



Sitecore CMS 7.0 以降 CMS パフォーマンス チューニング ガイド

Sitecore CMS のパフォーマンス最適化のためのデベロッパー用ガイド

目次

Chapter 1	イントロダクション	4
Chapter 2	チューニング手順 - 全般	5
2.1	SQL Server インデックスの断片化レベルのチェック	6
2.2	SQL Server メンテナンス プランのチェック	8
2.3	データベース テーブルのクリーンアップ	11
2.4	データベース クリーンアップ エージェントのチェック	13
2.5	ソフトウェアおよびサーバーの設定	15
Chapter 3	チューニング手順 - データベースのプロパティ	17
3.1	SQL Server 2008 (100) に設定された互換性レベル	18
3.2	自動終了プロパティを false に設定	20
3.3	自動圧縮プロパティを false に設定	22
3.4	復旧モデルを単純に設定	24
Chapter 4	チューニング手順 - Sitecore のキャッシュ	26
4.1	初期キャッシュ値の設定	28
4.2	Sitecore のキャッシュのチューニング	32
4.3	プリフェッチ キャッシュ設定のガイドライン	36
4.4	出力 (レンダリング) キャッシュ設定のガイドライン	38
4.5	キャッシュ サイズの制限の無効化	43
Chapter 5	チューニング手順 - IIS 設定	44
5.1	HTTP キープアライブの有効化	45
5.2	IIS の Expire Web コンテンツ ヘッダー	47
5.3	IIS の静的なコンテンツの圧縮の有効化	49
5.4	IIS の動的なコンテンツの圧縮の有効化 (オプション)	51
Chapter 6	チューニング手順 - Sitecore のクライアント最適化	53
6.1	実行時間の長いバリデーターのチェック	54
6.2	過剰なアイテム バージョンのチェック	56
6.3	ノードごとの過剰アイテムのチェック	59
6.4	その他のクライアント固有の最適化	61
Chapter 7	チューニング手順 - Sitecore の多様なコンテンツ デリバリー サーバーの最適化	62
7.1	WebDAV の無効化	63
7.2	パフォーマンス カウンターの無効化	65
7.3	メモリモニタの無効化	66
Chapter 8	チューニング手順 - Sitecore 検索	67
8.1	バケットのデバッグの無効化	68
8.2	追加情報	69

この文書に記載されている情報は、文書の公開日現在の Sitecore Corporation の見解であり、事前の予告なく変更されることがあります。この文書と記載されている内容は、現状のまま提供され、一切の保証を伴いません。また、Sitecore による提供または責務を表明するものではありません。Sitecore は、記載されている情報の正確性を保証するものではありません。Sitecore はこの文書に関し、明示または黙示を問わず、いかなる保証もしません。

この文書で他社製品に言及する記述が含まれる場合、それらはお客様の便宜上の理由により提供されています。このような記述はすべて、Sitecore による推奨またはサポートを意味するものではありません。Sitecore は、これらの記述の正確性を保証できず、また、製品はいずれ変更される可能性があります。またこれらの記述は、製品の包括的な説明を目的としておらず、製品を理解する上で重要なポイントの簡単な説明を目的としています。これらの製品の正式な記述については、製品のそれぞれのメーカーにお問い合わせください。

記載されているすべての商標は、それぞれ各社の所有物です。

©2014 Sitecore Corporation. All rights reserved.

Chapter 1

イントロダクション

本ガイドは、一連のチューニング タスクで、関係のあるセクションごとに分かれています。これは、Sitecore の実装が正常に最適のパフォーマンスで稼働していることを、確実にするのに役立ちます。

この Sitecore 診断ガイドは、関連問題の診断パフォーマンスの手助けとして有効な手引きガイドです。

このマニュアルには次の章があります。

- **Chapter 1 — イントロダクション**
- **Chapter 2 — チューニング手順 - 全般**
- **Chapter 3 — チューニング手順 - データベースのプロパティ**
- **Chapter 4 — チューニング手順 - Sitecore のキャッシュ**
- **Chapter 5 — チューニング手順 - IIS 設定**
- **Chapter 6 — チューニング手順 - Sitecore のクライアント最適化**
- **Chapter 7 — チューニング手順 - Sitecore の多様なコンテンツ デリバリー サーバーの最適化**
- **Chapter 8 — チューニング手順 - Sitecore 検索**

Chapter 2

チューニング手順 - 全般

チューニング手順 – 全般は、Sitecore の実装が最高のパフォーマンスで動作するように構成されているかどうかを確認するために設計された一連のタスクです。

それぞれのタスクには、入門情報、必要なスキル、起こりうる症状、設定を確認する手順、問題の解決方法、記録する結果についての情報が含まれます。

より具体的なチューニング手順は、次の章で説明します。

この章には次のセクションがあります。

- SQL Server インデックスの断片化レベルのチェック
- SQL Server メンテナンス プランのチェック
- データベース テーブルのクリーンアップ
- データベース クリーンアップ エージェントのチェック
- ソフトウェアおよびサーバーの設定

2.1 SQL Server インデックスの断片化レベルのチェック

インデックスが古くなるにつれ、連続しないデータの挿入と削除が負担となって、断片化を生じる原因となる可能性があります。頻繁に使用される CMS データベースでは、ほんの数日間でこの現象が起こることがあります。少量の断片化であれば、通常はパフォーマンスに悪影響を及ぼしません。しかし、断片化の割合が上がるにつれ、パフォーマンスは劇的に影響を受けます。

2.1.1 必要なスキル

- SQL Server 2008 Management Studio の実務知識。

2.1.2 症状

- CPU 使用の劇的な増加。
- クエリにおけるパフォーマンスの低下。
- データベース書き込みにおけるパフォーマンスの低下。
- データベース サーバーへの接続の中断。
- 時間のかかるレンダリング。
- 時間のかかる Sitecore クライアント ツール — デスクトップ、コンテンツ エディターなど。

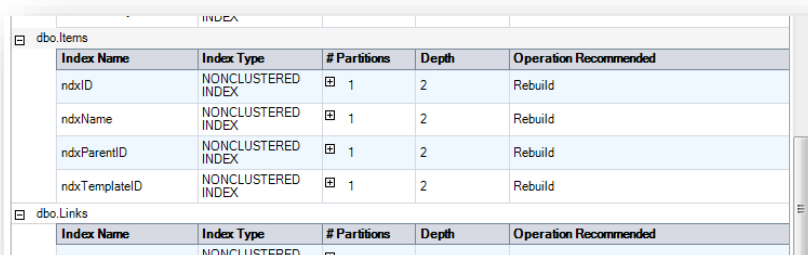
2.1.3 断片化されたインデックスのチェック手順

インデックス断片化の割合をチェックするために、CMS データベース (Core, Master, Web) に対して物理統計の標準レポートのインデックスを、次のように実行します:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ** の *Master* データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックし、**互換性レベル**が **[SQL Server 2008 (100)]** に設定されていることを確認します。
4. **[OK]** をクリックします。
5. **オブジェクト エクスプローラ**の *Master* データベースを右クリックし、**[レポート]**、**[標準レポート]**、**[インデックスの物理統計]** をクリックします。
6. SQL Server Management Studio は、テーブル名、インデックス名、インデックスの種類、パーティション数、および推奨されている操作**についての情報を表示する**レポートを生成します。
7. コアおよび *Web* データベースに関してはステップ 5 - 6 を繰り返し、インデックスに生じた断片化のレベルをチェックします。

2.1.4 結果の理解

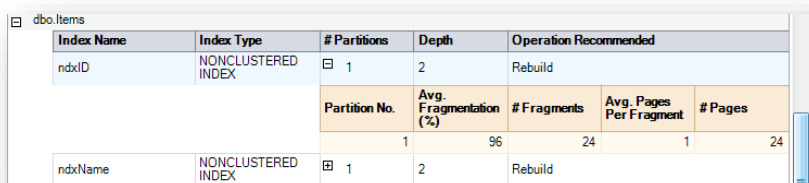
出力はこのように見えます:



Index Name	Index Type	# Partitions	Depth	Operation Recommended
ndxID	NONCLUSTERED INDEX	1	2	Rebuild
ndxName	NONCLUSTERED INDEX	1	2	Rebuild
ndxParentID	NONCLUSTERED INDEX	1	2	Rebuild
ndxTemplatelD	NONCLUSTERED INDEX	1	2	Rebuild

レポートに表示されるキー値の 1 つに、[推奨されている操作] フィールドがあります。いずれの再構築という値も、インデックスが断片化されていることを示します。

[# パーティション] フィールドを展開することで、所定のインデックスの断片化された割合 (%) を表示することができます。



Index Name	Index Type	# Partitions	Depth	Operation Recommended			
ndxID	NONCLUSTERED INDEX	1	2	Rebuild			
					Partition No.	Avg. Fragmentation (%)	# Fragments
			1	96	24	1	24
ndxName	NONCLUSTERED INDEX	1	2	Rebuild			

2.1.5 推奨事項

Sitecore は、インデックスの断片化を 10% 未満に保つことを推奨しています。

2.1.6 解決方法

CMS データベース (Core, Master, Web) のインデックスをデフラグするために、デフラグのメンテナンス プランを次のように実行します:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ**で、[管理/メンテナンス プラン] フォルダを展開します。
3. *[**インデックス デフラグのメンテナンス プラン**] を右クリックして、[実行] をクリックします。

このようなメンテナンス プランが存在しない場合は、「SQL Server メンテナンス プランのチェック」のセクションを参照してください。

2.2 SQL Server メンテナンス プランのチェック

メンテナンス プランによって、自動化されたタスク セットをスケジュールに従って実行することで、データベースのマニュアル メンテナンスの必要性が取り除かれます。このプランでは、データベースに対し定期的にチェックとメンテナンスを行い、データベースを最適の状態に保ちます。

2.2.1 必要なスキル

- SQL Server 2008 Management Studio の実務知識
- T-SQL スクリプトの実行の実務知識

2.2.2 症状

- データベースが最適のパフォーマンスまで達していない。
- インデックスが断片化されつつあり、対処が必要。

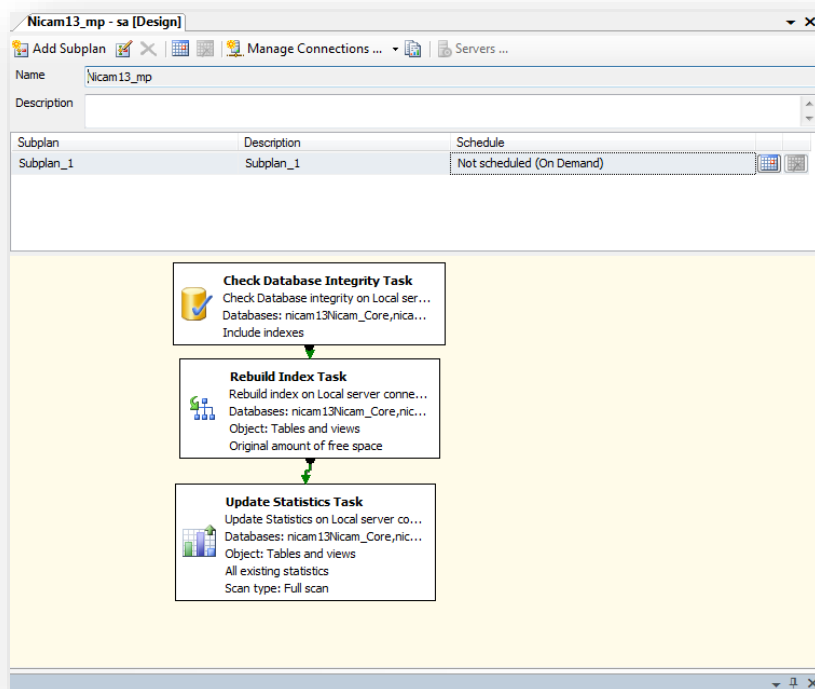
2.2.3 SQL Server メンテナンス プランのチェック手順

SQL Server メンテナンス プランが存在していて、かつベスト プラクティスに従っていることを確認するには、次を実行してください:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ**で、**[管理/メンテナンス プラン]** フォルダーを展開します。
3. メンテナンス プランが存在する場合、ダブルクリックしてどのように構成されているか確認します。このタスクの「調査結果」セクションで使用されます。

2.2.4 結果の理解

出力はこのように見えます:



メンテナンス プランには、以下のタスクが含まれます:

- データベースの整合性確認タスク
- インデックスの再構築タスク
- 統計の更新タスク

2.2.5 推奨事項

Sitecore は、CMS データベースに SQL Server メンテナンス プランを設定することを推奨しています。メンテナンス プランには、データベースの整合性確認タスク、インデックスの再構築タスク、および統計の更新タスクが含まれます。

2.2.6 解決方法

SQL Server Management Studio は、メンテナンス プランの作成を単純化するために IDE を提供します。インデックスをデフラグするために、MP を作成します:

1. SQL Server Management Studio を起動します。
2. **オブジェクト エクスプローラ**で、**[管理]** フォルダを展開します。

3. [メンテナンスプラン] フォルダーで右クリックし、[新しいメンテナンスプラン] を選択します。
4. MP に「Defragment CMS Indexes (CMS インデックスのデフラグ)」のような意味のある名前を付けます。
5. ツールボックスから [データベースの整合性確認タスク]、[インデックスの再構築タスク]、[統計の更新タスク] をドラッグ アンド ドロップし、縦に同じ順番で並べます。
6. ボックスから別のボックスへの矢印をドラッグすることでタスクをつなげ、次のように接続します: [データベースの整合性確認タスク] -> [インデックスの再構築タスク] -> [統計の更新タスク]
7. [データベースの整合性確認タスク] を右クリックして、[編集] を選択します。
8. [接続] および CMS データベース — コア、マスター、Web — を選択して、[OK] をクリックします。
9. [インデックスの再構築タスク] を右クリックして、[編集] を選択します。
10. [接続] および CMS データベース — コア、マスター、Web — を選択し、[インデックスの再作成中にオンラインのインデックスを保持する] チェック ボックスを選択して — SQL Server Enterprise Edition の場合のみ — [OK] をクリックしてください。
11. [統計の更新タスク] を右クリックし、[編集] を選択します。
12. [接続] および CMS データベース — コア、マスター、Web — を選択し、オブジェクトに [テーブルとビュー]、[すべての既存の統計]、および [スキャンの種類] として [フル スキャン] を設定し、[OK] をクリックします。
13. [スケジュール] アイコン (右上の隅) の横の [カレンダー] アイコンをクリックし、スケジュールを毎週実行するように設定します。
14. **保存**します。

2.2.7 メモとコメント

このタスクでは、Core、Master、Web の各データベースのインデックスを再構築する単一のメンテナンスプランの作成、および毎週メンテナンスプランを実行するスケジューリングの設定について説明します。いくつかのデータベースのインデックスのフラグメンテーションが他のデータベースより進んでいる場合、Core、Master、Web の各データベースでメンテナンスプランを分割します。これにより、インデックスのフラグメンテーションの進度に従い異なるスケジューリングを設定できます。

2.3 データベース テーブルのクリーンアップ

web.config ファイルには、*History*、*PublishQueue* および *EventQueue* テーブルからアーティファクトデータを削除する一連のクリーンアップ エージェントが設定されています。しかし、時にはこれらのテーブルが大きくなりすぎて、エージェントの実行中にタイムアウトが発生する場合があります。

このタスクはテーブル サイズを確認するように設計されており、1000 行以上のテーブルの場合は切り捨てられます。

2.3.1 必要なスキル

- SQL Management Studio の実務知識。
- SQL スクリプト実行の実務知識。

2.3.2 症状

- 公開時のパフォーマンス低下。
- インデックス作成時のパフォーマンス低下。

2.3.3 History、PublishQueue、EventQueue テーブルのクリーンアップ手順

History、*PublishQueue*、*EventQueue* テーブルのクリーンアップ方法:

1. SQL Server Management Studio を起動します。
2. **新規クエリ** をクリックします。
3. 次のクエリを実行します。実行する際、***/* database name */*** を、スクリプトを実行するデータベース名と置き換えます。

```
USE /* database name */

/* TRUNCATE History TABLE */
IF OBJECT_ID('History', 'U') IS NOT NULL
  IF ((SELECT COUNT(*) FROM [History]) > 1000)
  BEGIN
    TRUNCATE TABLE [History];
    PRINT 'Truncated the History Table';
  END

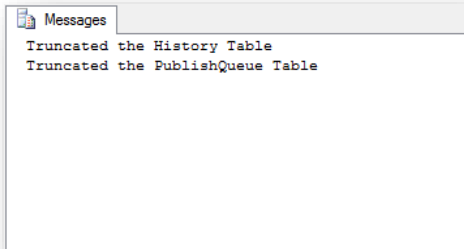
/* TRUNCATE EventQueue TABLE */
IF OBJECT_ID('EventQueue', 'U') IS NOT NULL
  if ((SELECT COUNT(*) FROM [EventQueue]) > 1000)
  BEGIN
    TRUNCATE TABLE [EventQueue];
    PRINT 'Truncated the EventQueue Table';
  END

/* TRUNCATE PublishQueue TABLE */
IF OBJECT_ID('PublishQueue', 'U') IS NOT NULL
  IF ((SELECT COUNT(*) FROM [PublishQueue]) > 1000)
  BEGIN
    TRUNCATE TABLE [PublishQueue];
```

```
PRINT 'Truncated the PublishQueue Table';  
END
```

2.3.4 結果の理解

出力結果は、次のようなものとなります:



- [メッセージ] ウィンドウに、切り捨てられたテーブル一覧が表示されます。

2.3.5 推奨事項

Sitecore は、History、PublishQueue、EventQueue テーブルの行数 (エン트리数) を 1000 より少なくすることを推奨しています。これにより、クリーンアップエージェント実行中のタイムアウトを防ぐことができます。

2.3.6 解決方法

前述の手順を実行します。

2.4 データベース クリーンアップ エージェントのチェック

`web.config` ファイルには、`History`、`PublishQueue` および `EventQueue` テーブルからアーティファクトデータを削除する一連のクリーンアップ エージェントが設定されています。

このタスクは、それらのタスクの設定および定期的なスケジューリングによる実行をチェックするよう設計されています。

2.4.1 必要なスキル

- Sitecore `web.config` ファイルの実務知識。

2.4.2 症状

- 公開時のパフォーマンス低下。
- インデックス作成時のパフォーマンス低下。

2.4.3 History、PublishQueue、EventQueue クリーンアップ タスクの設定チェック手順

クリーンアップ タスクが定期的に行われるよう設定されているかを確認するにはどこを参照すればよいかを示す手順は次のとおりです：

1. お使いのエディターで、`web.config` ファイルを開きます。
2. `<configuration><sitecore><scheduling></scheduling>` ノードに移動します。

2.4.4 結果の理解

出力結果は次のようなものとなります — 分かりやすくするために、`<agent>` ノードはいくつか削除しています：

```
<scheduling>
  <!-- Time between checking for scheduled tasks waiting to execute -->
  <frequency>00:05:00</frequency>

  <!-- Agent to clean up history data -->
  <agent type="Sitecore.Tasks.CleanupHistory" method="Run" interval="04:00:00"/>
  <!-- Agent to clean up publishing queue -->
  <agent type="Sitecore.Tasks.CleanupPublishQueue, Sitecore.Kernel" method="Run"
interval="04:00:00">
    <DaysToKeep>30</DaysToKeep>
  </agent>
  <!-- Agent to clean up the event queue -->
  <agent type="Sitecore.Tasks.CleanupEventQueue, Sitecore.Kernel" method="Run"
interval="04:00:00">
    <DaysToKeep>1</DaysToKeep>
  </agent>
```

- エージェントの実行準備が整っているかの確認のために、`<scheduling><frequency>` に `00:00:00` 以外の値を設定します。

- *Sitecore.Tasks.CleanupHistory* の間隔が **00:00:00** より大きい値になっていることを確認します。
- *Sitecore.Tasks.CleanupPublishQueue* の間隔が **00:00:00** より大きい値になっていることを確認します。
- *Sitecore.Tasks.CleanupEventQueue* の間隔が **00:00:00** より大きい値になっていることを確認します。

2.4.5 推奨事項

Sitecore は、値を **00:00:00** より大きい値に設定することでスケジューリングの頻度を有効にし、*History*、*PublishQueue* および *EvenQueue* のクリーンアップエージェントの間隔に **00:00:00** より大きな値を設定することを推奨します。

頻度のデフォルト値は **00:05:00** であり、間隔のデフォルト値は **04:00:00** です。

2.4.6 解決方法

スケジューリング頻度、*Sitecore.Tasks.CleanupHistory* の間隔、*Sitecore.Tasks.CleanupPublishQueue* の間隔、および/または *Sitecore.Tasks.CleanupEventQueue* が **00:00:00** に設定されている場合、実行するには値を大きくする必要があります。未変更の *web.config* ファイルで指定しているデフォルト値は次のとおりです:

- スケジューリングの頻度 = **00:05:00** (5 分)
- *Sitecore.Tasks.CleanupHistory* 間隔 = **04:00:00** (4 時間)
- *Sitecore.Tasks.CleanupPublishQueue* 間隔 = **04:00:00** (4 時間)
- *Sitecore.Tasks.CleanupEventQueue* 間隔 = **04:00:00** (4 時間)

2.5 ソフトウェアおよびサーバーの設定

このチェックは、ソフトウェア (OS、IIS および SQL Server) およびサーバー設定が、パフォーマンスを最適化するための推奨プラクティスに従っているかどうかを確認します。サーバーのパフォーマンスレベルの記録には、元となるベースラインから開始する加点システムを使用します。スコアリングはセクションに分かれており、各セクションには独自の値が割り当てられています。どのセクションでも、得点 1 は Sitecore が推奨する最低限の要件を満たしていることを示しています。得点が 1 より大きい場合、より高いレベルで要件を満たしていることを示しています。得点が 1 より小さい場合、赤信号となり、サーバーのパフォーマンスが少なくとも Sitecore の推奨要件を満たすにはリソースの追加または変更が必要であることを示しています。

スコアリング システムは、SDN に示した (Sitecore バージョン 6.4.x をベースとした) インストール ガイドの最小要件に基づいています。

メモ – コンテンツ デリバリー ネットワーク (CDN) の使用

負荷の高い Web サイトを使用する場合、CDN を使ってある種のアセットを提供し、パフォーマンスを向上することは業務的な試みです。Sitecore では、画像ファイルやドキュメント、CSS ファイル、JavaScript ファイルなどのアセットに CDN を使用することを推奨しています。これらのアセットは適度に静的であり、Sitecore セキュリティーを必要としません。これによって最大限のサーバー リソースによる動的で安全なコンテンツを提供することが可能になります。

CDN から提供されるアセットのために、分析、パーソナライゼーション、エンゲージメント オートメーションなどの CEP 機能への深い考慮が必要です。これは、CEP 機能が Sitecore 環境の外部からのコンテンツの提供によって悪影響を受ける可能性があるためです。

CDN の使用、設定、構築についてはこのドキュメントでは取り扱いません。

この内容に関する有用な情報については、以下のブログのエントリを参照してください。

<http://blog.image0.com/sitecore/using-sitecore-publishing-pipeline-to-refresh-external-cdn-cache/>

<http://www.cognifide.com/blogs/sitecore/the-ultimate-approach-to-storing-sitecore-media-items-in-cdn/>

2.5.1 必要なスキル

- システム インフラストラクチャの実務知識。

2.5.2 症状

- 低いパフォーマンス。
- ラフィック増加に伴うパフォーマンス低下。

2.5.3 推奨事項

Sitecore の推奨するサーバー要件:

- すべてのサーバーは 64 ビット モードで稼働していなければなりません。
- すべてのサーバーは Windows Server 2008 R2 で、最新のセキュリティパッチを適用した状態で稼働していなければなりません。

Sitecore の推奨するコンテンツ オーサリング環境 (Web サーバー):

- IIS 7.x
- Quad Core プロセッサ
- 8 GB のメモリー
- 250 GB

Sitecore の推奨するコンテンツ オーサリング環境 (SQL Server):

- SQL Server 2008 R2 x64
- Dual Quad Core プロセッサ
- 12 GB のメモリー
- データ ファイル用の 250 GB のディスク
- ログ ファイル用の 250 GB のディスク

Sitecore の推奨する以下のコンテンツ デリバリー環境 (Web Server):

- IIS 7.x
- Dual Quad Core プロセッサ
- 8 GB のメモリー
- 250 GB のディスク

Sitecore の推奨する以下のコンテンツ デリバリー環境 (SQL Server):

- SQL Server 2008 R2 x64
- Dual Quad Core プロセッサ
- 12 GB のメモリー
- データ ファイル用の 250 GB のディスク
- ログ ファイル用の 250 GB のディスク

Chapter 3

チューニング手順 - データベースのプロパティ

チューニング手順 - データベースのプロパティには、SQL Server のデータベース プロパティの設定をチェックするために設計された一連のタスクが含まれています。これらのプロパティを適切に設定することにより、Sitecore を最善のパフォーマンスで実行できます。

この章には次のセクションがあります。

- SQL Server 2008 (100) に設定された互換性レベル
- 自動終了プロパティを false に設定
- 自動圧縮プロパティを false に設定
- 復旧モデルを単純に設定

3.1 SQL Server 2008 (100) に設定された互換性レベル

互換性レベルは SQL 構文およびクエリ解析に影響を与えます。また、パフォーマンスへ影響を与えてはなりません。互換性レベルを SQL Server 2008 (100) に設定することにより、CMS パフォーマンス チューニング ガイドで使用されている多くのスクリプト/コマンドで使用されている、新しい T-SQL 機能を活用できます。

3.1.1 必要なスキル

- SQL Management Studio の実務知識。

3.1.2 症状

- パフォーマンスを向上するためのスクリプトが実行できません。

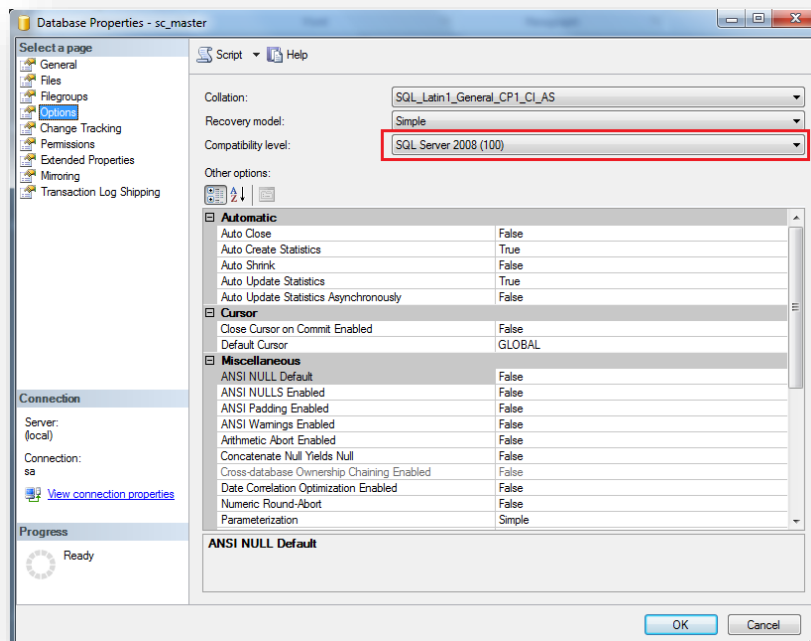
3.1.3 データベースの互換性レベルのチェック手順

データベースの互換性レベルの値をチェックする方法:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ**で、*CMS Master* データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックし、互換性レベルのプロパティを参照します。
4. **[OK]** をクリックします。
5. *CMS Core* および *Web* データベースでも、手順 1 から 4 を繰り返します。

3.1.4 結果の理解

出力結果は、次のようなものとなります:



3.1.5 推奨事項

Sitecore は、**互換性レベル** プロパティの設定として **SQL Server 2008 (100)** を推奨しています。

3.1.6 解決方法

互換性レベル プロパティの設定方法は次のとおりです:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ** で、**CMS Master** データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックして、**互換性レベル** が **SQL Server 2008 (100)** に設定されていることを確認します。
4. **[OK]** をクリックします。
5. **CMS Core** および **Web** データベースでも、手順 1 から 4 を繰り返します。

3.2 自動終了プロパティを false に設定

SQL Server がデータベースを開くと、その状態を維持するためのリソースが割り当てられます。ロック、バッファ、セキュリティトークンなどのためのメモリーがすべて割り当てられます。これらの操作には時間がかかります。自動終了プロパティは、これらのリソースをどのように処理するかを設定します。「true」または「ON」に設定されている場合、最後の接続が切断されるとリソースは解放されます。これは適切な処理のように思えますが、その後短時間で新しい接続が確立される場合 (1/10 秒以下)、これらすべてのリソースを再度割り当てる必要があります。自動終了プロパティを *false* または *OFF* に設定すると、この状態を回避できます。

3.2.1 必要なスキル

- SQL Server 2008 Management Studio の実務知識。

3.2.2 症状

- データベースへの接続に時間がかかります。

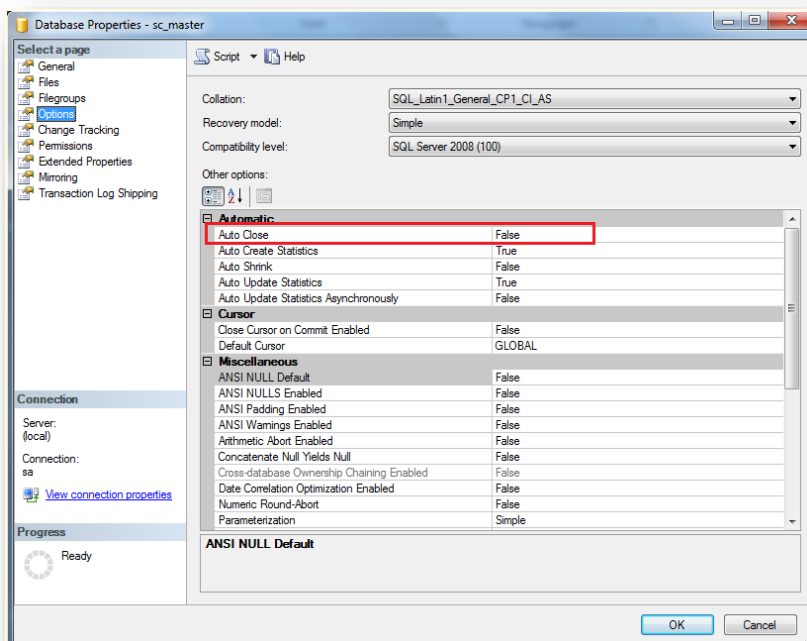
3.2.3 自動終了プロパティ値のチェック手順

自動終了プロパティ値のチェック方法:

1. SQL Server Management Studio を起動します。
2. オブジェクト エクスプローラで、CMS Master データベースを右クリックし、[プロパティ] をクリックします。
3. [オプション] ページをクリックし、自動終了プロパティを参照します。
4. [OK] をクリックします。
5. CMS Core および Web データベースでも、手順 1 から 4 を繰り返します。

3.2.4 結果の理解

出力結果は、次のようなものとなります:



3.2.5 推奨事項

Sitecore は、**自動終了プロパティ**を **false** に設定することを推奨しています。

3.2.6 解決方法

自動終了プロパティの設定方法は次のとおりです:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ**で、**CMS Master** データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックし、自動終了プロパティが **false** に設定されていることを確認します。
4. **[OK]** をクリックします。
5. **CMS Core** および **Web** データベースでも、手順 1 から 4 を繰り返します

3.3 自動圧縮プロパティを false に設定

自動圧縮プロパティには、a) コール時に大量のリソースを消費する、b) いつコールされるか制御できない、というデメリットがあります。自動圧縮と自動拡張を組み合わせると、データベースが継続的に圧縮と拡張を繰り返すというスパイラルに陥り、他のデータベースの重要なリソースを使ってしまっただけでなく、フラグメンテーションの問題も発生します。データベースまたはファイルで SHRINK コマンドを使用しなければならない場合、スクリプト、コマンドまたはスケジューリングされたメンテナンス プランから使用すべきです。自動圧縮プロパティを *false* または *OFF* に設定すると、この機能を無効にできます。

3.3.1 必要なスキル

- SQL Server 2008 Management Studio の実務知識。

3.3.2 症状

- パフォーマンスの低下。

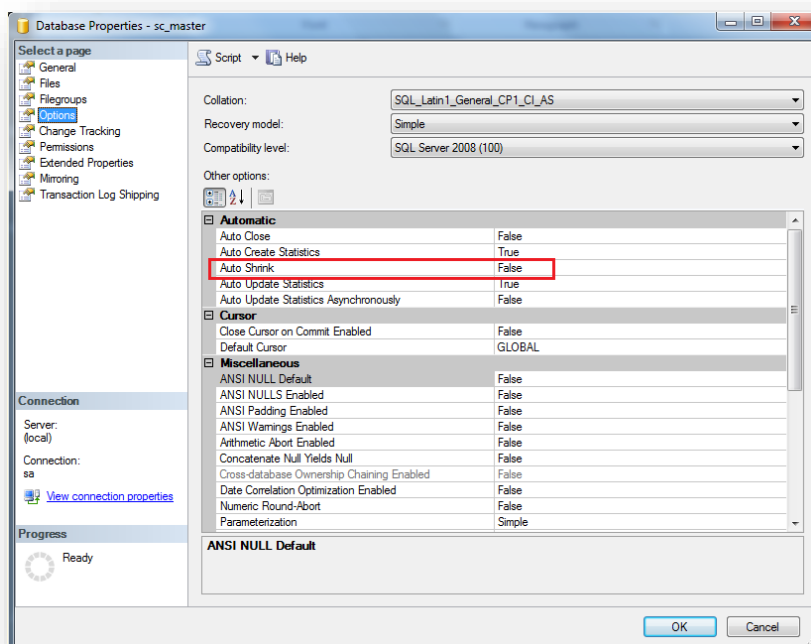
3.3.3 自動圧縮プロパティ値のチェック手順

自動圧縮プロパティ値のチェック方法:

1. SQL Server Management Studio を起動します。
2. **オブジェクト エクスプローラ**で、*CMS Master* データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックし、自動圧縮プロパティを参照します。
4. **[OK]** をクリックします。
5. *CMS Core* および *Web* データベースでも、手順 1 から 4 を繰り返します。

3.3.4 結果の理解

出力結果は、次のようなものとなります:



3.3.5 推奨事項

Sitecore は、**自動圧縮**プロパティを **false** に設定することを推奨しています。

3.3.6 解決方法

自動圧縮プロパティの設定方法は次のとおりです:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ**で、**CMS Master** データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックし、**自動圧縮**プロパティが **false** に設定されていることを確認します。
4. **[OK]** をクリックします。
5. **CMS Core** および **Web** データベースでも、手順 1 から 4 を繰り返します。

3.4 復旧モデルを単純に設定

SQL Server の復旧モデルが単純に設定されている場合、トランザクション ログには最小限の情報を記録します。トランザクション ログが 70% 使用された場合、またはトランザクション ログのアクティブな箇所の復旧に、サーバーレベル設定の復旧間隔で指定した以上の時間がかかる場合、SQL Server はトランザクション ログを切り捨てます。

復旧モデルを単純に設定すると、完全または一括ログと比べて最小限の負荷で済みます。このことは、CMS データベースのパフォーマンス要件にとって非常に重要です。

3.4.1 必要なスキル

- SQL Server 2008 Management Studio の実務知識。

3.4.2 症状

- 復旧間隔時のパフォーマンスの低下。

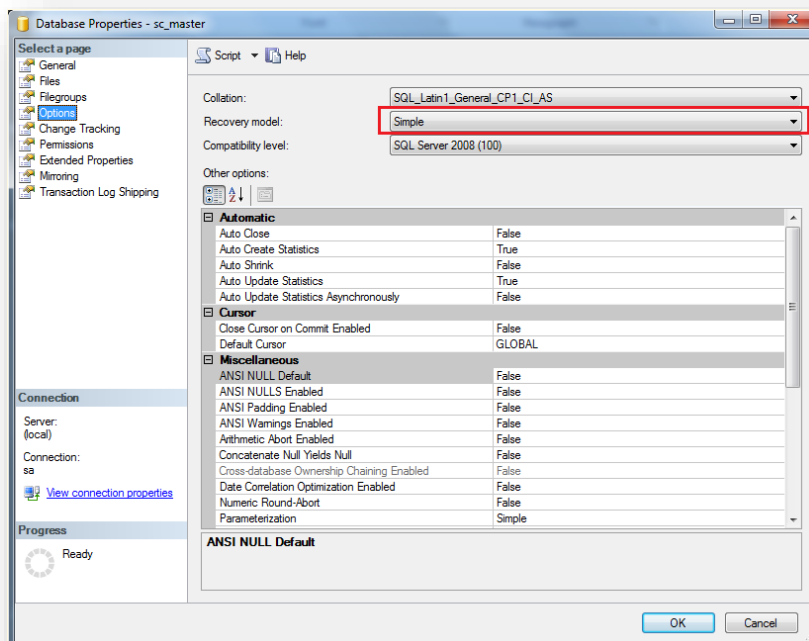
3.4.3 復旧モデル プロパティ値のチェック手順

復旧モデル プロパティ値のチェック方法:

1. SQL Server Management Studio を起動します。
2. オブジェクト エクスプローラで、CMS Master データベースを右クリックし、[プロパティ] をクリックします。
3. [オプション] ページをクリックし、復旧モデル プロパティを参照します。
4. [OK] をクリックします。
5. CMS Core および Web データベースでも、手順 1 から 4 を繰り返します。

3.4.4 結果の理解

出力結果は、次のようなものとなります:



3.4.5 推奨事項

Sitecore は、**復旧モデル** プロパティを**単純**に設定することを推奨しています。

3.4.6 解決方法

復旧モデル プロパティの設定方法は次のとおりです:

1. **SQL Server Management Studio** を起動します。
2. **オブジェクト エクスプローラ**で、**CMS Master** データベースを右クリックし、**[プロパティ]** をクリックします。
3. **[オプション]** ページをクリックし、**復旧モデル** プロパティが**単純**に設定されていることを確認します。
4. **[OK]** をクリックします。
5. **CMS Core** および **Web** データベースでも、手順 1 から 4 を繰り返します。

Chapter 4

チューニング手順 - Sitecore のキャッシュ

Sitecore キャッシュの調査および設定は、複数のタスクに分割できます。これにより、各タスクを目的別に明確に分け、単純化できます。目的は、Sitecore データベース キャッシュ (プリフェッチ、データおよびアイテム キャッシュ) を設定、チューニングすることです。

出力レンダリング キャッシング プロパティの設定について、ユーザーはこれらのキャッシュを適切に有効にする方法およびキャッシュを期限切れにする方法の確認のために、Sitecore キャッシュ設定リファレンスおよび Sitecore プレゼンテーション コンポーネントリファレンスの両方を参照する必要があります。

Sitecore キャッシュの設定およびチューニングは、テスト環境または開発環境でのみ実施し、本番環境で実行すべきではありません。キャッシュを変更すると、保持されていたキャッシュが消去されてしまうためです。キャッシュが消去されると、本番環境でのユーザー エクスペリエンスが低下する可能性があります。

キャッシュのチューニングが完了後は、ピーク時を避けて、設定ファイルへの変更を本番環境に適用できます。

キャッシュ サイズ制限の無効化が可能な 64 ビット環境で、Sitecore 6.4.1 アップデート 4 (rev. 110928) 以降を使用している場合は、キャッシュ サイズの制限の無効化を参照してください。

この章には次のセクションがあります。

- 初期キャッシュ値の設定
- Sitecore のキャッシュのチューニング
- プリフェッチ キャッシュ設定のガイドライン
- 出力 (レンダリング) キャッシュ設定のガイドライン

- キャッシュ サイズの制限の無効化

4.1 初期キャッシュ値の設定

使用する環境により、このタスクでは Sitecore データベースおよび html 出力キャッシュの初期値を設定できます。これらの値は、キャッシュのチューニング フェーズで使用されます。

注意

キャッシュ値の変更は、テスト環境または開発環境でのみ行うべきです。キャッシュのチューニングが完了後は、キャッシュへの変更を適用すると、キャッシュの内容が消去されるということを理解したうえで、新しいキャッシュ値を本番環境に適用できます。

4.1.1 必要なスキル

- Sitecore 設定ファイルの実務知識。

4.1.2 症状

- キャッシュに割り当てられるメモリの制限
- メモリ制限によるキャッシュからのデータの頻繁な消去
- 低いページレンダリング パフォーマンス

4.1.3 初期値

次の表は、データベース キャッシュおよび html 出力キャッシュの推奨初期値を示しています。これらの値は、a) 環境と、b) 64 ビット モードで稼働していることに基づいています。32 ビット モードで稼働中のシステムについては、この半分の値を設定します。利用可能なメモリの制限により、32 ビット モードを使用しないことを強く推奨します。

環境/ターゲット	キャッシュ	値
コンテンツ配信のみ		
Web	プリフェッチ	200 MB
Web	データ	200 MB
Web	アイテム	200 MB
出力 (サイトごと)	html	100 MB
CMS Only		
Master	プリフェッチ	200 MB

環境/ターゲット	キャッシュ	値
Master	Data	200MB
Master	Item	200 MB
CMS およびコンテンツ配信が同じサーバーにある (Master および Web データベース)		
Master	プリフェッチ	200 MB
Master	データ	200 MB
Master	アイテム	200 MB
Web	プリフェッチ	150 MB
Web	データ	150 MB
Web	アイテム	150 MB
出力 (サイトごと)	html	100 MB
CMS およびコンテンツ配信が同じサーバーにある (ライブ モード)		
Master	プリフェッチ	300 MB
Master	データ	300 MB
Master	アイテム	300 MB
出力 (サイトごと)	html	100 MB

4.1.4 データ キャッシュおよびアイテム キャッシュを初期値に設定する手順

データ キャッシュおよびアイテム キャッシュの初期値への設定方法:

1. web.config ファイルをエディターで開きます — web の root ディレクトリにあります。
2. <configuration><sitecore><databases><database id="x"><cacheSizes> に移動します — 「x」は、上記の初期値表に示されたデータベースです。

```
<cacheSizes hint="setting">
  <data>20MB</data>
  <items>10MB</items>
  <paths>500KB</paths>
  <standardValues>500KB</standardValues>
</cacheSizes>
```

3. データ キャッシュおよびアイテム キャッシュを初期値に設定します。
4. 実行環境に基づき、各データベースでこの手順を繰り返します。

4.1.5 プリフェッチ キャッシュを初期値に設定する手順

プリフェッチ キャッシュの初期値への設定方法:

1. プリフェッチ キャッシュの設定は、App_Config\Prefetch ディレクトリにあります。各データベース — Core, Master, and Web — にはそれぞれ別個の設定ファイルがあります。上記の初期値表に基づき、作業しているデータベースのファイルを適切に編集します。

2. プリフェッチ設定ファイルを開きます。
3. <configuration><cacheSize> に移動します。

```
<configuration>
  <cacheSize>20MB</cacheSize>
```

4. プリフェッチ キャッシュの初期値を設定します。
5. 実行環境に基づき、各データベースでこの手順を繰り返します。

4.1.6 html 出力キャッシュを初期値に設定する手順

データ キャッシュおよびアイテム キャッシュの初期値への設定方法:

1. web.config ファイルをエディターで開きます — web の root ディレクトリにあります。
2. <configuration><sitecore><sites><site name="website"> に移動します。

```
<site name="website" virtualFolder="/" physicalFolder="/" rootPath="/sitecore/content"
startItem="/home" database="web" domain="extranet" allowDebug="true" cacheHtml="true"
htmlCacheSize="100MB" registryCacheSize="0" ViewStateCacheSize="0"
xslCacheSize="5MB" filteredItemsCacheSize="2MB" enablePreview="true" enableWebEdit="true"
enableDebugger="true" disableClientData="false"/>
```

3. html 出力キャッシュの初期値を設定します。
4. 実行環境に基づき、各ウェブサイトでこの手順を繰り返します。

4.1.7 推奨事項

Sitecore は、プリフェッチ、データ、アイテムおよび html 出力の各キャッシュを、パフォーマンス キャッシュのチューニングを行う前に初期値に設定することを推奨します。

4.1.8 解決方法

プリフェッチ、データ、アイテムおよび html 出力キャッシュを、お使いの環境に従った初期値に設定するには、上記の手順に従ってください。

4.2 Sitecore のキャッシュのチューニング

このタスクでは、負荷生成ツールを使用し、さまざまなキャッシュのサイズおよび消去の監視に Sitecore の `cache.aspx` ページを使用しながら、Sitecore のプリフェッチ、データ、アイテムの各キャッシュをチューニングする手順について説明しています。出力 (html) およびレンダリング キャッシュの詳細は、Sitecore キャッシュ設定リファレンスおよび Sitecore プレゼンテーション コンポーネントリファレンスマニュアルを参照してください。

キャッシュチューニングは、時々実行する必要がある、継続的なプロセスです。コンテンツ/アイテムが追加されてウェブサイトが大きくなると、サイトを最善なパフォーマンスレベルで運営するためにキャッシュのサイズを変更しなければならない場合があります。

4.2.1 必要なスキル

- `cache.aspx` ページの実務知識。
- Sitecore 設定ファイルの実務知識。
- WCAT または Web Performance Load Tester のような負荷生成ツールの実務知識。

4.2.2 症状

- 低いソリューション パフォーマンス。
- 高負荷のサーバー。
- システム容量の低下。

4.2.3 手順

Sitecore データベース キャッシュのチューニングは、負荷を生成してテスト サイトまたは開発サイトに負荷をかけ、キャッシュのサイズを最大サイズ (または初期値) と比較し、消去率を確認し、キャッシュ値を変更して再確認する、という手順を繰り返し実行することによって行われます。

4.2.4 必要条件

Sitecore データベース キャッシュは、チューニング前に初期値に設定する必要があります。初期キャッシュ値の設定の項を参照してください。

キャッシュの確認には、負荷生成ツールとスクリプトが必要です。Web 負荷を生成するプログラムの詳細は、WCAT を使用した負荷生成または Web Performance Load Tester の使用を参照してください。

(Sitecore の推奨) OS システム タイプは 64 ビット モードで稼働しています。

4.2.5 Sitecore データベース キャッシュのチューニング手順

- すべての言語のすべてのアイテムがヒットするように、負荷を生成します。負荷生成の実行時間は、プロセス中の開始、停止が不要なように、チューニング手順全体を通じて実行されるようにしなければなりません。
- 負荷生成が実行中に、/sitecore/admin/cache.aspx ページに移動します。環境に応じて、該当するキャッシュは Master (データ)、Master (アイテム)、Web (データ)、Web (アイテム)、SqlDataProvider - プリフェッチ データ (Master)、SqlDataProvider - プリフェッチ データ (Web) となります。

Name	Count	Size	Delta	MaxSize
master[data]	6732	20MB	20MB	50MB
master[items]	9816	47MB	47MB	75MB
web[data]	0	0	0	20MB
web[items]	0	0	0	10MB
SqlDataProvider - Prefetch data(master)	7302	97MB	97MB	100MB

- キャッシュをチューニングするには、(ブラウザの [再読み込み] ボタンではなく) ページの [更新] ボタンをクリックする必要があります。これにより、負荷に対してキャッシュがどのように動作するかを確認できます。最大サイズに比べて、キャッシュのサイズがどのようなサイズになっているか、デルタ値に伴って発生する変動がどうなっているかを確認する必要があります。次に、これに従ったキャッシュサイズ調整のガイドラインを示します (キャッシュに割り当てることができるメモリの最大サイズはシステムで利用可能なメモリの空き容量に応じて異なります。メモリの空き容量が多ければ多いほど、割り当てることができるサイズも大きくなります [ハードウェアの推奨タスクを参照してください]):
- キャッシュ変動のデルタ値が一定である場合、またはキャッシュのサイズが常に最大サイズの 80% より大きい場合、キャッシュを 50% 増やします。
- キャッシュのサイズが最大サイズの 50% 未満を保っている場合、メモリー消費を抑えるためにキャッシュサイズを 25% 減らします。
- キャッシュが安定するまで、手順 3 を繰り返します。デルタの変動が一定で、キャッシュ サイズが 70% から 80% の間を保っているのが理想的です。

4.2.6 結果の理解

[cache.aspx ページのスクリーン ショットを挿入する]

4.2.7 推奨事項

Sitecore は、デルタの変動が一定で、キャッシュ サイズが 70% から 80% の間を保っている状態を推奨しています。

4.2.8 解決方法

上記の Sitecore データベース キャッシュのチューニング手順に従ってください。

調査結果レポート

4.3 プリフェッチ キャッシュ設定のガイドライン

プリフェッチ キャッシュ設定のガイドラインはタスクではなく、プリフェッチ キャッシュについての入門、および設定、活用方法の推奨プラクティスの説明となります。

Sitecore キャッシングについて十分理解するには、Sitecore キャッシュ設定リファレンスを通読することを強く推奨します。次の項を参照してください。

http://sdn.sitecore.net/upload/sitecore6/sc62keywords/cache_configuration_reference_us.pdf

4.3.1 プリフェッチ キャッシュの理解

プリフェッチ キャッシュは、プリフェッチ設定ファイルが提供する情報に基づいてアプリケーションの初期化時に生成されます。これにより、アプリケーションの再起動後のユーザー エクスペリエンスがより円滑になります。ただし、プリフェッチ キャッシュの過度な利用は、アプリケーションの再起動に要する時間の増加につながり、ユーザー エクスペリエンスへのデメリットとなります。

アプリケーション起動後のプリフェッチ キャッシュの活用について理解するには、Sitecore キャッシングがどのように動作しているかを理解する必要があります：

データベース アイテム キャッシュにアイテムのエントリが含まれていない場合、Sitecore はデータベース データ キャッシュから該当エントリを取得し、別のタイプへと変換し、変換されたデータをエントリとしてデータベース アイテム キャッシュに保存します。データベース データ キャッシュにアイテムのエントリが含まれていない場合、Sitecore はデータベースのプリフェッチ キャッシュから該当エントリを取得し、別のタイプへと変換し、変換されたデータをエントリとしてデータベース データ キャッシュに保存します。エントリがデータベースのプリフェッチ キャッシュに含まれていない場合、Sitecore はデータベースのデータ プロバイダーからデータを取得し、別のタイプへと変換し、変換されたデータをエントリとしてデータベースのプリフェッチ キャッシュに保存します。

つまり、プリフェッチ キャッシュは初期化時に生成されるだけでなく、アプリケーション実行時にも生成されます。プリフェッチ キャッシュのサイズを考慮する際、この流れを理解する必要があります。

4.3.2 プリフェッチ キャッシュ設定ファイルの理解

プリフェッチ キャッシュ設定ファイルは、`/App_Config/Prefetch` ディレクトリにあります。データベースごとに別個のプリフェッチ キャッシュ設定ファイルがあり (`Core.config`, `Master.config`, `Web.config`)、`/App_Config/Prefetch/Common.config` ファイルと統合されています。

アイテム、テンプレートなどは、GUID によって設定ファイルで識別されています。

設定要素

- `<configuration>` - 設定ファイルのルート ノードです。
- `<cacheSize>` - プリフェッチ キャッシュの最大値を制御します。次はその例です：
`<cacheSize>100MB</cacheSize>`.

- `<item>` - この要素は、アプリケーションの初期化時に、プリフェッチ キャッシュに読み込み特定のアイテムを指定します。次はその例です: `<item desc="home">{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}</item>`.
- `<template>` - この要素は、アプリケーションの初期化時に、指定されたテンプレートに基づきすべてのアイテムをキャッシュに読み込むよう指定します。次はその例です: `<template desc="layout">{3A45A723-64EE-4919-9D41-02FD40FD1466}</template>`.
- `<children>` - この要素は、指定されたアイテムのすべての子を読み込むように指定します。(次はその例です: `<children desc="main items">{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}</children>`).
- `<childLimit>` - キャッシュに読み込む子の数を指定します。アイテムの子の数がこの値より多い場合、Sitecore は子をキャッシュしません。次はその例です: `<childLimit>100</childLimit>`.

4.3.3 プリフェッチ キャッシュの推奨プラクティス

Sitecore のプリフェッチ キャッシュの設定時のガイダンスとなる推奨プラクティスの一覧は次のとおりです:

- Sitecore のプリフェッチ キャッシングを活用するには、まず設定する必要があります。デフォルトのプリフェッチ キャッシング ファイルの情報は、多くの場合正確ではありません。たとえば、`/App_Config/Prefetch/Master.config` ファイルには html 関連コントロールの複数のエントリが含まれます。これらのコントロールは現在では Core データベースに含まれています。
- プリフェッチ キャッシュのサイズ設定時には、プリフェッチ キャッシュはアプリケーションの初期化時だけでなく、アプリケーションが実行されている間ずっと生成される場合があることを考慮する必要があります。
- アイテム キャッシュおよびデータ キャッシュと同様に、プリフェッチ キャッシュの拡大、サイズ、消去についても監視する必要があります。詳細は、上記の Sitecore キャッシュのチューニングを参照してください。
- Sitecore は、`<setting name="ContentEditor.RenderCollapsedSections" value="false" />` を `/web.config` ファイルに追加することを推奨しています。これは、デフォルトでは「true」に設定されている隠し設定です。この設定が「true」(デフォルト) のままの場合、コンテンツ エディターの実装により、Sitecore は壊れたアイテムの子をメモリー (キャッシュ) に読み込み続け、プリフェッチによりその子も読み込まれます。これにより、Sitecore は 2 つのレベルを余分に読み込むこととなります。たとえば、ホーム アイテムを表示すると、動作が遅くなります。

4.4 出力 (レンダリング) キャッシュ設定のガイドライン

html 出力 (レンダリング) キャッシュのガイドラインの設定はタスクではなく、Sitecore 出力キャッシュについての入門、および設定、活用方法の推奨プラクティスの説明となります。

Sitecore 出力キャッシングについて十分に理解するには、*Sitecore* プレゼント コンポーネントリファレンスの第 4 章、出力キャッシングを読むことを推奨します。次の項を参照してください。

<http://sdn.sitecore.net/Reference/Sitecore%206/Presentation%20Component%20Reference.aspx>

4.4.1 出力 (レンダリング) キャッシュの理解

Sitecore 出力キャッシングを活用することにより、ウェブサイトのパフォーマンスを大幅に改善できます。レンダリング操作の結果を取得し、それらの結果をメモリーから表示すると、レンダリング コードを実行するよりもずっと早く処理できます。

Sitecore 出力キャッシングは、管理されている各ウェブサイトに関連付けられています。

デベロッパーは Sitecore 出力キャッシングを使用し、キャッシュするレンダリング コンポーネントをピックアップ、選択できます。あわせて、そのキャッシュの消去タイミングの規則を定義する VaryBy パラメーターを使用できます。これにより、同一の結果ウェブページ内のヘッダーやフッターなどの一般的なページ コンポーネントにキャッシュし、ニュースフィードなどの動的コンポーネントは動的に設定できます。

レイアウト エンジンのデフォルトの動作は、キャッシュプレゼンテーション コンポーネントではありません。

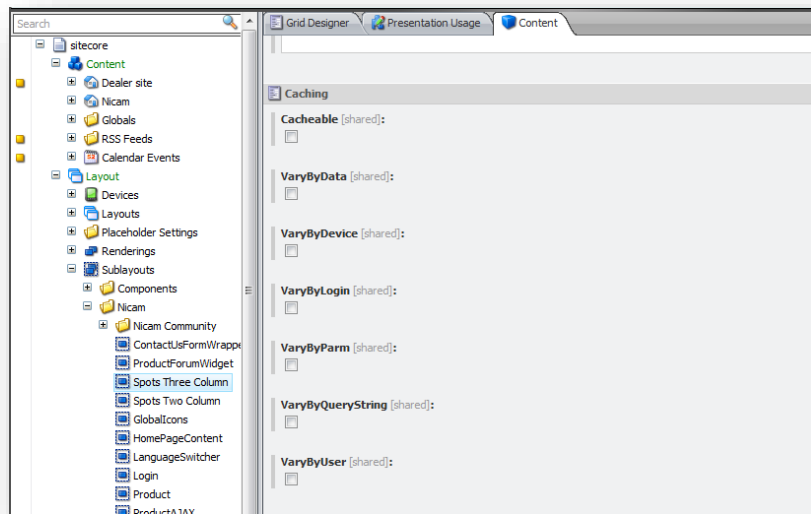
Sitecore 出力キャッシングを活用するには、プレゼンテーション コンポーネントを適切に設定する必要があります。この項では、設定方法のガイダンスと推奨プラクティスについて説明します。

Sitecore 出力キャッシングは、ASP.NET ページとフラグメントキャッシング (Web フォームと Web ユーザー コントロールにおいて、OutputCache ディレクティブと共に実装) と混同してはなりません。デベロッパーは、公開などの操作後に ASP.NET キャッシュを適切に消去する方法を理解せずに、Sitecore 出力キャッシングと ASP.NET ページとフラグメントキャッシングをあわせて使用してはなりません。

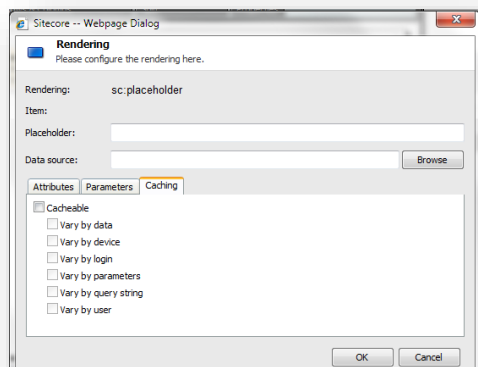
4.4.2 キャッシュ設定の設定場所および適用されるキャッシュ設定

デベロッパーは、3つの場所で Sitecore の出力キャッシュを設定できます:

- サブレイアウトおよびレンダリング定義アイテムの **[キャッシング]** セクション。



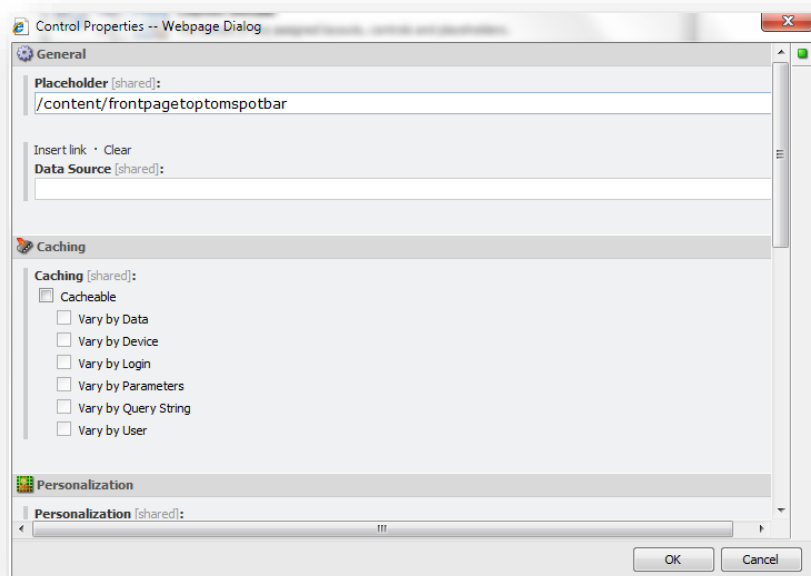
- プレゼンテーション コンポーネントをレイアウトまたはサブレイアウトに静的にバインドする際の、そのプロパティ。



- 宣言コード:

```
<sc:sublayout runat="server" renderingid="D366A65-54FF-49B5-A57F-2EBB9F426433%7d" placeholder="content" cacheable="true" varybydata="true" path="/layouts/Starter Kit/Sublayouts/Header Fixed.ascx" id="HeaderFixed"></sc:sublayout>
```

- プレゼンテーション コンポーネントを、レイアウト詳細のプレースホルダーにバインドする際の [コントロール プロパティ] ダイアログ ボックスの [キャッシング] セクション。



レイアウト エンジン は、次の 2 つの条件下で、定義アイテムの [キャッシング] セクションで定義されたキャッシュ基準を使用します:

- デベロッパーが、レンダリングをレイアウトまたはサブレイアウトに静的にバインドする場合、Sitecore はレンダリング定義アイテムからコントロールへキャッシング プロパティをコピーします (プレゼンテーション コンポーネントへの静的参照)。そのためには、定義アイテム内でキャッシング プロパティが設定されている必要があることにご注意ください。キャッシング プロパティの設定前にレンダリング定義アイテムでレイアウトまたはサブレイアウトが静的にバインドされている場合は、コピーされません。
- レイアウト詳細が、プレースホルダーに動的にバインドされているプレゼンテーション コンポーネントのキャッシング基準を指定しない場合。

レイアウト詳細を使用してプレースホルダーへのレンダリングを動的にバインドする場合、レイアウト詳細で明示的に定義されているキャッシュ設定が、レンダリング定義アイテムに定義されているキャッシュ設定を上書きします。定義アイテムに定義されているキャッシュ設定は、[コントロール プロパティ] ダイアログ ボックスの [キャッシング] セクションにキャッシングが設定されていない場合のみ適用されます。

4.4.3 出力キャッシング プロパティ

キャッシング可能なプレゼンテーション コンポーネントには次のプロパティがあります:

- **キャッシング可能 - キャッシング可能**プロパティは、*VaryBy* プロパティの設定内容に関わらず、プレゼンテーション コンポーネントをキャッシュするかどうかを定義します。false に設定されている場合、要求ごとにプレゼンテーション コンポーネントを実行します。true に設定されている場合、最初の要

求時にプレゼンテーション コンポーネントを実行し、その後の要求ではキャッシュから取得します。このプロパティが true に設定されていて、さらに 1 個以上の *VaryBy* プロパティが true に設定されている場合、プレゼンテーション コンポーネントを起動するか、それともキャッシュから取得するかを *VaryBy* プロパティが制御します。

- **VaryBy** プロパティ - **VaryBy** プロパティは、プレゼンテーション コンポーネントを実行するか、またはキャッシュから取得する場合を制御します。VarBy プロパティは、キャッシュ可能が true に設定されている場合のみ有効になります。
 - **VaryByData** - 異なるデータソースで使用される場合、異なる出力結果を生成するコンポーネントに対して true に設定します。
 - **VaryByDevice** - 異なるデバイスで使用される場合、異なる出力結果を生成するコンポーネントに対して true に設定します。
 - **VaryByLogin** - 承認されたユーザーと未承認ユーザーで、異なる出力結果を生成するコンポーネントに対して true に設定します。レイアウトエンジンは、すべての匿名ユーザーを単一の承認済みユーザーとして処理することにご注意ください。
 - **VaryByParm** - 異なるレンダリング パラメーターが渡された場合、異なる出力結果を生成するコンポーネントに対して true に設定します。
 - **VaryByQueryString** - 異なるクエリ文字列パラメーターが渡された場合、異なる出力結果を生成するコンポーネントに対して true に設定します。
 - **VaryByUser** - 異なるユーザーに対して異なる出力結果を生成するコンポーネントに対して true に設定します。メモリーの過剰な使用を避けるために、*VaryByUser* は比較的ユーザー数の少ないソリューションでのみ使用してください。

メモ

Visual Studio を使用して、レイアウトまたはサブレイアウトに対してレンダリングを静的に配置している場合、キャッシング プロパティを手動で設定できます。次はその例です: `<sc:sublayoutrunat="server" renderingid="D366A65-54FF-49B5-A57F-2EBB9F426433%7d" placeholder="content" cacheable="true" varybydata="true" path="/layouts/Starter Kit/Sublayouts/Header Fixed.ascx" id="HeaderFixed"></sc:sublayout>`

4.4.4 出力キャッシングの推奨プラクティス

Sitecore の出力キャッシングの設定時のガイダンスとなる推奨プラクティスの一覧は次のとおりです:

- プレゼンテーション コンポーネントの起動に必要な CPU 使用率を削除することにより、Sitecore 出力キャッシングのレンダリング パフォーマンスに大きなメリットをもたらします。出力キャッシングの使用により最大の効果を得るには、プレゼンテーション コンポーネントをキャッシング可能にする前に適切なコーディング プラクティスに従っていることを確認する必要があります。

- `htmlCacheSize` プロパティが、すべてのレンダリングをキャッシングするのに十分な大きさであることを確認します。設定の影響の詳細については、「初期キャッシュ値の設定」のセクションを参照してください。
- 子コントロールを含むコンテナ (レイアウト、プレースホルダー、サブレイアウト) のキャッシングにより、以降の要求では、コンテナのすべてのレンダリング結果をキャッシュから取得することとなります。これには、コンテナ内のすべての子コントロールを含みます。
- `stats.aspx` ページを使用し、プレゼンテーション コンポーネントのキャッシング活動とレンダリング時間を監視します。
- カスタマイズされたキャッシング基準の作成の詳細は、次の Sitecore ブログを参照してください:
<http://www.sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog/Posts/2011/05/Custom-Caching-Criteria-with-the-Sitecore-ASPNET-CMS.aspx>

4.5 キャッシュ サイズの制限の無効化

このタスクは Sitecore 6.5.1 アップデート 4 以降を使用しているシステムにキャッシュ サイズの制限を無効にするものです。キャッシュ サイズの無効化によって、十分なメモリ量を持つシステムが、他のシステムからのメモリの使用を制限することなくキャッシュを行い、有効なメモリを使用することが可能になります。

4.5.1 必要スキル

- Sitecore 構成ファイルの実務知識

4.5.2 症状

- キャッシュを行うメモリ量の制限
- メモリ制限による頻繁なキャッシュの削除
- ページのレンダリング パフォーマンスの低下

4.5.3 キャッシュ サイズの制限の無効化手順

キャッシュ サイズの制限を無効化にする方法:

1. エディターで、web.config ファイル(Web ルート ディレクトリ内にあります) を開きます。
2. `<setting name="Caching.DisableCacheSizeLimits" value="false" />` に移動して、値を"true"に設定します。

```
<setting name="Caching.DisableCacheSizeLimits" value="true" />
```

3. 変更を保存します。

4.5.4 推奨事項

キャッシュ サイズの制限は、十分なメモリを持つ 64 ビット システム上で無効化することを推奨します。

4.5.5 解決方法

Caching.DisableCacheSizeLimits の設定値を true に設定するための手順に従います。

Chapter 5

チューニング手順 - IIS 設定

チューニング手順 - IIS 設定には、IIS Web サーバーの設定をチェックするために設計された一連のタスクが含まれています。IIS を適切に設定することにより、Sitecore 実装は最善のパフォーマンスで実行できます。

この章には次のセクションがあります。

- HTTP キープアライブの有効化
- IIS の Expire Web コンテンツ ヘッダー
- IIS の静的なコンテンツの圧縮の有効化
- IIS の動的なコンテンツの圧縮の有効化 (オプション)

5.1 HTTP キープアライブの有効化

HTTP キープアライブを有効にすることにより、確立する必要がある接続数を削減できます。HTTP キープアライブが無効になっている場合、要求された Web ページ上のすべてのオブジェクトに対して新たな接続が確立されます。

5.1.1 必要なスキル

- 実行中の IIS マネージャー。

5.1.2 症状

- 要求ページの読み込み時間が常に長い。
- 低いパフォーマンス。

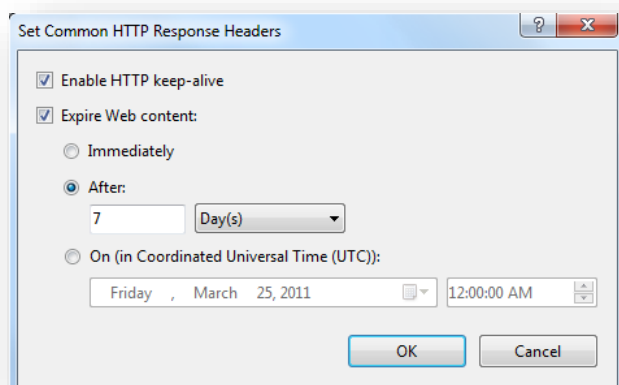
5.1.3 手順

HTTP キープアライブが有効になっているかどうかの確認方法:

1. IIS マネージャーを起動します。
2. HTTP キープアライブが有効であるかを確認したいサイトに移動します。
3. [HTTP 応答ヘッダー] をダブルクリックします – IIS グルーピング内にあります。
4. [操作] パネルで、[共通ヘッダーの設定] をクリックします。

5.1.4 結果の理解

[共通 HTTP 応答ヘッダーの設定] ダイアログが表示されます。次の例では、HTTP キープアライブが有効になっています。



5.1.5 推奨事項

Sitecore は、HTTP キープアライブを有効にすることを推奨しています。

5.1.6 解決方法

HTTP キープアライブを有効にする方法:

1. **IIS マネージャー**を起動します。
2. **HTTP キープアライブ**を有効に設定するサイトに移動します。
3. **[HTTP 応答ヘッダー]** をダブルクリックします (IIS グループ内にあります)。
4. **[操作]** パネルで、**[共通ヘッダーの設定]** をクリックします。
5. **[HTTP キープアライブの有効化]** を選択します。
6. **[OK]** をクリックします。

5.2 IIS の Expire Web コンテンツ ヘッダー

Expire Web コンテンツ ヘッダー ([共通 HTTP 応答ヘッダー] 内にあります) では、Web ページ コンテンツの期限が切れた後に送信された要求に対し、要求 Web ページの新しいバージョンを返すかどうかを IIS がどのように決定するかを指定します。IIS はコンテンツの期限切れのための設定を使用し、送信前に各 Web ページをマークします。エンドユーザーのブラウザは期限切れマークを変換します。

Expire Web コンテンツを即時以外に設定することで、2 次アクセスの読み込み時間を 50% から 70% 削減できます。この設定は、動的に生成されたコンテンツには影響を与えません。

下記の手順は IIS 7.x 用であることにご注意ください。過去のバージョンで Expire Web コンテンツ ヘッダーを有効にする方法の詳細は、Microsoft のドキュメントを参照してください。

5.2.1 必要なスキル

- 実行中の IIS マネージャー。

5.2.2 症状

- 要求ページの読み込み時間が常に長い。
- 低いパフォーマンス。

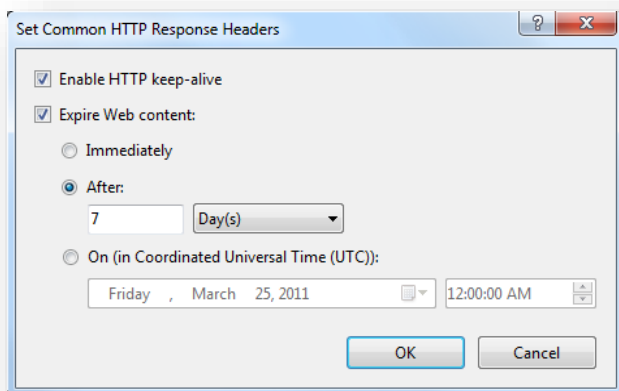
5.2.3 手順

Expire Web コンテンツ ヘッダーが有効になっているかどうかの確認方法:

1. **IIS マネージャー**を起動します。
2. **Expire Web コンテンツ ヘッダー**が有効であるかを確認したいサイトに移動します。
3. **[HTTP 応答ヘッダー]**をダブルクリックします (IIS グループ内にあります)。
4. [操作] パネルで、**[共通ヘッダーの設定]**をクリックします。

5.2.4 結果の理解

[共通 HTTP 応答ヘッダーの設定] ダイアログが表示されます。次の例では、Expire Web コンテンツが有効になっており、失効までの期間が 7 日に設定されています。



5.2.5 推奨事項

Sitecore は、Expire Web コンテンツの「失効までの期間」を 30 日に設定することを推奨しています。

5.2.6 解決方法

Expire Web コンテンツ ヘッダーを有効にする方法:

1. IIS マネージャーを起動します。
2. Expire Web コンテンツ ヘッダーを有効に設定するサイトに移動します。
3. [HTTP 応答ヘッダー] をダブルクリックします (IIS グループ内にあります)。
4. [操作] パネルで、[共通ヘッダーの設定] をクリックします。
5. [期限切れの Web コンテンツ] を選択します。
6. [失効までの期間] を選択します。
7. 日数を 30 に設定します。
8. [OK] をクリックします。

5.3 IIS の静的なコンテンツの圧縮の有効化

IIS の静的なコンテンツの圧縮を有効にすることにより、CPU リソースを低下させることなく、静的な応答を圧縮して、複数の要求にまたがりディスク内でキャッシュすることができます。

デフォルトでは、IIS の静的コンテンツの圧縮は有効に設定されています。このタスクでは、実際に有効になっているかどうかを確認します。

5.3.1 必要なスキル

- 実行中の IIS マネージャー。

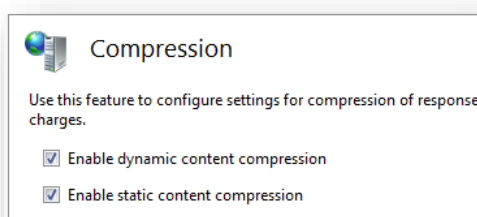
5.3.2 症状

- 要求ページの読み込み時間が常に長い。
- 低いパフォーマンス。

5.3.3 手順

静的なコンテンツの圧縮が有効になっているかどうかの確認方法:

1. IIS マネージャーを起動します。
2. **静的なコンテンツの圧縮**が有効であるかを確認したいサイトに移動します。
3. **[圧縮]** アイコンをダブルクリックします (IIS グループ内にあります)。
4. **[圧縮]** ダイアログが表示されます。次の例では、静的なコンテンツの圧縮が有効になっています。



5.3.4 推奨事項

Sitecore は、**IIS の静的なコンテンツの圧縮**の設定を有効にすることを推奨しています。

5.3.5 解決方法

IIS の静的なコンテンツの圧縮を有効にする方法:

1. **IIS マネージャー**を起動します。
2. **静的なコンテンツの圧縮**が有効であるかを確認したいサイトに移動します。
3. **[圧縮]** アイコンをダブルクリックします (IIS グループ内にあります)。
4. **[静的なコンテンツの圧縮を有効にする]** チェックボックスをオンにします。
5. **[適用]** をクリックします。

5.4 IIS の動的なコンテンツの圧縮の有効化 (オプション)

動的なコンテンツの圧縮の有効化は、CPU リソースがどの程度利用できるかによって効果が異なるため、オプションとなっています。

動的なコンテンツの圧縮は、動的コンテンツの応答を圧縮することにより、必要となる帯域幅を低減できます。動的コンテンツは要求/応答ごとに圧縮されるため、帯域幅を低減する代わりに CPU 利用率が高くなります。そのため、CPU 使用率が既に高いシステムの場合、動的なコンテンツの圧縮を有効にすべきではありません。

動的な圧縮を使用したメリットおよびその設定方法を紹介した 2 つの記事を、次の示します。

- <http://weblogs.asp.net/owscott/archive/2009/02/22/iis-7-compression-good-bad-how-much.aspx>
- <http://www.west-wind.com/weblog/posts/2011/May/05/Builtin-GZipDeflate-Compression-on-IIS-7x>

5.4.1 必要なスキル

- 実行中の IIS マネージャー。

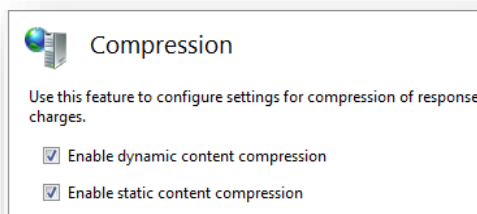
5.4.2 症状

- 要求ページの読み込み時間が常に長い。
- 低いパフォーマンス。

5.4.3 手順

動的なコンテンツの圧縮が有効になっているかどうかの確認方法:

1. IIS マネージャーを起動します。
2. **動的なコンテンツの圧縮**が有効であるかを確認したいサイトに移動します。
3. **[圧縮]** アイコンをダブルクリックします (IIS グループ内にあります)。
4. **[圧縮]** ダイアログが表示されます。次の例では、動的なコンテンツの圧縮が有効になっています。



5.4.4 推奨事項

Sitecore は、CPU 使用率が常に高い値を示しているシステム以外では **IIS の動的なコンテンツの圧縮** を有効にすることを推奨しています。

5.4.5 解決方法

IIS の静的なコンテンツの圧縮を有効にする方法:

1. **IIS マネージャー** を起動します。
2. **動的なコンテンツの圧縮** が有効であるかを確認したいサイトに移動します。
3. **[圧縮]** アイコンをダブルクリックします (IIS グループ内にあります)。
4. **[動的なコンテンツの圧縮を有効にする]** チェックボックスをオンにします。
5. **[動的なコンテンツの圧縮を有効にする]** チェックボックスがグレーアウトしている場合、**[コントロールパネル]**、**[プログラムの追加と削除]**、**[Windows コンポーネントの追加と削除]** で IIS の動的圧縮モジュールを追加してください。
6. **[適用]** をクリックします。

Chapter 6

チューニング手順 - Sitecore のクライアント最適化

チューニング手順 - Sitecore のクライアント最適化には、Sitecore クライアントツールの設定と推奨プラクティスをチェックするために設計された一連のタスクが含まれています。

適切な設定と、次に示す推奨プラクティスにより、Sitecore クライアント ツール使用時の、ユーザー エクスペリエンスの最善なパフォーマンスを確保するのに役立ちます。

この章には次のセクションがあります。

- 実行時間の長いバリデーターのチェック
- 過剰なアイテム バージョンのチェック
- ノードごとの過剰アイテムのチェック
- その他のクライアント固有の最適化

6.1 実行時間の長いバリデーターのチェック

実行時間の長いバリデーターは、Sitecore コンテンツ エディターの使用時のパフォーマンスに悪影響を与えます。このタスクでは、そのような検証ルールを見つけるために設計されています。

6.1.1 必要なスキル

- Sitecore ログ解析の実務知識。

6.1.2 症状

- Sitecore コンテンツ エディター使用時の低いパフォーマンス。

6.1.3 実行時間の長い検証ルールのチェック手順

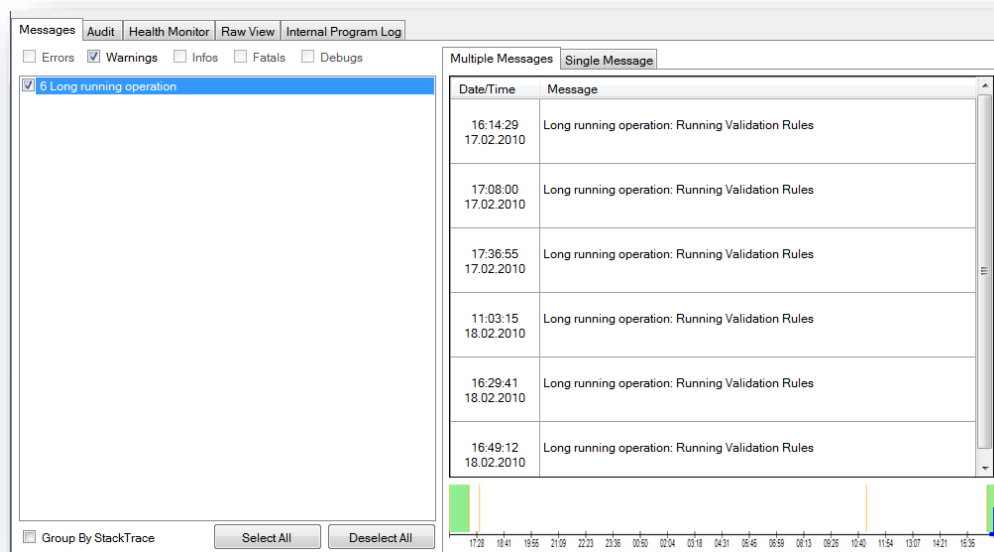
実行時間の長い検証ルールは、Sitecore ログ内で次のメッセージと共に表示されます: *Long running operation: Running Validation Rules*. このメッセージを見つけるには、お使いのテキスト エディターで上記の文字列を検索するか、Sitecore Log Analyzer tool でログ ファイルを解析します。

Sitecore Log Analyzer を使用してログ ファイルを解析する方法:

1. **Sitecore Log Analyzer** を起動します。
2. **[Select Location]** ボタンを使って構文解析用のログ ファイルを選択します。
3. **[String filter]** フィールドを *Long running operation: Running Validation Rules* 値に設定します。
4. **[Analyze / Refresh*]** ボタンを押します。
5. **[Messages]** タブに移動して、**[Warnings]** チェックボックスをオンにします。
6. メッセージのリストにあるエントリ **[“* Long running operation”]** のチェックボックスをオンにします。

6.1.4 結果の理解

出力結果は、次のようなものとなります:



上記の表は、実行時間の長い検証ルールにより、6つの警告が生じたことを示しています。どの検証ルールにより警告が生じたかは、ログファイルからは分かりません。

6.1.5 推奨事項

Sitecore は、Sitecore コンテンツ エディター使用時のパフォーマンス向上のため、実行時間の長い検証ルールを調査し、無効にすることを推奨しています。

6.1.6 解決方法

どの検証ルールから警告が生じたかを突き止めるには、多少の試行錯誤が必要となります。影響を与えている検証ルールを突き止めるのに役立つ方法を次に説明します:

1. カスタマイズされた検証ルールの実行時間が長くないかを確認します。長い場合は、無効にします。
2. 不要な標準のバリデーターは、アイテムの検証ルール セクションから削除して無効にします。
3. `web.config` ファイルで宣言されているバリデーターもあります (メディア ライブラリ検証ルールなど)。

6.2 過剰なアイテム バージョンのチェック

1 つ以上のアイテムで過剰なアイテム バージョンを保持することは、パフォーマンスに悪影響を与える場合があります。特に、コンテンツ エディター内のコンテンツ ツリーの表示に関連するパフォーマンス、およびキャッシングに使用されるメモリー量に影響を与えます。

バージョン数が増えると、すべてのバージョンの処理に必要な時間も増えます。たとえば、コンテンツ エディターで表示または編集するアイテムを開く場合、表示されているバージョンだけではなく、そのアイテムに関連するすべてのバージョンも処理されます。

このタスクは、過剰なバージョン数、およびバージョン管理の共有ソース モジュールへのポインターを見つけ出すよう設計されています。

6.2.1 必要なスキル

- Sitecore コンテンツ エディターの実務知識。
- 共有ソース モジュールのインストールおよび設定経験。

6.2.2 症状

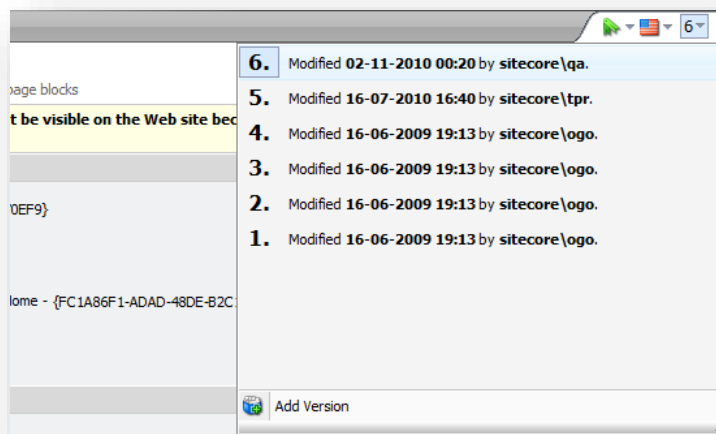
- Sitecore コンテンツ エディター使用時の低いパフォーマンス。

6.2.3 過剰なアイテム バージョンのチェック手順

通常、大幅に変更されたアイテムには過剰なバージョン数が存在します (サイトのホームページなど)。どのページが最も頻繁に更新されているかを、コンテンツ作成者に確認します。

1. **Sitecore コンテンツ エディター**を起動します。
2. コンテンツ ツリー内で、該当するアイテムに移動します。

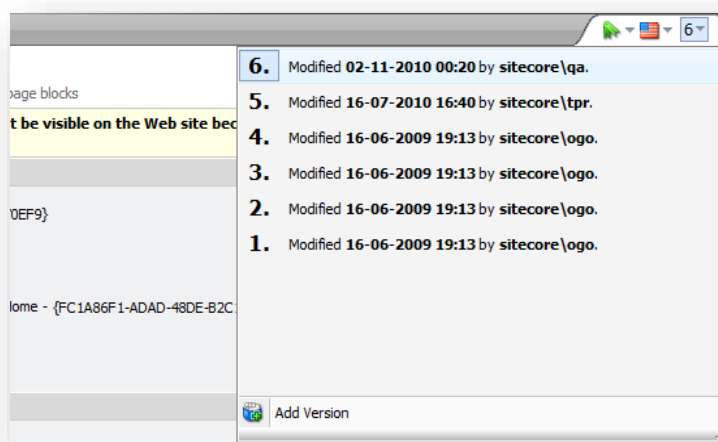
3. アイテムの右上隅に、表示されているアイテムのバージョン数を示すドロップダウンリストがあります:



4. アイテムのバージョン数にご注意ください (10 以上のバージョン数の場合は、過剰というフラグを付けます)。
5. このプロセスを、他の該当アイテムについても繰り返します。

6.2.4 結果の理解

出力結果は、次のようなものとなります:



この画面は、現在ホームページには 6 個のバージョンが保存されていることを示しています。

6.2.5 推奨事項

Sitecore は、特定のアイテムに対して 10 以下のバージョン数に限定することを推奨しています。ただし、会社のポリシーに保存すべきバージョン数が指定されている場合、推奨より多くのバージョンを保存することも

可能です。また、Sitecore は保存バージョン数の管理のためにバージョン マネージャーの共有ソース モジュールの実行を推奨しています。

6.2.6 解決方法

保存するバージョン数を管理するには、次のバージョン マネージャー共有ソースを活用することができます:
<http://trac.sitecore.net/VersionManager>

6.3 ノードごとの過剰アイテムのチェック

コンテンツ ツリーの構造は、いかなる親ノードの下にも過剰なアイテム数が配置されることがないように、バランスよく配置しなければなりません。アイテム数は 100 を越えてはなりません。アイテムの整理にフォルダーを使用した構造を活用し、注意深く計画をたてる必要があります。

ある親ノード下に 100 を越える過剰なアイテム数がある場合、コンテンツ エディターでコンテンツ ツリー内を移動中の Sitecore ユーザーのパフォーマンスが低下します。

このタスクでは、コンテンツ ツリー内の過剰なアイテム数を見つけるために設計されています。

6.3.1 必要なスキル

- Sitecore コンテンツ エディターの実務知識。

6.3.2 症状

- Sitecore コンテンツ エディター使用時の低いパフォーマンス。

6.3.3 ノードごとの過剰なアイテム数のチェック手順

100 以上のアイテムを含むノードの検索は、手動タスクで実行することも、スクリプトを作成して自動処理することも可能です。過剰なアイテム数を含むノードを手動検索する方法:

1. Sitecore コンテンツ エディターを起動します。
2. コンテンツ ツリーで、すべてのノードを展開します。
3. 100 アイテム以上あるノードを探します。

スクリプトを使用して、子の数が過剰に多いノードを探すには、指定されたホーム ノードから、コンテンツ ツリーを繰り返し確認するスクリプトを作成します。次はその例です:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Sitecore.Data.Items;
using Sitecore.Collections;

namespace Sitecore.Utilities
{
    public partial class NodeCount : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Item homeItem =
                Sitecore.Configuration.Factory.GetDatabase("master").Items["/sitecore/content/home"];
            getNumberOfChildren(homeItem);
        }

        private void getNumberOfChildren(Item node)
        {

```

```
ChildList children = new ChildList(node);
foreach (Item child in children)
{
    if (child.HasChildren && child.Children.Count >= 100)
    {
        Response.Write("Node: " + child.Name + " - " + child.Children.Count + "<br
/>");
        getNumberOfChildren(child);
    }
    else
    {
        Response.Write("<br />");
    }
}
}
```

6.3.4 結果の理解

100 を越えるアイテム数を含むノードの数により、結果は異なります。100 を越えるアイテム数を含むすべてのノードには、注意のためにフラグを立てる必要があります。

6.3.5 推奨事項

Sitecore は、すべての親ノード下のアイテム数を 100 以下にすることを推奨しています。

6.4 その他のクライアント固有の最適化

次に、Sitecore クライアントのパフォーマンスを改善すると考えられる一連の最適化を示します。

このタスクでは、その設定を `web.config` ファイルに追加する方法について説明します。

6.4.1 必要なスキル

- Sitecore `web.config` ファイルの実務知識。

6.4.2 症状

- Sitecore コンテンツ エディター使用時の低いパフォーマンス。

6.4.3 ContentEditor.RenderCollapsedSections の設定

`ContentEditor.RenderCollapsedSections` は `/web.config file` の隠し設定であり、デフォルトでは `true` に設定されています。<setting name="ContentEditor.RenderCollapsedSections" value="false" /> を `/web.config` ファイルに追加することで、特に大きなコンテンツ ツリーを持つクライアントのパフォーマンスを改善します。

また、値を `false` に設定することで、プリフェッチ キャッシングが指定された子をメモリーに 2 度追加してしまう状況を防ぐことができます。

コンテンツ エディターでスタンダード フィールドを表示してはなりません。

コンテンツ エディターのスタンダード フィールドの表示を無効にすることにより、アイテムごとの情報のレンダリングを減らすことが可能になり、パフォーマンスが改善します。

6.4.4 コンテンツ作成者は完全公開を使用不可

完全公開、または全アイテムの公開には、多くのリソースを消費します。完全公開中は、操作中にシステムを使用中のすべての Sitecore ユーザーのパフォーマンスが低下する恐れがあります。最適化公開または差分公開を使用することを強く推奨します。

6.4.5 ContentEditor.CheckHasChildrenOnTreeNodees の設定

この設定は、ツリー ノードのレンダリング時にエディターが `HasChildren` メソッドを使用するかどうかを指定します。パフォーマンスを改善するためにこの値を `false` に設定します。

Chapter 7

チューニング手順 - Sitecore の多様なコンテンツ デリバリー

— サーバーの最適化

この章では、コンテンツ デリバリー サーバーの設定と推奨プラクティスをチェックするために設計された一連のタスクが含まれています。

この章で説明する最適化は、通常診断やコンテンツ管理機能のために使用される機能を無効にするためのものです。これらの機能を無効にすることで、CPU のプロセス時間を回復することができます。

適切な設定と以下の推奨プラクティスを行うことで、Sitecore のコンテンツ デリバリー サーバーを最適なパフォーマンス レベルで実行することを確実にします。

この章には次のセクションがあります。

- WebDAV の無効化
- パフォーマンス カウンターの無効化
- メモリモニタの無効化

7.1 WebDAV の無効化

WebDAV を使用することによって、ブラウザなどの Web クライアントのユーザーが、HTTP または HTTPS などの Web プロトコルを使って Web サーバー上でファイルを管理できるようになります。

コンテンツ デリバリー サーバー上で WebDAV を無効化するとは、これらの環境で使用されないことを意味します。

7.1.1 必要なスキル

- Sitecore 構成ファイルの実務知識。

7.1.2 症状

- コンテンツ デリバリー サーバー上に不必要なログ ファイルの作成。

7.1.3 WebDAV を無効化にする手順

WebDAV を無効化にする方法:

1. エディターで web.config ファイルを開きます — Web ルートのディレクトリ内にあります。
2. <log4net/> セクションに移動して WebDAV の参照を削除します。

```
<log4net>
  <appender name="LogFileAppender" type="log4net.Appender.SitecoreLogFileAppender, Sitecore.Logging">
    <file value="$(dataFolder)/logs/log.{date}.txt" />
    <appendToFile value="true" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%4t %d{ABSOLUTE} %-5p %m%n" />
    </layout>
  </appender>

  <!--<appender name="WebDAVLogFileAppender" type="log4net.Appender.SitecoreLogFileAppender,
Sitecore.Logging">
    <file value="$(dataFolder)/logs/WebDAV.log.{date}.txt" />
    <appendToFile value="true" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%4t %d{ABSOLUTE} %-5p %m%n" />
    </layout>
  </appender-->
</root>
<priority value="INFO" />
<appender-ref ref="LogFileAppender" />
</root>

<!--<logger name="Sitecore.Diagnostics.WebDAV" additivity="false">
  level value="INFO" />
  <appender-ref ref="WebDAVLogFileAppender" />
```

```
</logger>-->  
</log4net>
```

3. <system.webServer/> セクションに移動して、WebDAV の参照を削除します。

```
<!--<remove name="WebDAVModule" />-->  
<!--  
<add name="WebDAVRoot" path="*" verb="OPTIONS,PROPFIND" modules="IsapiModule"  
scriptProcessor="%windir%\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll" resourceType="Unspecified"  
preCondition="classicMode, runtimeVersionv2.0, bitness32" />  
<add name="WebDAVRoot64" path="*" verb="OPTIONS,PROPFIND" modules="IsapiModule"  
scriptProcessor="%windir%\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll"  
resourceType="Unspecified" preCondition="classicMode, runtimeVersionv2.0, bitness64" />  
<add verb="*" path="sitecore_webDAV.ashx" type="Sitecore.Resources.Media.WebDAVMediaRequestHandler,  
Sitecore.Kernel" name="Sitecore.WebDAVMediaRequestHandler" />  
-->
```

4. <httpHandlers/> セクションに移動して、WebDAV の参照を削除します。

```
<!--  
<add verb="*" path="sitecore_webDAV.ashx" type="Sitecore.Resources.Media.WebDAVMediaRequestHandler,  
Sitecore.Kernel" />  
-->
```

5. 変更を保存します。

6. 最後に webroot/App_Config/Include ディレクトリから Sitecore.WebDAV.config ファイルを削除することができます。

7.1.4 推奨事項

Sitecore では、多数のログ ファイルの作成を減少させるために、実働コンテンツ デリバリー サーバー上で WebDAV を無効にすることを推奨しています。また WebDAV 機能が使用されていない場合は、コンテンツ管理サーバー上でも WebDAV を無効にすることを推奨しています。

7.2 パフォーマンス カウンターの無効化

パフォーマンス カウンターは少数のオーバーヘッドを作成します。トラブルシューティング モードを開始する場合のみ無効化することを推奨します。

7.2.1 推奨スキル

- Sitecore の構成ファイルの実務知識。

7.2.2 症状

- パフォーマンス カウンターの維持を要求する少数のオーバーヘッド。

7.2.3 パフォーマンス カウンターの無効化手順

パフォーマンス カウンターを無効化にする方法:

1. エディターで web.config ファイルを開きます — Web ルート ディレクトリ内にあります。
2. `<setting name="Counters.Enabled" value="true" />` セクションに移動し、バリューを false に設定します。

```
<setting name="Counters.Enabled" value="false" />
```

3. 変更を保存します。

7.2.4 推奨事項

Sitecore では、カウンターの維持を要求するオーバーヘッドを減少するために、実働コンテンツ デリバリー サーバー上でパフォーマンス カウンターを無効にすることを推奨しています。

7.3 メモリモニタの無効化

メモリモニタは頻繁に CPU スパイクを作成します。メモリに関連する問題のトラブルシューティングを行う場合にのみ無効化します。

7.3.1 推奨スキル

- Sitecore 構成ファイルの実務知識。

7.3.2 症状

- 頻繁な CPU スパイク。

7.3.3 メモリモニタの無効化手順

WebDAV を無効化する方法:

1. エディター (Web ルート ディレクトリ内にあります) で web.config ファイルを開きます。
2. <hooks/> セクションに移動し、

<hook type="Sitecore.Diagnostics.MemoryMonitorHook, Sitecore.Kernel" /> をコメントアウトします。

```
<hooks>
<hook type="Sitecore.Diagnostics.HealthMonitorHook, Sitecore.Kernel" />
<!--<hook type="Sitecore.Diagnostics.MemoryMonitorHook, Sitecore.Kernel">
  <param desc="Threshold">800MB</param>
  <param desc="Check interval">00:00:05</param>
  <param desc="Minimum time between log entries">00:01:00</param>
  <ClearCaches>>false</ClearCaches>
  <GarbageCollect>>false</GarbageCollect>
  <AdjustLoadFactor>>false</AdjustLoadFactor>
</hook-->
</hooks>
```

3. 変更を保存します。

7.3.4 推奨事項

実稼働環境においてメモリモニタを無効化することを推奨します。メモリに関連する問題のトラブルシューティングを行う場合にのみ無効化します。

Chapter 8

チューニング手順 - Sitecore 検索

この章には、コンテンツ デリバリー サーバー上での Sitecore 検索の設定と推奨プラクティスをチェックするために設計された一連のタスクが含まれています。

この章で説明する最適化は、通常診断のために使用される機能を無効化するためのものです。これらの機能を無効化することで、CPU プロセスの時間が回復され、オーバーヘッドを減少することができます。

適切な設定と以下の推奨プラクティスを活用することによって、Sitecore コンテンツ デリバリー サーバーと Sitecore に基づく検索を最適なパフォーマンス レベルで実行することを確実にします。

この章には次のセクションがあります。

- バケットのデバッグの無効化
- 追加情報

8.1 バケットのデバッグの無効化

バケットのデバッグが有効な場合は、アイテム バケットに対して実行されたすべてのクエリは、検索のログ ファイルに記入されます。

8.1.1 推奨スキル

- Sitecore の構成ファイルの実務知識。

8.1.2 症状

- 大規模なログ ファイルに検索クエリの蓄積。

8.1.3 バケット デバッグの無効化手順

- エディターで、App_Config/Include/Sitecore.Buckets.config ファイルを開きます。
- BucketConfiguration.EnableBucketDebug 設定を false に設定します。

```
<setting name="BucketConfiguration.EnableBucketDebug" value="false" />
```

- 変更を保存します。

8.1.4 推奨事項

Sitecore では、コンテンツ デリバリー サーバーにログインするバケット デバッグを無効化することを推奨しています。これにより、ログ ファイルのサイズを縮小し、ログ ファイルのポリューションを軽減し、ログ ファイルの記入と維持を要求するオーバーヘッドを最小限にすることができます。

8.2 追加情報

Sitecore 検索と、使用環境にとって適切な検索操作の設定方法についての追加情報は、以下の SDN にあるドキュメントを参照してください。

- [Developer's Guide to Item Buckets and Search](#) (アイテム バケットと検索についての開発ガイド)
- [Sitecore Search Scaling Guide](#) Sitecore (検索の拡張性ガイド)
- [Sitecore Search Admin Guide](#) Sitecore (検索の管理ガイド)