

## 第13回 DB2とWebSphereのパフォーマンスチューニング概略

沖林 正紀

### 1. はじめに

今回はこの連載の最後として、WebSphereやDB2におけるパフォーマンスチューニングについて取り上げます。それぞれのソフトウェアに対するチューニングやアプリケーションに対するチューニングの概略をご紹介します。詳細はリンクの引用元のサイトもしくはファイルを参考にしてください。

### 2. DB2のパフォーマンスチューニング

DB2 UDBに対してパフォーマンスチューニングを施す上での指針を以下に示します。パフォーマンスの改善に終わりはありませんが、この指針を元に目標を決めてチューニングを進めると良いでしょう(DB2 オンラインマニュアル 管理の手引きより)。

#### 収穫逓減の法則を覚えておく

パフォーマンスの効果を最大にするには、最初の努力が重要です。一般にその後の変更では、得られる効果は少なくなり、必要な努力は多くなります。

#### チューニングのためのチューニングを行わない

チューニングは、識別されている制約を軽減するために行ってください。パフォーマンス上の問題の主要な原因ではないリソースのチューニングを行っても、応答時間に対する効果は主要な制約を軽減するまでほとんどまたはまったくなく、それ以降のチューニング作業が行いにくくなります。著しく改善できる可能性があるとなれば、それは応答時間の主要な要因となっているリソースのパフォーマンスを改善することにあります。

#### システム全体を考慮する

他に影響を与えることなく1つのパラメーターまたはシステムのチューニングを行うことはできません。調整を行う前に、その調整によってシステム全体がどのような影響を受けるかを考慮してください。

#### 一度に1つのパラメーターを変更する

一度に複数のパフォーマンス・チューニング・パラメーターを変更しないでください。すべての変更が有益であることを確信している場合でも、それぞれの変更の効果を評価することができなくなります。一度に複数のパラメーターを変更すると、行ったトレードオフを効果的に判断することもできません。1つのパラメーターを調整して1つの分野を改善すると、考慮に入れていなかった少なくとも1つの別の分野が影響を受けます。

#### レベルごとに測定と再構成を行う

一度に変更するパラメーターを1つにするのと同じ理由で、一度にチューニングするシステムのレベルは1つにしてください。次のリストは、システム内の各レベルを示しています。

#### ハードウェア

#### オペレーティング・システム

#### アプリケーション・サーバーとリクエスター

#### データベース

#### SQLステートメント

#### アプリケーション・プログラム

#### ハードウェアおよびソフトウェアの問題を検査する

一部のパフォーマンス問題は、ハードウェアかソフトウェアのどちらか(あるいはその両方)にサービスを適用することによって修正することができます。単にサービスを適用するだけで済む場合は、システムのモニターとチューニングに余分な時間がかかりません。

#### ハードウェアをアップグレードする前に問題を理解する

記憶域を増やしたりプロセッサの能力を高めたりすることでパフォーマンスを即座

### [1.はじめに](#)

### [2.DB2のパフォーマンスチューニング](#)

### [3.WebSphereのパフォーマンスチューニング](#)

### [4.アプリケーションのパフォーマンスチューニング](#)



沖林 正紀

IBMソフトウェア事業部  
データ・マネジメント事業  
推進技術部のメンバー

に改善できるように思えても、時間を取って障害がどこに存在しているのかを理解してください。費用をかけてディスク装置を追加しても、それを活用するだけの処理能力やチャンネルがないことに気付く場合があります。

チューニングを始める前にフォールバック手順を実施する

前述のとおり、チューニングを行うと予想外のパフォーマンス結果が生じることがあります。パフォーマンスが低下した場合は、そのチューニングを元に戻して別のチューニングを行う必要があります。簡単に元に戻せるように元の設定を保管しておけば、誤った情報を取り消すことはずっと簡単になります。

パフォーマンスチューニングを行うための基礎データを揃えるのに使用するツールとして、スナップショットモニター、イベントモニター、パフォーマンスモニター、SQL Explain機能があります。それぞれの概要をご紹介します。

### スナップショットモニター

(DB2 オンラインマニュアル システム・モニター 手引きおよび解説書より抜粋)

スナップショット・モニターを使用すると、個々のモニター・レベルにおいて以下の2つのカテゴリーの情報が得られます。

- 状態
  - データベースの現在の状況
  - 現在または最新の作業単位に関する情報
  - アプリケーションに保持されているロックのリスト
  - アプリケーションの状況
  - 現在データベースに接続している数
  - アプリケーションによって最後に実行された SQLステートメント
  - 構成可能システム・パラメーターの実行時の値
- カウンター
  - モニターを開始してからスナップショットが取られるまでの活動のカウンタを累算します。たとえば、以下のものをカウントできます。

#### デッドロックの発生数

データベースに関して実行されたトランザクションの数

アプリケーションがロックのために待機した時間<

### イベントモニター

(DB2 オンラインマニュアル システム・モニター 手引きおよび解説書より抜粋)

スナップショットは特定の時点で取るのに対して、イベント・モニターは以下のいずれかのイベントが起こった時点で、データベース・システム・モニターのデータをファイルまたは名前付きパイプに書き出します。

- トランザクション終了時
- ステートメント終了時
- デッドロック時
- 接続開始時
- 接続終了時
- データベース活動化時
- データベース非活動化時
- ステートメントのサブセクション終了時(データベースが区分化されている場合)
- フラッシュ・イベント・モニター・ステートメントの発行時

イベント・モニターによって戻される情報は、スナップショット API を使用して得られる情報と似ています。つまり、スナップショットが取られる時点を制御するイベントに関する情報が戻されます。たとえば、接続イベント・モニターの場合は、基本的には接続が終了する直前にアプリケーションのスナップショットが取られます。

### パフォーマンスモニター

(DB2 オンラインマニュアル 管理の手引きより抜粋)

パフォーマンス・モニターは、DB2 ユニバーサル・データベース、およびそれが制御するデータの状態についての情報を提供します。これは、データベース環境に合わせてカスタマイズ可能なグラフィカル・ユーティリティです。パフォーマンス・モニターが収集する値が受け入れ可能な範囲内でない場合に、警告またはアラームの引き金となる限界値またはゾーンを定義することができます。

パフォーマンス・モニターからの情報は、以下の目的で使用します。

- パフォーマンス上の問題の検出
- 最適パフォーマンスを得るためのデータベースの調整

- パフォーマンス傾向の分析
- データベース・アプリケーションのパフォーマンスの分析
- 問題発生の予防

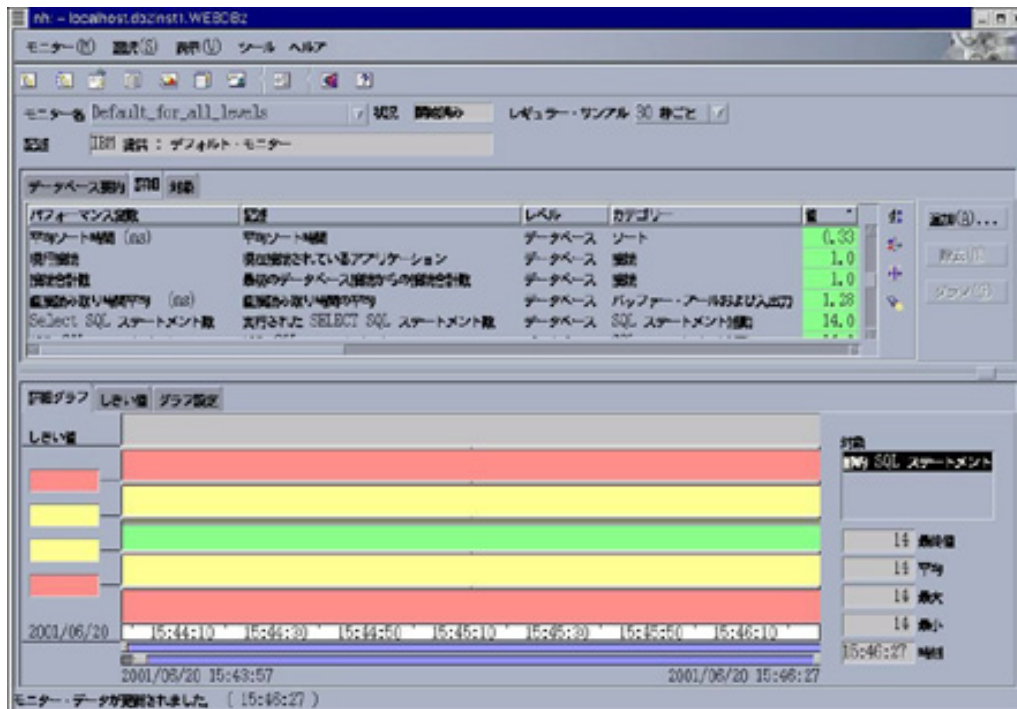
パフォーマンス・モニターを使用すると、ディスク活動、バッファ・プール使用状況、事前取り出しの量、ロックの使用状況、および特定の間隔でのレコード・ブロックなど、データベース情報のビジュアルな表示を作成することによって、傾向を分析することができます。

既存の問題のモニターが必要なとき、またはシステムのパフォーマンスを監視したい場合、このツールを使用します。これによってデータベース活動のスナップショット、および指定時刻でのパフォーマンス・データをとることができます。これらのスナップショットは、時系列の比較に使用されます。

(執筆注:参考)

パフォーマンスモニターを動作させた時の画面は以下ようになります。この画面はコントロールセンターから起動させることができます。なお、コントロールセンターを起動させるにはIBM JDK1.1.8が必要です。

パフォーマンスモニターの起動画面



## SQL Explain機能

(DB2 オンラインマニュアル 管理の手引きより抜粋)

SQL Explain 機能はSQLコンパイラの一部で、この機能を使用すると、静的または動的SQLステートメントのコンパイルを行う環境に関する情報を取得できます。取得した情報を使用して、SQLステートメントの構造や、潜在的な実行のパフォーマンスを理解することができます。これには、次の情報が含まれます。

- 照会を処理する操作の順序
- コスト情報
- 述部および選択可能性の見積もり
- Explain 機能の実行時点の、SQLステートメントで参照されている全オブジェクトに関する統計

この情報は次のような場合に役に立ちます。

- 照会用に選択した実行プランを理解する
- アプリケーション・プログラムの設計を援助する
- アプリケーションを再バインドするべき時期を判別する
- データベース設計を援助する

(執筆者注)

SQL Explain機能の具体的な解説はカンタン!DB2テクテク第1歩の

第3回「Explain アクセス・プラン編<前編>」(PDF形式/580KB)

第4回「Explain アクセス・プラン編<後編>」(PDF形式/532KB)

に掲載されていますので、参考にしてください。

一般書籍では、DB2 ユニバーサル・データベース for Linuxの第9章「パフォーマンス・チューニングとデータベースのモニター」でも上記の各モニターの使い方が紹介されていますので、こちらも合わせて参考にしてください。

### 3. WebSphereのパフォーマンスチューニング

WebSphereのチューニングには以下の4種類あります。

アプリケーションのパフォーマンスをチューニング  
 WebSphere リソースの自動再ロードの緩和  
 WebSphere システム・キューのチューニング  
 WebSphere システム・パラメーターのチューニング

このうち、WebSphere リソースの自動再ロードの緩和、WebSphere システム・キューのチューニング、WebSphere システム・パラメーターのチューニングの概略を示します( WebSphere Application Server V3.5アドバンスド版 InfoCenter WebSphere パフォーマンス・チューニング・ガイドより抜粋)。

#### ステップ1 自動再ロードの緩和

次のものの自動再ロードを緩和もしくは使用不可にします。  
 サブレット・エンジン、JSP ファイル、Web サーバー構成  
 (執筆者注)

「緩和」とは、自動再ロードを行う時間間隔を長くすることをいいます。サブレットの場合、WebSphereの管理コンソールで初期パラメータminTimeBetweenStatに100000などの大きな値を設定することで、自動再ロードを行う時間間隔を長くすることが可能です。

「使用不可にする」とは、自動再ロードを行わないようにすることをいい、Webアプリケーションの「拡張」タブをクリックした画面にある「自動再ロード」を「偽」に設定することで、自動再ロードを行わないようにすることができます。

#### ステップ2 WebSphere システム・キュー

1. 実稼働環境で使用するWebSphereシステム・キューを識別します。
2. 再現可能な代表的ワークロードを識別する
3. スループット曲線を作成します。
4. 飽和点を見つけ、キューを調整します。

(執筆者注)

調整できるキューは、以下の通りです。

- Web サーバー IBM HTTP: MaxClients (Unix の場合)、および ThreadsPerChild (NT の場合)
- WebSphere サブレット・エンジン: Max Connections
- EJB コンテナのスレッド・プール・サイズ
- WebSphere データ・ソースの接続プール・サイズ
- 準備済みステートメント・キャッシュ・サイズ

#### ステップ3 Javaメモリーのチューニング

1. アプリケーションでどのようにメモリーを使用するかを理解します。  
 ワークロードを固定し、Java ヒープ・パラメーターを変えながら実験を繰り返します。
2. アプリケーションでメモリー・リークが起こっているかどうか判断します。

ヒープ・パラメーターを変えずにワークロードを増やしながら実験を繰り返します。

### 3. ヒープ・パラメーターを調整します。

使用状況およびリークに関する実験結果を使用して、最も良いヒープ設定値を決定します。

(執筆者注)

ヒープ・パラメータとは、JavaVMが使用するヒープメモリのサイズを設定するためのパラメータをいい、WebSphereの管理コンソールから、Default Serverの「コマンドライン行引き数」に例えば

-Xmx128m -Xms64m -Xoss819200 -Xss819200 -Xnoclassgc

というように設定してアプリケーションサーバーを起動します。

## 4. アプリケーションのパフォーマンスチューニング

ここまではDB2やWebSphereのソフトウェアに対してのパフォーマンスチューニングについてご紹介してきましたが、アプリケーションの側にもパフォーマンスチューニングを施す方法があります。具体的な項目を以下の表に掲げます。なお、重要度は1が最も高く、以下2~5の順です。(一般技術情報「パフォーマンス・アップの決め手！WebSphereベスト・プラクティス」のWasdevbp.pdfより)

番号	カテゴリー	概要	重要度
1	Servlet	HttpSessionに保存するオブジェクト・サイズを小さくする	2
2	Servlet	終了時にHttpSessionを消去する	3
3	JSP	JSPでデフォルトで使用されるHttpSessionに注意する	4
4	Servlet	Servlet内での逐次化を最小限に留める	2
5	Servlet	SingleThreadModeを使用しない	5
6	All	JDBCコネクション・プーリングを使用する	3
7	All	JDBC接続のデータソースを再利用する	1
8	All	使用後にJDBCのリソースを解放する	3
9	Servlet	1度だけ実行すればよい処理はHttpServletのinitメソッドで行う	4
10	Servlet,EJB	System.out.printlnの使用は最小限にする	2
11	Servlet,EJB	Stringの連結処理で“+=”の使用を避ける	1
12	EJB	セッション・ビーンを使用してエンティティ・ビーンにアクセスする	3
13	EJB	EJB Homeの再利用を行う	1
14	EJB	適切な個所に“Read-Only”メソッドを使用する	3
15	EJB	適切なTransaction Isolationレベルを選択する	5
16	EJB	同一JVMでEJBとServletが稼働する時は“No Local Copies”を使用	2
17	EJB	使用後はステートフル・セッション・ビーンを削除する	2
18	Servlet,EJB	SQLは可能な限りPreparedStatementを使用する	

更に具体的なプログラミングの方法については、リンク元のPDF形式ファイルWasdevbp.pdfを参考にしてください。

この連載も今回で最後です。DB2 Developer Domainがオープンした2001年3月30日以来13週という長期間に渡ってお付き合いいただいた方々に感謝申し上げます。この記事がWebアプリケーションへの理解に少しでもお役に立てるものでありましたら、これほど幸いなことはありません。また別の機会にお会いできることを楽しみにしています。