

SQL Server 性能問題の3大要因 システムリソース/クエリ/待機を検証

日本ユニシス株式会社 森嶋 荘一郎 MORISHIMA, Shoichiro

パート2では、SQL Server の性能問題における原因とその調査/分析方法を解説する。SQL Server には「SQL Server Profiler」や「動的管理ビュー」、「パフォーマンスモニタ」などグラフィカルな調査ツールが標準で搭載されているため、性能問題を調査/分析しやすい。適切なチューニングを行なううえで調査/分析は特に重要となるため、しっかり把握しておきたい。

SQL Server の パフォーマンスチューニング

DBA ならば一度は経験したことがあるデータベースの性能問題。「システム本番中に急にクエリが遅くなった」「負荷テストをしたら全般的にクエリが遅い」「新たに開発したクエリが遅い」など、性能問題はさまざまな場面で発生する問題であろう。このような性能問題に対して必要となるのが「パフォーマンスチューニング」である。し

かし、特にまだ経験の少ないDBAにとって性能問題に直面しても「何を調査したら良いのか」「何が原因で性能が悪いのか」も分からず、どこから手を付けるべきか悩む人もいると思う。

DBMS の性能問題への取り組みには、大きく分けて次の3種類の工程がある。

- ① 設計工程で性能要件を意識した設計をする
- ② テストや本番開始後に、性能要件を満たさない事象に対して性能を測定し改善する
- ③ 性能悪化を未然に防止するための運用設計を実施する

一般的に、パフォーマンスチューニングとは②の工程を指すことが多い。つまり、システムに求められる性能要件(クエリの応答時間、スループットなど)を満たすために改善/調整することと言えよう。

なお、本パートにおけるSQL Server のパフォーマンスチューニングは、**図1**に示すようなフローで実施する。

発生しているシステムの構成なども把握する。

2 性能目標値の設定

性能要件から妥当性のある性能目標値を設定する。やみくもに性能改善するのではなく、目標値(パフォーマンスチューニングの終了条件)を確認したうえで性能改善をすることが重要である。

3 調査と分析

4 問題の改善策の適用

5 性能目標値の確認

性能悪化の箇所を切り分けしたうえで、原因追究のために調査/分析をする(3 調査と分析)。そして改善策を施し(4 問題の改善策の適用)、性能目標値を確認し(5)、達成するまで3~5の手順を繰り返す。

SQL Server では、さまざまなシステムリソースにボトルネックが発生する。多数のクエリが同時実行している環境では各クエリがリソース競合を起こすこともある。したがって、パフォーマンスチューニングではこのような現象を的確に捉えて原因を究明するための調査/分析が非常に重要である。性能問題の原因が分かれば、改善策は自ずと立てられる場合が多い。

本パートでは、SQL Server での性能問題に対する調査/分析方法を中心に解説していく。

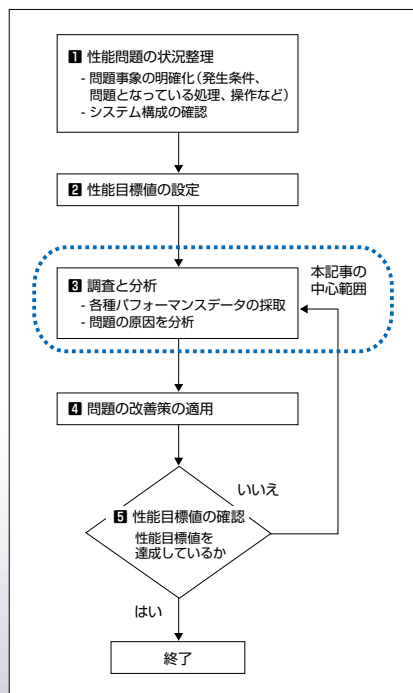


図1：パフォーマンスチューニングの流れ

1 性能問題の状況整理

性能問題は、主にユーザーから「処理が遅い」「画面をクリックしたけど反応がない」など“感覚的”な問い合わせや指摘から発生することが多くある。そのとき問題の発生条件や事象を整理し、明確化することが必要である。また、問題が

SQL Server 性能問題の 3 大要因 システムリソース / クエリ / 待機を検証

表 1：原因別の調査ツールと利用方法

原因	調査ツール	利用方法
システムリソースのボトルネック	パフォーマンスモニタ	データベースサーバーのシステムリソースの利用状況、SQL Server の稼働状況を分析する
非効率なクエリ	SQL Server Profiler	実行された各クエリの実行状態を把握し、問題となっているクエリ、改善対象とするクエリを抽出する
	動的管理ビュー	プロシージャキャッシュ内の各クエリの実行状態を取得し、問題となっているクエリ、改善対象とするクエリを抽出する
	(Management Studio)*	個別クエリの実行状態の把握、実行プランの表示
待機の事象	動的管理ビュー	待機の種類ごとの待機時間傾向を分析する。また、各クエリのロック待ちリソースを取得する
	パフォーマンスモニタ	インスタンス全体でのロックの数やロック待ちの時間の傾向を把握する
	SQL Server Profiler	各クエリのロックの取得状況やデッドロックの情報を確認する

*Management Studio は各種パフォーマンスデータを採取できないため、本記事では調査ツールとして位置付けていない。

性能問題が発生する 主な原因

ここでは、DBMS で発生する性能問題の原因について解説する。まず、性能問題が発生する主要な原因は次の 3 種類に絞ることができる（ほかにも性能問題の原因はあるが、イベントログや DBMS のエラーログなどから原因を追うことができる場合が多い）。

- システムリソースのボトルネック
- 非効率なクエリ
- 待機の事象

システムリソースの ボトルネック

SQL Server の性能問題で注目する主なシステムリソースを挙げるとすれば、次の 3 つに絞られるだろう。

- プロセッサ
- メモリ
- ディスク

プロセッサの処理できる要求数、メモリに確保できるデータ量、ディスクに読み込み／書き込みできる単位時間あたりのデータ量には限界がある。そのため、大量の処理要求がある場合、限界を超えたリソース部分で実施している処理が遅延し、結果としてクエリのスループットは低下する。これを「ボトルネック」と言う。

DBMS では、次の状況でシステムリソースのボ

トルネックが発生する。

- 同時実行ユーザー数の増加
- 1 つのクエリが多くのシステムリソースを消費
- ハードウェアのキャパシティ不足

ボトルネックの解決策（チューニング手法）は、必然的に「リソース使用量を節約する」、または「システムリソースを広げる」ということになる。前者はインデックスの設定によりアクセスするデータの範囲を絞るなど、リソース使用量を減らして性能向上を図ることになる。後者は、ハードウェアの追加やグレードアップである。

非効率なクエリ

クエリはデータベースエンジンにより統計情報を基にコンパイルされ、最適な実行プランが生成される。クエリはこの実行プランに基づいて実行される。

非効率なクエリの原因として、大きく次の 2 つの状況が考えられる。

- 条件やインデックスの不足などのために、ディスク I/O の負荷が高いクエリやソートなどのプロセッサリソースを多く使うクエリを実行している
- 統計情報が古いなどの理由により、実際のデータの分布状況と乖離しているため最適な実行プランが得られない

待機の事象

クエリは、トランザクションの一貫性を保つた

め、リソースにロックをかけることにより排他制御をしながら実行される。OLTP 環境のように複数のクエリが同時実行している環境では、この排他制御により片方のクエリが待たされ、結果としてクエリが遅くなることがある（ブロッキング）。また、複数のプロセッサで並列処理されているクエリは同期が完了するまで待たされることがある。このように、ブロッキングや各種同期処理などで待たされる事象を「待機」と呼ぶ。

性能問題の原因を 調査する

SQL Server には、性能の監視やチューニングを行なうためのツールが標準機能で提供されている。性能問題の原因を追究するには、これらのツールの特徴や使い方などをマスターし、場面に応じて必要な情報を取得する。

性能問題の調査ツール

SQL Server の性能問題を調査するためのツールとして、次の 3 つがある。

- パフォーマンスモニタ
- SQL Server Profiler
- 動的管理ビュー

表 1 に、性能問題の原因別に使用すべきツールとその利用方法をまとめた。性能問題の原因を調査するのに万能なツールはないため、問題発生状況に応じて使い分ける必要がある。

パフォーマンスモニタ

「パフォーマンスモニタ」はWindows OSで標準提供されているツールで、プロセッサやメモリ、ディスクなどの利用状況を表示し、ログとして保存するツールである。SQL Serverを導入することで、SQL Serverの稼動状況も採取できるようになり、どのシステムリソースでボトルネックが発生しているのかを確認できる。ただし、パフォーマンスモニタはボトルネックの直接の原因となっているクエリの特定まではできない。

パフォーマンスモニタには、採取するリソースの種類を表わす「オブジェクト」と、具体的なリソースを示す「カウンタ」がある(カウンタによってはさらに細かい採取単位である「インスタンス」も設定できる)。また、採取間隔も設定する必要がある。これは、問題の発生状況により適宜変更する。例えば、数分間の事象を把握したい場合は数秒から10数秒間隔で、1日全体のリソース利用状況を把握したい場合は数分間隔で設定する。

通常、各種システムリソースの利用状況を定期的にログとして採取／保存し、問題発生時に分析する。

SQL Server Profiler

「SQL Server Profiler (以下、Profiler)」は、SQL Serverで標準提供しているツールで、実行されたクエリに関する情報(実行ユーザー、ク

エリの実行時間、リソースの使用量など)やデッドロック発生時の詳細な情報をトレースとして取得できる。各クエリの実行状況が取得できるため、問題のあるクエリの抽出や分析に用いる。

Profilerは、クエリごとにさまざまな情報を取得できるが、DBサーバーに対する負荷が比較的高く、トレースのデータ量も大きくなるため、使用する場合は十分に注意する必要がある。Profilerを常時実行するのではなく、問題が発生する時間帯のみトレースを取得するという使い方を推奨する。

Profilerを用いて実行されたクエリをトレースとして取得するには、Profilerを起動後[ファイル]メニューの[新しいトレース]を選択して、次の設定をする必要がある。

●イベントと列の設定

採取する動作(イベント)とその情報(列)を指定する。採取するイベントと列が多ければ多いほどProfilerがシステムに与える負荷が大きくなるため、採取するイベントは最小限にする。

Profilerには、用途ごとに特定の種類のイベントをまとめたテンプレートが各種用意されている。問題の発生しているクエリを抽出するには、[全般]タブの[使用するテンプレート]で「Tuning」テンプレートを選択して[実行]ボタンをクリックし、**図2**のように「SP:StmtCompleted」「SQL:StmtCompleted」イベントを追加すれば、スタアドプロ

シージャや各クエリ、各ステートメントの実行時間やCPU時間などが取得できるため十分と考える。また、ほかにもさまざまな情報が取得できるため、適宜問題発生状況に応じてイベントを追加しても良い(図2を参照)。詳しいイベントの種類については、次のWebサイトを参照してほしい。

●SQL Server イベントクラスの参照

<http://technet.microsoft.com/ja-jp/library/ms175481.aspx>

●データ列を使用したイベントの説明

<http://technet.microsoft.com/ja-jp/library/ms190762.aspx>

●フィルタの適用

不要なトレースデータを採取しないようフィルタを適用する。[イベントの選択]タブの[列フィルタ]を選択して、列に対する条件を指定することによって適宜フィルタリングを設定する(画面1)。

●トレースの出力先選択

トレース結果はファイル形式(.trc)、またはSQL Serverのテーブルに出力できる。SQL Serverのテーブルへ出力した場合、並べ替えや絞り込み操作など分析／調査の自由度が高い反面、トレース採取自体にサーバーへの負荷が高い。分析時に、トレース結果のファイルをSQL Serverのテーブルに取り込むことができるため、SQL Serverのトレースを取得する際は、ファイル形式への出力を推奨する(図3)。

動的管理ビュー (DMV)

「動的管理ビュー (DMV)」は、SQL Server

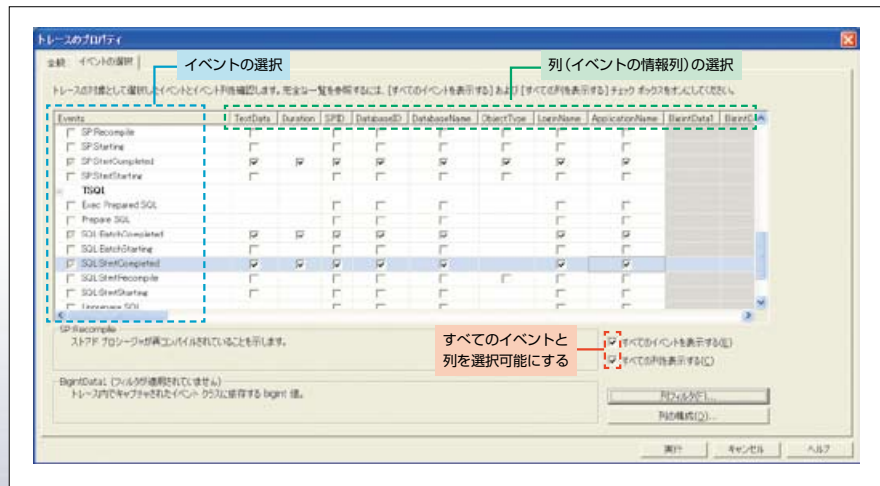
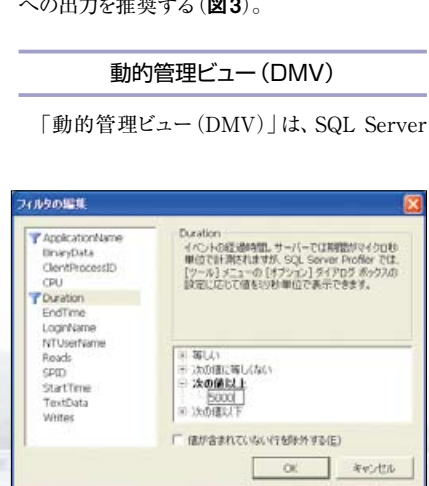


図2：採取するイベントと列の選択

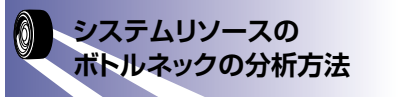


画面1：フィルタの適用

SQL Server 性能問題の 3 大要因
システムリソース / クエリ / 待機を検証

2005より新たに提供されたシステムビューである。クエリを実行することでSQL Serverの内部実行状態、メタデータや統計情報を参照できる。動的管理ビューはSQL Server 2008で約130種類用意されており、表2に示すような情報が取得できる。

動的管理ビューではデータ取得のための特別な設定は不要で、データ取得における負荷も少ないため、パフォーマンス監視や問題発生時の原因追究に使いやすいツールである。状況に応じて適切な動的管理ビューを選択して使用する。



ここでは、パフォーマンスモニタを使用して各システムリソースの利用状況を把握し、ボトルネックの有無を分析する方法を紹介しよう。具体的には、ボトルネックを特定するための主要な「パフォーマンスカウンタ」と、その値の評価方法について解説する。

ここでパフォーマンスカウンタの値を評価するうえで2点注意したい。1点目は、各パフォーマンスカウンタの適正值はシステムの状態により異なるものも多い。そこでお勧めしたいのが「ベースライン管理」である。これは、正常運用時のパフォーマンスデータである。事前に正常時のパフォーマンスカウンタ値（すなわちベースライン）を取得しておき、正常時の値と比較することでリソース使用量を評価できるわけだ。

2点目は、パフォーマンスカウンタの値は一時的に値が高くなることも多々あるが、瞬間的に値が高くなって継続的に値が高くなければ問題ではない場合が多いことである。パフォーマンスカウンタの値は継続的な値に注目しよう。

プロセッサ

ソート処理などでプロセッサを多く使用する、あるいは実行クエリが多い場合、プロセッサへの待

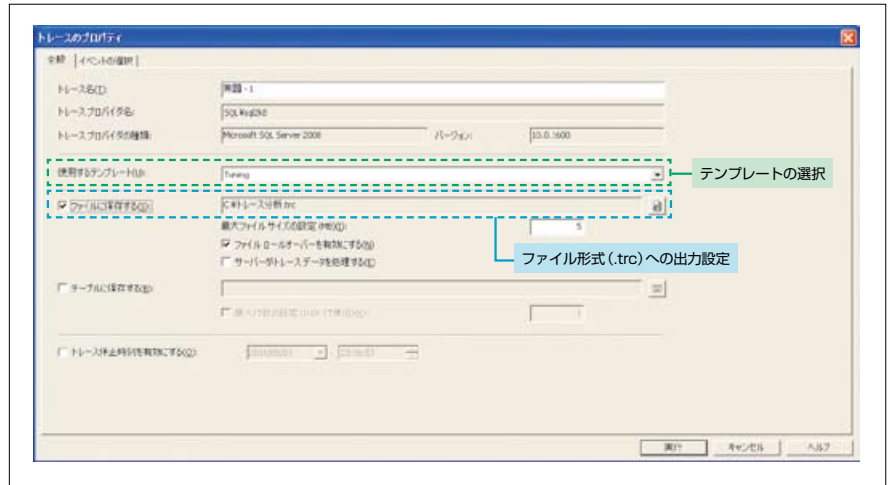


図3：テンプレート選択とトレース出力先選択

表2：動的管理ビューで取得可能な情報の例

監視対象	使用する動的管理ビュー	取得できる情報
実行中のクエリ	sys.dm_exec_requests	クエリ経過時間、CPU時間、ディスクの読み取り/書き込み情報
メモリ	sys.dm_os_buffer_descriptors	メモリアルール内のデータページの使用量
インデックスの断片化	sys.dm_db_index_physical_stats	インデックスの断片化率、断片化数
インデックスの利用状況	sys.dm_db_index_usage_stats	テーブル/インデックスの使用回数
tempdbの利用状況	sys.dm_db_db_filespace_usage	tempdbの空き領域
ロック競合	sys.dm_tran_locks sys.dm_os_waiting_tasks	ブロックを引き起こしているセッション、要求の状態
待機の事象	sys.dm_os_wait_stats	SQL Server 内部の待機の種類、時間、発生回数

表3：プロセッサリソースに関する主なパフォーマンスカウンタ

パフォーマンスカウンタ	説明	適正とする目安	用途・分析ポイント
Processor :	プロセッサの使用率	80%以下	プロセッサのボトルネックの有無を判断する
% Processor Time			
System :	プロセッサの待ち行列数	1プロセッサあたりの待ち行列数が2以下	プロセッサのボトルネックの有無を判断する
Processor Queue Length			
Processor :	特権モードで使用されるプロセッサ使用率	—	SQL Serverによる過度のディスクI/Oが発生している場合高くなる
% Privileged Time			
SQL Server : SQL Statistics :	1秒あたりのコンパイル数	—	1秒あたりのSQLコマンドのバッチ数(SQL Server : SQL Statistics : Batch Requests/sec)と比較してコンパイルの頻度を確認する
SQL Compilations/sec			

ち行列が発生する。このとき、クエリ実行に必要なプロセッサリソースが不足してクエリが遅くなる。

プロセッサリソースの分析に用いる主なパフォーマンスカウンタを表3に示す。

ボトルネックを判断するには、「Processor : % Processor Time」「System : Processor Queue Length」に注目する。SQL Serverのページ読み出し/書き込みが増加している場合にもプロセッサの使用率が上がるため、「Processor :

% Privileged Time」も確認する。また、コンパイル数が多い場合もプロセッサの使用率は高くなるため、「SQL Server : SQL Statistics : SQL Compilations/sec」にも注意する。

メモリ

メモリは、SQL Serverの性能にとって非常に重要なリソースである。SQL Serverに割り当てられているメモリが多ければ多いほどディスクI/

Oを減らすことができるため、SQL Serverにできる限りメモリを割り当てるのが性能向上のポイントになる。

表4に、メモリ関連のボトルネックを見分ける際に用いる主要なパフォーマンスカウンタを示す。

ディスク

DBシステムにおいて最も性能に影響を及ぼす可能性が高いリソースである。ディスクのボトルネックは、メモリが不足していることによって発生している場合がある。したがって、ディスク関連のボトルネック調査をする場合には、併せてメモリ関連の調査も行なう必要がある(表4の「SQL Server : Buffer Manager」オブジェクトのカウンタを中心に確認する)。

ディスクI/Oの処理をSQL Serverから見ると、主に次の処理がある。

- データページをSQL Serverのメモリ内に読み

込む。メモリ内のデータページをディスクに書き出す

- トランザクションログを書き込む
- tempdbへのI/O処理。ソートやインデックスの再構築、行のバージョン管理などで利用する

したがって、ディスクI/Oを調べるときは「どのデータベースファイルが配置されているドライブで問題となっているか」を把握して評価する必要がある。ユーザーデータベースのデータファイルが配置されているディスクにボトルネックがある場合は、ディスクI/Oの多いクエリを改善できないかを検討することになる。

表5に、ディスク関連のボトルネックを見分ける主要なパフォーマンスカウンタを示す。



非効率なクエリの分析方法

ここでは、問題となっているクエリを特定/抽

出し、個々のクエリを分析する方法を解説する。Profilerを用いて問題のあるクエリを抽出し、実行プランの詳細を把握することで非効率なクエリを分析できる。

Profilerを用いた問題のあるクエリの抽出方法

性能に問題のあるクエリを特定する方法として、採取したトレースファイルを分析する。

- トレースファイルのテーブルへの取り込み

ファイルに出力したトレースファイルをSQL Serverのテーブルへ取り込むには、Profilerでトレースファイルを開き、画面2のように保存先をテーブルにすればテーブルにインポートできる。

- 問題のあるクエリの特定

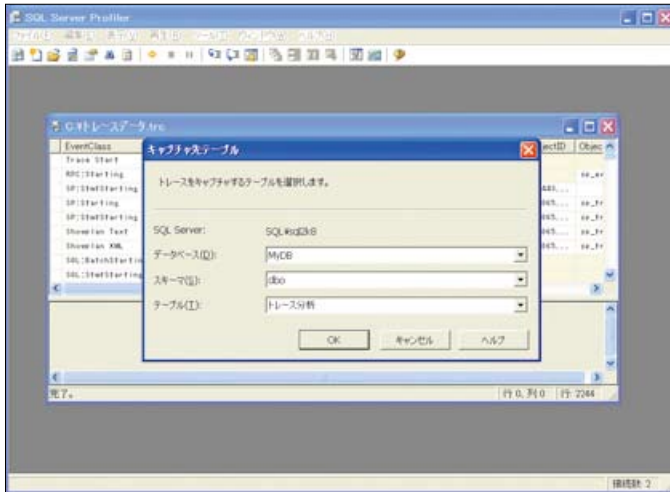
実行時間が長いクエリを抽出する場合、TSQ イベント(SQL : BatchCompleted、SQL : St

表4：メモリリソースに関する主なパフォーマンスカウンタ

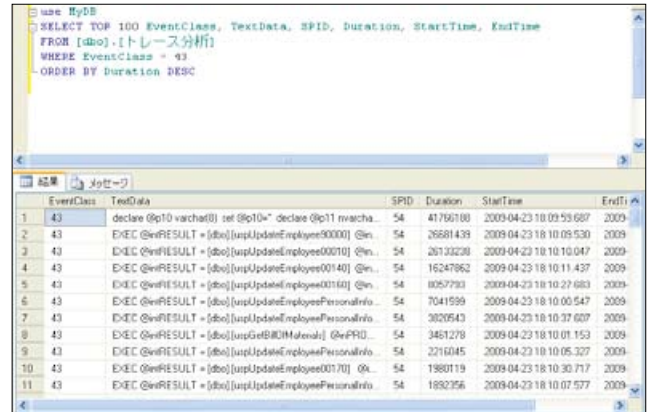
パフォーマンスカウンタ	説明	適正とする目安	用途・分析ポイント
Memory : Available Bytes	システム全体の利用できる空きメモリ	—	この値が少ない場合、ページングが多く発生する。[Memory : Available Bytes]と併せて確認する
Memory : Pages/sec	ページングにより物理メモリからディスクに書き込んだ回数とディスクから物理メモリに取り込んだ回数	—	この値が低い場合はディスクI/Oが少なく良好と言え、逆に多い場合は物理メモリの不足が考えられる
SQLServer : Buffer Manager : Buffer Cache Hit Ratio	クエリを処理するために必要なデータがSQL Serverで確保しているメモリ内で見つかった比率	90以上	OLTP環境では、この値が低い場合(90%以下)、SQL Serverで確保しているメモリの不足が考えられる
SQLServer : Buffer Manager : Page Reads/sec	SQL Serverがメモリ内にディスク上のデータページを読み込んだ回数	—	この値が多い場合、ディスクI/Oが多くSQL Serverで確保しているメモリアールの不足が考えられる
SQLServer : Buffer Manager : Page Writes/sec	SQL Serverがディスクにメモリ内のデータページを書き込んだ回数	—	この値が多い場合、ディスクI/Oが多くSQL Serverで確保しているメモリアールの不足が考えられる
SQLServer : Buffer Manager : Page Life Expectancy	SQL Serverで確保しているメモリ内でデータページが参照されていない場合に保持される秒数	300以上	メモリの不足を判断する。この値が高い場合、データページがメモリ内に滞留されている場合が多くなり、ディスクI/Oの削減が見込める

表5：ディスクに関する主なパフォーマンスカウンタ

パフォーマンスカウンタ	説明	適正とする目安	用途・分析ポイント
PhysicalDisk : Avg. Disk Queue Length	カウンタ採取間隔におけるディスクI/O要求の待ち行列長の平均	1ディスクあたり2以下	ディスクのボトルネックの有無を判断する
PhysicalDisk : Current Disk Queue Length	カウンタ採取時におけるディスクI/O要求の待ち行列長	1ディスクあたり2以下	ディスクのボトルネックの有無を判断する
PhysicalDisk : % Disk Time	ディスクに対するI/O要求を処理するのに要した時間の比率	50%以下	ディスクのボトルネックの有無を判断する
PhysicalDisk : Disk Read Bytes/sec	1秒あたりの物理ドライブごとのデータ書き込みバイト数	—	ディスクI/Oの傾向を把握する。平常時の値と比較する
PhysicalDisk : Disk Write Bytes/sec	1秒あたりの物理ドライブごとのデータ読み込みバイト数	—	ディスクI/Oの傾向を把握する。平常時の値と比較する

SQL Server 性能問題の 3 大要因
システムリソース / クエリ / 待機を検証

画面 2 : トレースファイルのテーブルへのインポート



画面 3 : 実行時間の長いストアドプロシージャの抽出例

mtCompleted) やストアドプロシージャイベント (SP : Completed, SP : StmtCompleted) に記録された Duration 列 (経過時間) で実行時間を確認する。実行時間がかかっている行に記録された TextData 列で、そのクエリを確認できる。また、ボトルネックの箇所に応じて CPU 列 (使用した CPU 時間)、Reads 列 (ページ読み取り I/O 数)、Writes 列 (ページ書き込み I/O 数) を確認する。

各パラメータを確認後、問題となるクエリの上位いくつかをピックアップして改善が可能かを検討する。また Profiler ではデータベースに対するクエリをすべて採取できるため、実行回数の多いクエリを特定して改善することが可能かを検討することも有効である。

画面 3 は、実行時間の長いストアドプロシージャの取得例だ。トレースデータを EventClass 列 43 (Stored Procedures : SP - Completed イベント) で条件付けてトレースデータを取得していることに注意してほしい。

各イベントの EventClass の値は、次の Web サイトを参照してほしい。

- SQL Server イベントクラスの参照

<http://technet.microsoft.com/ja-jp/library/ms175481.aspx>

PI COLUMN

オプションを活用したより詳細なクエリ分析

Management Studio のクエリエディタを利用してクエリの分析を行なう際は、実際のクエリ実行ができる場合は実行プランの表示と併せてオプション [STATISTICS IO] と [STATISTICS TIME] を有効化して実行することを推奨する。この 2 つのオプションを有効にすることで、実行時に出力される情報に次の内容が追加される (図 A)。

- SET STATISTICS IO

ステートメントが生成するディスク操作量に関する情報。テーブルやインデックスのスキャン回数、論理読み取りページ数、物理読み取りページ数、先行読み取りページ数など

- SET STATISTICS TIME

クエリ処理の経過時間と使用された CPU 時間。SQL Server の構文解析、コンパイルの時間、実行時間 (CPU 時間、経過時間)

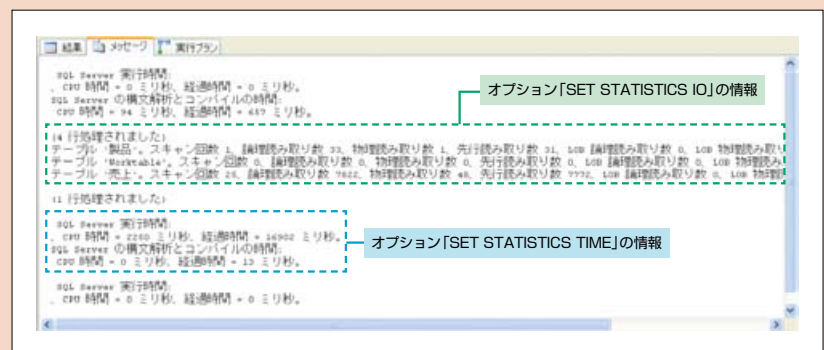


図 A : 実行時の情報の出力例

実行プランの分析ポイント

問題のあるクエリを特定した場合、次に実行プランの分析をする。実行プランは、次の方法で取得できる。

- Management Studio で取得する
- Profiler で Performance : Showplan Text / Showplan ALL / Showplan XML イベントを採取する

- 動的管理ビュー (sys.dm_exec_query_plan) を用いてプロシージャキャッシュ内の実行プランを取得する

ここでは、上記の中でも代表的な実行プランの表示方法である「Management Studio」を用いて実行プランを分析する方法を紹介しよう。

Management Studioで得られるクエリの実行プランには、「推定実行プラン」と「実際の実行プラン」の2種類がある。推定実行プランはクエリを実行せずに確認できるが、実際の実行プランはクエリを実行しないと確認できない。また、実際の実行プランは、結果セットの行数など実行時の情報も取得できる。状況に応じて使い分けをする。

Management Studioで実行プランを分析するには、クエリエディタに分析するクエリを入力

し、ツールバーの「推定実行プランの表示」ボタン、または「実際の実行プランを含める」ボタンをクリックする。生成された実行プランは、「実行プラン」タブで確認できる(図4)。

● 実行プランの見方

グラフィカルに表示された実行プランは、右から左、上から下に読む。右側に表示されたノードから先に実行される。各ノードに表示されるコストは、クエリの総コストに占める割合として表示される。この実行プランにより、結合方法や期待通りにインデックスが使用されているかなどを確認できる。また、ノード上にカーソルを置くと各種ノードでの詳細情報を確認できる(図4)。

実行プランのグラフィカル表示で出力される主なノードアイコンの例を表6に示す。そのほかのノードアイコンについては、次のWebサイトの情

報を参照してほしい。

- グラフィカルな実行プランのアイコン

<http://msdn.microsoft.com/ja-jp/library/ms175913.aspx>

分析方法は、実行プランの各ノードのコストに注目し、コストの高いノードに改善できる点がないかを確認する。特にテーブルスキャンなどディスクI/Oが多いノードがコストの多くを占めている傾向がある。このような場合、インデックスの追加によりコストの低減ができないかを検討してほしい。

待機事象の調査方法

問題の原因となっている待機の種類の特定を

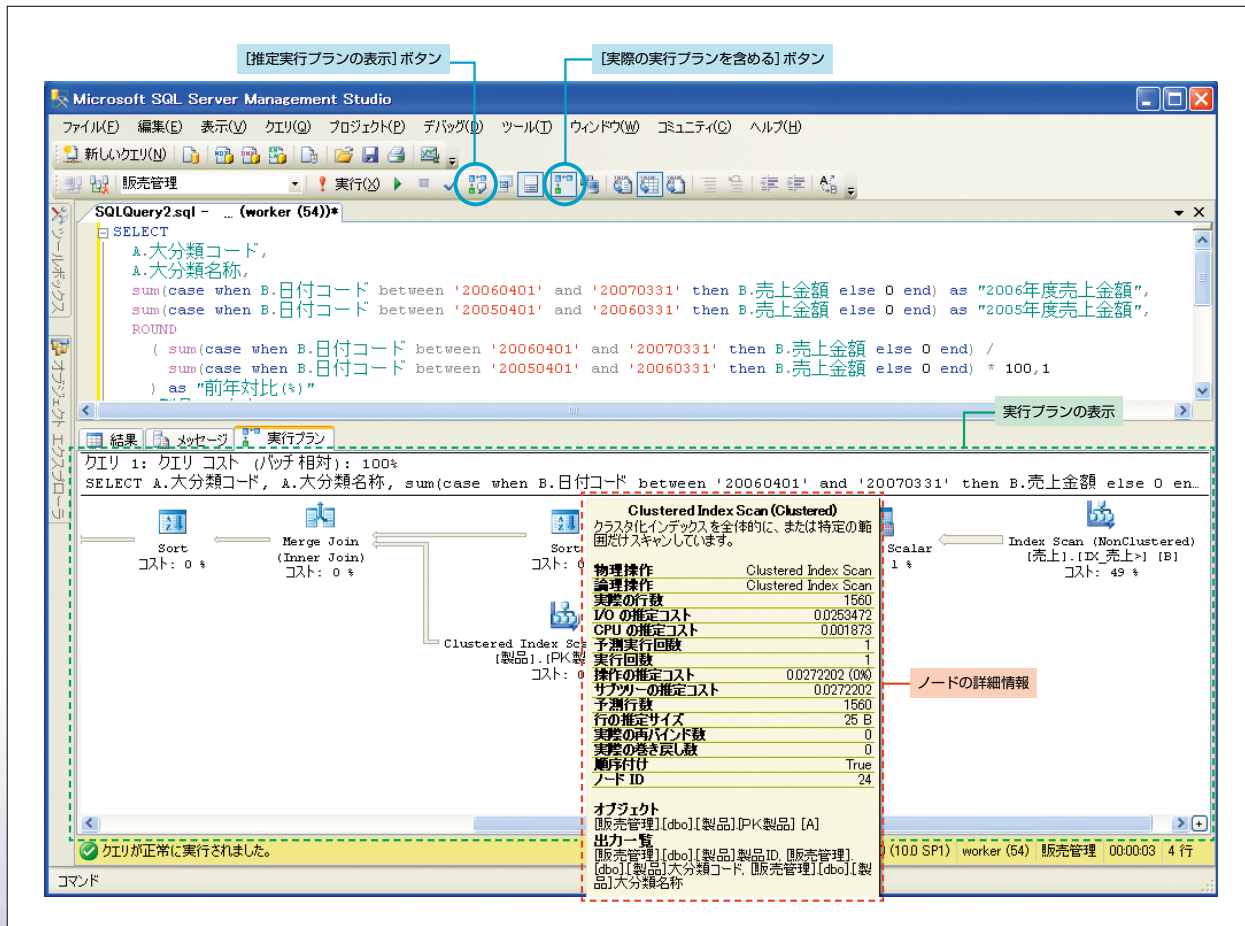


図4：実行プランの表示例

SQL Server 性能問題の 3 大要因
システムリソース / クエリ / 待機を検証

したうえで、それに応じた対応が必要となる。SQL Serverでの待機の事象は、大きく分けて次の2種類がある。

- ロックなどに代表されるマルチユーザーの処理による排他制御に伴う待機
- 並列処理におけるプロセス待機やトランザクションログの書き込み待機といった SQL Server 内部処理の動作に伴う待機

ここでは、待機の種類の特定方法と、特に発生しやすいロック待ちの調査方法を紹介する。

待機の種類の特定方法

SQL Serverは、インスタンスが稼動してからの待機の種類ごとの数、総時間などを内部に蓄積している。動的管理ビュー sys.dm_os_wait_statsを参照することで、これらの情報を確認できる。LIST1のクエリを実行すると、5秒間隔で待機の種類ごとの待機時間を取得できる。これにより、インスタンスレベルでどういう傾向の待機が発生しているかを確認できる。

詳しい待機の種類については、次のWebサイトを参考にしてほしい。

- sys.dm_os_wait_stats (Transact-SQL)

<http://msdn.microsoft.com/ja-jp/library/ms179984.aspx>





ロック待ちにより待機しているクエリとリソース要求状況の調査方法

待機しているクエリの情報を動的管理ビュー sys.dm_os_waiting_tasks/sys.dm_tran_locksを用いて取得する。待機しているクエリのSPIDや要求しているリソース情報と、それをブロックしているSPIDを把握できる。これらの動的管理ビューを問題の発生している時間帯に定期的に行うことで、ロックの競合情報を取得できる。

ロック待ち監視クエリ

LIST2、画面4は、ロック待ち情報を監視するクエリとその実行例である。このクエリを実行す

表6：実行プランのグラフィカル表示で出力される主なノードアイコンの例

	テーブルスキャン処理。テーブルからすべての行を読み取るため、非効率なプランと言える。インデックスのないテーブルで発生する
	インデックスシーク処理。インデックスを利用して対象となる行だけを読み取る。インデックスシークは効率の良いプランと言える
	インデックススキャン処理。インデックスのすべての行を読み取るため、効率的にはテーブルスキャンと変わらない。カーディナリティが低い場合、SQL Serverはシークではなくスキャンを選択することもある
	ソート処理。受け取ったすべての行を並べ替える。一般的に tempdb とプロセッサに負荷のかかる処理

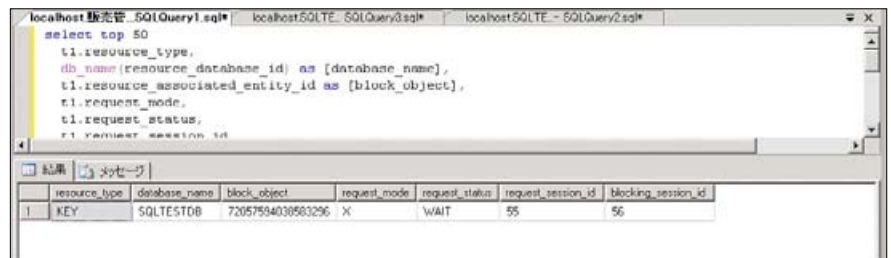
LIST1：

総待機時間 (wait_time_ms) の多い待機の種類を5秒間隔で抽出するクエリ例。1、8行目の「DBCC SQLPERF ('sys.dm_os_wait_stats',clear)」は sys.dm_os_wait_stats で取得できる各種蓄積値をクリアするコマンド

```
DBCC SQLPERF ('sys.dm_os_wait_stats',clear)
while 1=1
begin
select getdate()
use master
select * from sys.dm_os_wait_stats
order by wait_time_ms desc
DBCC SQLPERF ('sys.dm_os_wait_stats',clear)
waitfor delay '00:00:05'
end
```

LIST2：ロック待ち監視クエリ

```
select top 50
t1.resource_type,
db_name(resource_database_id) as [database_name],
t1.resource_associated_entity_id as [block_object],
t1.request_mode,
t1.request_status,
t1.request_session_id,
t2.blocking_session_id
from
sys.dm_tran_locks as t1,
sys.dm_os_waiting_tasks as t2
where
t1.lock_owner_address = t2.resource_address
```



resource_type	database_name	block_object	request_mode	request_status	request_session_id	blocking_session_id
KEY	SQLTESTDB	72057594030983296	X	WAIT	55	56

画面4：ロック待ち監視クエリ (LIST2) の実行結果

ることにより、ロック待ちをしているクエリのSPIDや要求しているリソースの情報、ロックを保持しているSPIDなどの情報が取得できる。画面4のクエリの実行例では、SPID“55”のクエリがロックにより待機していること、SPID“56”のクエリがブロックを引き起こしていることが分かる。

このようにロック待ちの状況を分析したうえで、ロックヒントやインデックスを設定してロックの範囲を狭める、実行の時刻をずらすなど、ロックの競合を防ぐような改善策を実施する。

* * *

以上、今回はSQL Serverの性能問題における原因とその調査／分析方法について解説した。SQL Serverには標準でグラフィカルな調査

ツールが用意されているため、調査／分析はしやすいのではないと思う。

誌面の関係上、解決策まではなかなか詳細に説明できなかったが、本稿が読者のSQL Serverのシステムで発生した性能問題の解決の一助になれば幸いである。

DBM

森嶋荘一郎 (もりしましろういちろう)
日本ユニシスへ入社後、主にアプリケーション開発畑を歩む。近年、SQL Serverの構築支援、技術支援を担当している。アプリケーションからSQL Serverまで、一気通貫したパフォーマンスチューニングを得意とする。