルダがある場合にのみ必要になります。

Java のメモリー不足エラーは致命的なエラーです。このエラーが発生した場合、多くの場合 `Xmx` の値を増やす必要があります。

複数の CPU から構成されるシステムを使用している場合、Java バージョン 1.4.1 以降を使用し、複数 CPU 用のガベージ・コレクタをチューニングする必要があります。Java アプリケーションのチューニングの詳細、および Java のドキュメントについては、http://java.sun.com/docs/performance/ を参照してください。

その他のチューニング・オプション

前述の項目以外にも、パフォーマンスのチューニングに役立つよう、Content Server のコア機能にいくつかの変更を加えることができます。

- Publisher または Site Studio から配信されるページを簡略化できます。ページごとに複数のリクエストが実行されるページもありますが、そのリクエストの多くは不要なものです。問合せの数や複雑性を減らし、タイムスタンプやナビゲーションの問合せなど、キャッシュされない問合せの使用は避けてください。
- Content Server 7.5 以降では、データベース検索を使用して完全一致を検索できます。この機能は、検索パフォーマンスのチューニングを行った後、およびデータベースがロードされていない場合にのみ行ってください。

  データベース検索を行うには、GET_SEARCH_RESULTS サービスに対する次のパラメータを設定します。

  ```
  SearchEngineName=DATABASE
  ```

  データベース検索機能を使用するには、検索対象のテキストを指定する `QueryText` パラメータを設定してください。

  ```
  QueryText=dDocType LIKE 'ADACCT'
  ```

  これにより、SQL に似た問合せを使用してデータベースを直接検索できます。
- `eval` 機能の使用を避けます。かわりに、`setResourceInclude` または `setValue` を使用してください。`setResourceInclude` を使用すると、動的に作成されるスクリプトがインクルードに割り当てられます。`setValue` を使用すると、キーに基づいてターゲット領域に値が設定されます。
- フルテキスト検索を使用しないようにするか、条件付きのフルテキスト索引（カスタマイズ機能）の使用を検討します。フルテキスト検索では、より多くのリソースが使用され、多大な時間がかかる場合があります。フルテキスト検索が必須ではない場合、メタデータのみの検索を実装できます。こうすることでキャッシュされた検索を利用できるため、オーバーヘッドが減少します。

  また、別の Content Server インスタンスを使用して、フルテキスト索引を必要とするコンテンツと必要としないコンテンツを分けることができます。その結果、パフォーマンスの負荷が軽減されますが、この方法の場合、より多くのライセンスが必要となります。
- Dynamic Converter ツールを使用すると、必要に応じてあるいはチェックイン時にコンテンツを HTML に変換できます。ポータル統合デプロイメントで Dynamic Converter を使用すると、消費環境で大容量のアクティブ・コンテンツの変更を削除できます（Site Builder や Publisher を使用してアクティブな消費環境にコンテンツを変換および再公開する場合）。

- ピーク時に発生するチェックイン数を減らし、Content Publisher による公開をオフピーク時に制限するようにします。
- 検索トランザクションを減らすもう 1 つの方法は、コンテンツ・ナビゲーションの事前生成です。問合せに対するコンテンツ・ナビゲーションを必要に応じて生成すると、Content Server 内でより多くの検索トランザクションが発生します。一部のナビゲーション入力は事前に生成できます。その結果、システム内で発生する検索トランザクション数が減少します。
- ポートレットでは、Content Publisher を使用して Content Server からファイル・システムにコンテンツを公開できます。また、ポータルを使用して、公開された場所からコンテンツを取得することもできます。この場合、ポータル・アプリケーションと Content Server 間の依存性が排除されるため、パフォーマンスが向上します。フルテキスト検索は、ポータルで管理する必要があります。また、ポートレットは、ローカルに再度書込みされる必要があり、静的なサイトのコンテンツを表示します。
- サイトのセキュリティ構成を調査します。セキュリティ・モデルが複雑になると、作成される問合せが長くなり、問合せの実行にかかる時間も長くなります。たとえば、次の問合せを実行するとします。

  ```
  dDocName <matches> 'test'
  ```

  実行される実際の問合せは次のとおりです。

  ```
  (dDocName <matches> 'test') <and>
  (dSecuritiyGroup <not> 'Secure')
  ```

  10 個のセキュリティ・グループを設定しており、各ユーザーがその 10 個のうち 5 個のグループにアクセスできる場合、検索時にすべてのセキュリティ・グループがチェックされるため、検索処理は必要以上に複雑になります。
- サイトで使用している Web サーバー・フィルタを確認します。Content Server では、ブラウザを通してページを配信するための Web サーバーが必要です。Web サーバー・フィルタは、ユーザー・リクエストを認証するためにインストールされます。Web フィルタ・オプションは、「Administration」トレイの「**Filter Administration**」オプションをクリックすることにより変更できます。

  「Configure Web Server Filter」ページのオプションを使用して、gzip 圧縮の無効化、使用する認証タイプの設定、キャッシュのタイムアウト値の設定、およびフィルタと Content Server 間で送信されるデータおよびヘッダーのロギングの有効化を行うことができます。

  Web サーバー・フィルタの構成の詳細は、『Oracle Universal Content Management セキュリティおよびユーザー・アクセスの管理』を参照してください。

5

アドオン・コンポーネントのチューニング

概要

Content Server のコア部分のチューニングを実行した後は、Content Server のアドオン・コンポーネントのチューニングに進みます。ここでは、アドオン・コンポーネントのチューニングに関する次の項目について説明します。

バッチローダ

Batch Loader ユーティリティは、コンテンツ・サーバー内の大量のコンテンツを一括で挿入、削除または更新するために使用します。次の提案は、バッチローダ使用時のシステムのパフォーマンスの向上に役立ちます。

- 比較的小規模なシステムの場合、バッチロードの開始前にその他のアクティビティを一時的に無効にします。たとえば、自動更新インデクサ・サイクル、Inbound Refinery および PDF Converter を無効にします。これらの機能はすべて、バッチロードの後に自動的に実行されます。したがって、バッチロードの開始前に無効にすることで時間を節約できます。

注意 : 同時処理が可能な大規模なマルチプロセッサ・システムの場合、他のアクティビティを無効にする必要はありません。この場合、バックグラウンド・アクティビティをオフにしないでください。

- Verity を使用している場合、バッチの挿入前に検索コレクションを最適化します。詳細は、4-3 ページの「索引のチューニング」を参照してください。
- データベースに提供されているツールを使用して、バッチロード操作時のデータベースの使用状況を分析します。データベース問合せオプティマイザを改善できる箇所がないか確認します。データベースの使用状況を確認するには、まず小さいサイズのバッチ挿入を行います。

DYNAMIC CONVERTER

Dynamic Converter は、ネイティブ・ビジネス・ドキュメントの HTML、XML および無線形式へのオンデマンド公開に使用します。

Dynamic Converter の処理速度を上げる簡単な方法は、テンプレートの使用です。テンプレートを使用するには、「Dynamic Converter Admin」メニューの「**Template Selection Rules**」を選択します。テンプレート・ルールにより、変換およびレイアウトが指定されます。実行時、コンテンツは変換およびキャッシュされます。この後にコンテンツを取得するユーザーは、すでに変換済のファイルを取得するため、変換処理が完了するのを待機する必要がありません。

SITE STUDIO

Site Studio は、Web サイト作成の自動化に使用し、WYSIWYG 形式ベースの環境でのコンテンツの投稿を可能にします。複数の様々な方法を使用して、Site Studio のパフォーマンスを拡張できます。

- ネイティブ・コンテンツを使用して Web ページを生成する場合、Dynamic Converter がバックグラウンドで使用され、コンテンツが変換されます。したがって、「Template Selection Rules」を使用してコンテンツをあらかじめ変換しておくと、レスポンス時間が短縮されます。
- コンテンツ配信に使用される、動的リスト内の問合せを最適化します。問合せは単純なものにします。一致条件に、日付スタンプの他にタイムスタンプを使用することは避けてください。タイムスタンプが一致することはほとんどありません。
- ページ上で実行されるサービスを最適化します。実行中のサービス数を確認し、ページ配信に必要なサービスのみが実行されるようにしてください。不要なサービス、処理またはデータは削除してください。
- 可能な場合、ナビゲーションには動的リストではなく静的リストを使用します。静的リストを使用してページをレンダリングする場合、情報を取得するためにサーバーに戻る必要がないため、レスポンス時間が速くなります。
- 可能であれば、クライアント側のページ構築機能を使用します。提供されている多くのフラグメントでは、サーバー側のフラグメントではなく、クライアント側のメカニズムを使用して、ナビゲーションを構築します。クライアント側のフラグメントでは、サイト・ナビゲーションをより速く構築できます。

詳細は、Site Studio のドキュメントを参照してください。

PUBLISHING UTILITY/CONNECTION SERVER

Publishing Utility は、コンテンツ・サーバー環境からピュア Web サーバー環境への Web サイトの公開を可能にする統合インタフェースを Site Studio に提供します。Publishing Utility は、Content Server とは別のエンティティとして JVM で実行される Connection Server に基づいています。このため、JVM のチューニングと同じ方法で Connection Server をチューニングできます。詳細は、4-9 ページの「Java のチューニング」を参照してください。

Publishing Utility のパフォーマンスのチューニングに使用できるその他の方法として、次のものがあります。

- JVM の最大ヒープ・サイズを増やします。デフォルトは 384MB です。これを、最小でも 512MB に増やしてください。これを行うには、*<install_dir>*/bin/intradoc.cfg ファイルの `JvmCommandLine` 設定をコメントアウトし、`JAVA_OPTIONS` 設定を次の例のとおりに変更します。

  ```
  #JvmCommandLine=/home/contribution/shared/os/<os>/j2sdk1.4.1/bin/
  java -cp $CLASSPATH $STARTUPCLASS
  JAVA_OPTIONS= -Xmx512m
  ```

  詳細は、『Troubleshooting Guide』を参照してください。

- マルチプロセッサ・システムの場合、ガベージ・コレクション機能で使用するスレッド数を増やします。
- Connection Server は、専用のマシンで稼働させます。これは、頻繁に公開を実行する必要がある大規模なサイトの場合に特に有効です。
- すべてのトレース・ログを確認して、発生したソフト・エラーを解決します。ソフト・エラーは、Publishing Utility を停止するようなエラーではありませんが、処理速度の低下の原因となる場合があります。
- Site Studio ページのレンダリングを最適化します。Site Studio ページの最適化のヒントは、5-3 ページの「Site Studio」を参照してください。

INBOUND REFINERY

Inbound Refinery では、複数の異なる製品が使用されます。これらの製品は、全体のパフォーマンスを向上させるために、それぞれ最適化できます。

- PDF Converter は、Inbound Refinery と連携して使用されるアドオンで、ドキュメントを PDF 形式に変換します。リリース 7.0 の時点では、組込み PDF 変換テクノロジとして JAWS エンジンが使用されています。様々な JAWS オプションを設定し、速度の調整を行えます。速度と品質のバランスが取れた設定を見つけることが重要です。詳細は、『PDF Converter Administration Guide』を参照してください。
- Inbound Refinery は、Content Server とは別のサーバーに配置し、それぞれが専用のリソースを利用するようにします。
- 複数のリファイナリを実行し、選択的キュー処理を利用します。この処理は、各リファイナリのファイル・タイプの定義に使用します。たとえば、リファイナリ 1 を PowerPoint 用に最適化し、リファイナリ 2 を TIFF 変換用に最適化できます。すべてのリファイナリは同時に実行できるため、1 つの大規模なドキュメントによって、後に続くドキュメント用のキューが占有されることはありません。

CONTENT PUBLISHER

Content Publisher は、ビジネス・コンテンツ・フォーム（MS Office または Lotus Suite で作成されたフォームなど）を、HTML、XML および無線形式の完全にリンクされた Web サイトおよびページに自動的に変換および公開する、高度なテンプレートベースのテクノロジを提供します。

Content Publisher を使用すると、プロジェクトの作成、およびこのプロジェクトの手動変換または自動変換を行えます。Publisher の使用を最適化する方法は、次のとおりです。

- プロジェクトをマスター・プロジェクトとサブ・プロジェクトに分割します。サイト内で頻繁に更新する必要がある領域は、指定の頻度で更新するようにスケジュールできます。このため、Site Server を何度も実行する必要がありません。
- ボトルネックを特定し、それが Content Server の検索またはデータベースではなかった場合、別の Publisher を別のマシンに配置してください。これにより、プロジェクトを同時に処理できるようになります。
- ディレクトリ項目の問合せを最適化します。詳細は、5-4 ページの「Publishing Utility/Connection Server」および 5-3 ページの「Site Studio」を参照してください。
- 変換を実行する前に、スタイル数および不要な HTML 要素の数を減らします。HTML は、使用しやすいように変換時に最適化されます。

CONTENT INTEGRATION/PORTAL SUITE

Integration Suite は、エンタープライズ・アプリケーションと統合するためのスケーラブルな統合インフラストラクチャを提供します。Content Portal は、すぐに使用できる、または Integration Suite でのポートレットの実装方法の例として使用できる参照ポートレットを提供します。

Integration Suite および Portal Suite を最適化する主な方法は、キャッシュを最大限に利用することです。サービスレベルのキャッシュが使用可能ですが、デフォルトでは DOC_INFO のみキャッシュされます。パブリック・データであり、ユーザーまたはセキュリティに関連付けられていないデータ（お知らせおよびニュースなど）のキャッシュを試してみてください。

この目的のために、ページレベルの HTTP キャッシュおよびアプリケーション・サーバー固有のキャッシュを使用してください。

最適化支援ツール

パフォーマンスの改善、および最適化の必要な箇所の特定を支援するアドオン・ツールがいくつか用意されています。

Content Tracker

Content Tracker は、コンテンツおよびサーバー・アクセスの監視に使用します。また、Content Tracker を使用して、データを抽出しデータベース・リポジトリにロードすることもできます。このデータは、別のレポート・ツールを使用して表示できます。

Content Tracker では、どのコンテンツが頻繁に使用されているかを特定できます。このため、不要な項目および使用されていない項目を削除し、リポジトリのサイズを小さく保てます。

Content Tracker を使用することにより、ユーザーがどのようにリポジトリを検索しているかを把握でき、その情報に基づいて検索問合せを最適化できます。

また、Content Tracker の情報は、統計分析を実行するために別のデータベースに送信できます。このため、データベースのリソースを他の用途のために解放できます。

Compression

Content Server Compression は、圧縮済 MrSID 形式に自動的に変換される、サイズの大きいイメージ・ファイルのチェックインに使用されます。イメージはサムネールのように表示され、コンシューマは、そのイメージを高解像度で拡大できます。

このタイプの圧縮では、地図、衛星写真および X 線写真など、サイズの大きいイメージのファイル・サイズを縮小できます。イメージ・ビューでは、圧縮の幅は低下します。

6

システム・アーキテクチャのチューニング

概要

Content Server のパフォーマンスの問題は、基盤となるアーキテクチャの問題によって発生している場合があります。この章では、アーキテクチャのチューニングに関する次の項目について説明します。

パフォーマンス改善のための一般的なヒント

次のチェックリスト項目を使用して、Content Server のリポジトリおよび操作を処理するためにアーキテクチャが適切に構築されているかを検証できます。

- 使用しているオペレーティング・システムおよびその他の製品について、常に最新の更新およびパッチをインストールします。多くの場合、アップグレードを行うと、カスタマイズ機能を追加することなくパフォーマンスの改善が図れます。
- CPU の数を確認します。CPU 使用量に対する最も大きな影響は、検索時および Web ページの構築時に発生します。システム上の CPU リソースが不足していると、パフォーマンスに悪影響を及ぼします。
- 1CPU 当たり 1GB のメモリーを使用します。この数値は、マシン上の Content Server のインスタンス数によって異なります。Content Server は 32 ビット JVM 上で実行されるため、1 つのインスタンスは、16 以上の CPU を使用できても、4GB より多くの RAM を使用できません。可能な場合は、メモリー・エラーが発生しないかぎり、Java 仮想マシンで自己チューニングを実行するようにしてください。
- Content Server が配置されていない別のマシンにアプリケーションを移動します。Web サーバーは例外です。Web サーバーは、Content Server と同じマシンに配置します。可能な場合、Refinery、Publishing、Application Server およびその他のアプリケーションは、他のマシンに移動します。
- ネットワークのセグメント間のトラフィックを最適化します。可能な場合、トラフィックの遅延を最小限にするために、サーバーは同じサブネット内に配置してください。
- サイズの大きいコレクションの場合、Enterprise Search 機能付きのコレクションの Content Server インスタンスを複数使用することを検討します。
- 消費量が多いシステムの場合、垂直方向ではなく水平方向に運用方法を拡張します。水平方向の拡張では、クラスタリング、つまり複数のノードおよび新規サーバーの設定を行います。垂直方向の拡張では、既存のサーバーへのハードウェア（CPU および RAM）の追加を行います。消費量が多いサイトの場合、垂直方向の拡張の方がより効果の高いパフォーマンスの改善を図れます。

  垂直方向の拡張でも水平方向の拡張でも、ハードウェアの追加を行います。単一のインスタンスの垂直方向の拡張の値は JVM スレッドの同期化によって制限される場合があるため、水平方向の拡張の方がより効果的な場合があります。ただし、複数の Content Server インスタンスで垂直方向の拡張を使用することは可能です。いずれの場合も、拡張方法は経済的な意思決定により決まります。

- プロキシ・サーバーおよびリバース・プロキシ・サーバーを使用して、ネットワーク・トラフィックを最適化し、CPU およびデータベースの負荷を分散します。プロキシ・サーバーでは、頻繁にリクエストされるページのコピーを 1 つの場所にキャッシュすることにより、ページのレスポンス時間が短縮されます。ページ・リクエストが送信されると、リクエストをサーバーに渡すのではなく、プロキシ・キャッシュのページが使用されます。プロキシの使用の詳細は、『Reverse Proxy Resource Guide』および『Proxy Connections Component Administration Guide』を参照してください。
- クラスタを設定する前に必ず最適化を行います。クラスタリングに時間と費用を投資する前に、その他のパフォーマンスの改善を実行しておくようにしてください。場合によっては、クラスタリングによって問題が解決されないだけではなく、問題が悪化する場合があります。クラスタ化された環境での Content Server の設定の詳細は、『Clustering Concepts Guide』を参照してください。

もう 1 つの重要な要因は、システム要素の場所です。パフォーマンスの速度は、システム要素が離れた場所に配置されている場合に低下します。次の図は、システム要素の場所に基づくパフォーマンスの違いを表しています。

図 6-1　　システムの場所に基づくパフォーマンスの違い

パフォーマンス	場所		
	Web サーバー	ファイル・システム	データベース
最も速い	ローカル	ローカル	ローカル
	ローカル	ローカル	リモート
	ローカル	SAN	リモート
	ローカル	NAS	リモート
	ローカル	リモート	リモート
	リモート	ローカル	リモート
	リモート	SAN	リモート
	リモート	NAS	リモート
最も遅い	リモート	リモート	リモート

このように、システム要素間の距離が長くなるほど、パフォーマンスは低下します。

記憶域の問題の緩和

Content Server インスタンス内のコンテンツの量が増加すると、vault、weblayout および検索ディレクトリによってディスク領域が大量に使用されます。このため、Content Server が当初インストールされていたドライブ上の使用可能領域が不足する場合があります。

これらのディレクトリの一部またはすべてを、領域がより大きい別のドライブに移動することで、記憶域の問題は緩和されます。非ローカル・ドライブ（ネットワーク・ドライブなど）を使用することができますが、パフォーマンスに影響を及ぼす可能性があります。Content Server では、非ローカル・ドライブにあるすべてのファイルに中断なくアクセスする必要があります。これらのファイルへのアクセス遅延は、特に検索ディレクトリの場合、Content Server のパフォーマンスに直接的な悪影響を及ぼします。したがって、可能な限りローカル・ドライブを使用してください。

これらのディレクトリの移動方法の詳細手順を次に示します。手順内で示されているファイルを変更する前に、構成ファイルのバックアップを必ず取ってください。ファイルの修正には、任意のテキスト・エディタを使用できます。

重要 : これらのディレクトリを NAS ドライブなどのネットワーク共有ドライブに移動する場合、操作を実行するユーザーは、このネットワーク・ドライブへの読取り / 書込みのフル・アクセスが必要です。アクセス権限がない場合、Content Server はファイルにアクセスできず、ユーザーはファイルを使用できません。

Vault フォルダの場所の変更

1. Content Server を停止します。
2. Vault ディレクトリを新しい場所に移動します。たとえば、*d:/vault/* と指定します。
3. `VaultDir=d:/vault/` という設定を、次の構成ファイルに追加します。
   - <*install_dir*>/bin/intradoc.cfg
   - <*refinery_dir*>/connections/main/intradoc.cfg（このファイルは Inbound Refinery を使用している場合にのみ表示されます。）
4. Admin Service、Content Server インスタンスおよび Inbound Refinery（使用している場合）を再起動します。

Weblayout フォルダの場所の変更

1. Content Server を停止します。
2. Weblayout ディレクトリを新しい場所（*d:/weblayout/* など）に移動します。
3. Content Server Web サイトの Web サーバー（IIS、Apache、Sun One）で、「Virtual Directory」タブのローカル・パスを新しい weblayout の場所に変更します。
4. Web サーバーを再起動します。
5. `WeblayoutDir=d:/weblayout/` という設定を、次の構成ファイルに追加します。
   - <*install_dir*>/bin/intradoc.cfg
   - <*install_dir*>/admin/bin/intradoc.cfg
   - <*refinery_dir*>/connections/main/intradoc.cfg（このファイルは Inbound Refinery を使用している場合にのみ表示されます。）
6. Admin Service、Content Server インスタンスおよび Inbound Refinery（使用している場合）を再起動します。

検索索引の場所の変更

1. Content Server を停止します。
2. 検索ディレクトリを新しい場所（*d:/search/* など）に移動します。
3. `SearchDir=d:/search/` という設定を、次の構成ファイルに追加します。
   - <*install_dir*>/bin/intradoc.cfg
4. Content Server インスタンスを再起動します。

よくある間違い

インフラストラクチャの設定時に起こしやすい間違いについて、次に説明します。

- 不要なファイアウォールおよびウィルス・スキャン。ファイアウォールの使用状況を調査し、システム・セキュリティに必要不可欠ではないファイアウォールは削除するか、レベルを下げてください。
- 1 つのサーバーを過度に使用している。1 つのサーバー上で実行しているアプリケーションの数が多すぎる場合、すべてのアプリケーションのレスポンス時間が遅くなります。サーバー・リソースを調査し、必要に応じて調整してください。
- ネットワーク・アーキテクチャに、ルーター、ハブおよびスイッチなど不要なレイヤーがある。すべてのアプリケーションおよびサーバーは同じサブネット上に配置することが重要です。
- バックアップおよびアーカイブを実行するタイミングが不適切である。システムに大きな負荷がかかるアクティビティはすべて、システムを使用するユーザーがほとんどいないオフピーク時に必ず行います。
- サーバーとファイル・システム間のパイプラインが不十分である。多くのシステム設定では、サーバーとデータベース間の速度に重点が置かれるのに対し、Content Server では、コンテンツとファイル・システム間がより高速である必要があります。サーバーとファイル・システム間のパイプラインが適切であることを確認してください。

索引

い

う

か

き

く

け

こ

さ

し

す

せ

そ

ち

て

と

ね

は

ひ

ふ

へ

め

ゆ

り

ろ

わ