SYBASE®
An SAP® Company

パフォーマンス&チューニング・シリーズ：
モニタリング・テーブル

Adaptive Server® Enterprise

15.7

ドキュメント ID：DC01078-01-1570-01
改訂：2011 年 9 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

第 1 章

モニタリング・テーブルの概要

この章では、Adaptive Server® のモニタリング・テーブルに統計情報や診断情報を問い合わせる方法について説明します。

Adaptive Server のモニタリング・テーブル

Adaptive Server には、モニタリング情報および診断情報を格納したシステム・テーブルがあります。これらのシステム・テーブル内の情報では、Adaptive Server の状態に関する統計スナップショットが提供されます。その情報を基にサーバ分析を実行してサーバ・パフォーマンスを分析できます。たとえば、サーバ・プロセスとアプリケーションのアクティビティ、クエリ・パフォーマンス、データベース・テーブルの利用、データ・キャッシュの効率、データベース・デバイスの出入力アクティビティなど、システム・パフォーマンスに影響を及ぼす Adaptive Server の様々な側面に関する情報をレポートするための問い合わせができます。

モニタリング・テーブル内のデータは、ディスクには保存されません。データの計算は、モニタリング・テーブルに関するクエリの実行時に行われます。テーブル定義は、サーバ・インストール・スクリプトによって作成されたプロキシ・テーブル定義内に格納されています。これらのプロキシ・テーブルは、ユーザがクエリを実行すると、Adaptive Server へのインタフェースを使用してモニタリング・データを収集します。

モニタリング・テーブルは、サーバ・インストール・スクリプトを使用して作成する必要があります。「モニタリング・テーブルのインストール」(3 ページ) を参照してください。

Adaptive Server のバージョンによってモニタリング・テーブルの定義が変化する可能性があるため、アップグレード時には、モニタリング・テーブルを使用する前に適切なインストール・スクリプトを実行することをおすすめします。

注意 これらのテーブルのクエリには、mon_role が必要です。「mon_role と追加のアクセス制御」(11 ページ) を参照してください。

モニタリング・テーブルの情報源

Adaptive Server は、以下の場所からモニタリング・テーブル用の情報を収集します。

- グローバル・モニタ・カウンタ (行数に制限のある monSysWaits テーブルなど)。
- エンジン、プロシージャ、データ・キャッシュなどの単一サーバ・リソースに関連付けられているリソース固有のモニタ・カウンタ (結果ロー数が、プロシージャ・キャッシュ内にキャッシュされたコンパイル済みのオブジェクトのスナップショットによって異なる monCachedProcedures など)。
- アクティブなプロセス・ステータス構造体 (monProcessWaits や monProcessSQLText など)。monProcessWaits または monProcessSQLText の結果ロー数は、それぞれアクティブ・ユーザ接続数またはアクティブなプロセス・ステータス構造体数によって異なります。
- 循環バッファ (monSysStatement や monDeadLock など)。monSysStatement または monDeadLock の結果ロー数は、高速データ・パイプに格納するデータ量を設定する設定パラメータの値によって異なります。このバッファは、すべての履歴モニタリング・テーブルで使用されます。

大規模な運用サーバでは、モニタリング・テーブルの作成にリソースと時間がかかる場合があります。

Transact-SQL を使用したパフォーマンスのモニタリング

モニタリング情報がテーブルとして提供されるため、Transact-SQL を使用して Adaptive Server をモニタできます。たとえば、最大の CPU 時間を消費したプロセスまたは論理入出力を特定するには、以下を使用します。

```
select SPID, Login = suser_name(ServerUserID), CPUTime,
LogicalReads
from master..monProcessActivity
order by CPUTime desc
```

同じクエリを使用して、最大の物理入出力を使用しているプロセスを特定することもできます。この場合、CPUTime を PhysicalReads に変更します。

それぞれのモニタリング・テーブルの情報は、ソート、選択、ジョイン、他のテーブルへの挿入を実行でき、通常の Adaptive Server テーブルの情報とほぼ同様に扱うことができます。

モニタリング・テーブルは、クエリの実行ごとに生成されるメモリ内のテーブルであるため、読み取り専用であり、更新はできません。また、モニタリング・テーブル上にトリガを作成することはできません。

grant および revoke select などのアクセス制御コマンドを使用すると、モニタリング・テーブルへのアクセスを制限できます。

モニタリング・テーブルの定義は、コンポーネント統合サービス (CIS) プロキシ・テーブル機能を使用します。この機能は、Adaptive Server が、リモート・テーブルをローカル・テーブルとして扱えるよう定義できる機能です。

モニタリング・テーブルのインストール

Adaptive Server の 15.0.2 より古いバージョンでは、モニタリング・テーブルのインストール手順は、15.0.2 以降のバージョンの手順とは異なります。この項では、古いバージョンでのインストール手順について説明します。

Adaptive Server バージョン 15.0.2 以降および Cluster Edition のモニタリング・テーブルには、次の特性があります。

- *installmaster* スクリプトの実行時にインストールされる
- 実体化されたビューを使用する
- ループバック・サーバの作成を必要としない

15.0.2 より古いバージョン (Cluster Edition を除く)

$SYBASE/ASE-15_0/scripts ディレクトリ (Windows では *%SYBASE%\ASE-15_0\scripts*) にある *installmontables* スクリプトを使用してモニタリング・テーブルを作成します。

isql ユーティリティを使用して、*installmontables* スクリプトを実行します。次に例を示します。

```
isql -Usa -Ppassword -Sserver_name -i $SYBASE/ASE-15_0/scripts/installmontables
```

15.0.1 ESD #2 より古いバージョン用のループバック・プロキシ・サーバの設定

Adaptive Server のバージョン 15.0.1 ESD #2 以前のバージョンでは、*installmontables* スクリプトの実行時に「loopback」という名前のサーバが sysservers に存在する必要があります。このサーバを作成するには、次のように入力します。

```
declare @servernetname varchar(30)
select @servernetname=srvnetname
from master..sysservers
where srvname=@@servername
exec sp_addserver loopback, NULL, @servernetname
```

@@servername は NULL にできません。現在 NULL になっている場合は、sp_addserver を使用して「ローカル」サーバ名を定義します。サーバを再起動して *@@servername* に加えた変更内容を適用します。

モニタリング・テーブルへのリモート・アクセスと編集

バージョン 15.0.2 以降では、モニタリング・テーブルにリモートでアクセスする方法を示すサンプル・スクリプトとして *installmontables* が提供されています (Adaptive Server バージョン 15.0.2 以降でモニタリング・テーブルを作成するには、直接モニタされているサーバ上で *installmontables* スクリプトを実行する必要はありません)。*installmontables* を実行すると、編集手順が表示されます。次に例を示します。

```
isql -Usa -Psa_password -Sserver name -i $SYBASE/$SYBASE_ASE/scripts/ installmontables
---x---x-------------------------------------------------------------x---x---
It is no longer necessary to run this script to install the Monitoring Tables.Monitoring
Tables are now installed by the installmaster script.
This installmontables script is provided as a sample that can be copied and modified to
support remote access of Monitoring Tables. To do so you need to:
1) Replace all instances of @SERVER@ with the name of the remote ASE from which monitoring
data is to be obtained. Note that each remote ASE to be monitored must be added to the
local ASE's sysservers table using sp_addserver.
2) Create a database with the same name as the remote ASE. This database need only be of
the minimum size as these tables do not store any data.
3) Remove this header (i.e. these first 21 lines).
4) Run the script against the local ASE using the isql utility as follows:
    isql -Usa -P<password> -S<server name> -i<script name>
5) Retrieve remote monitoring data. E.g. to obtain monEngine information for an ASE named
REMASE you would execute the following SQL:
            use REMASE
            go
            select * from monEngine
            go
```

データ収集のためのモニタリング・テーブルの設定

デフォルトでは、Adaptive Server はモニタリング・テーブルに必要なモニタリング情報を収集しません。sp_configure を使用して、Adaptive Server がモニタリング情報の収集を開始するように設定します。次のリストのように、さまざまな領域でモニタリング・データの収集を制御する多数の設定オプションがあります。

モニタリング・テーブルの多くは、ユーザが 1 つ以上の設定オプションを有効にして、Adaptive Server によるデータ収集を開始するように設定する必要があります。テーブルによって異なるオプションが必要となります。表 1-1 に、各モニタリング・テーブルで必要な設定オプションを示します。

Adaptive Server が一般的なモニタリング情報を収集するようを設定するには、次の手順に従います。

1　デフォルトでは、Adaptive Server の最初の設定時に enable cis 設定パラメータが有効になっています (値 1 に設定されています)。このパラメータが有効になっていることを確認します。

2　enable monitoring 設定パラメータによって、他のモニタリング・オプションを有効にするかどうかを決定します。enable monitoring は 1 に設定します。

```
sp_configure "enable monitoringg", 1
```

次のように入力すると、モニタリングに特有の設定パラメータの完全なリストが表示されます。

```
sp_configure Monitoring
```

モニタリング情報の収集を制御する設定パラメータは以下のとおりです。

- enable monitoring
- deadlock pipe active
- deadlock pipe max messages
- errorlog pipe active
- errorlog pipe max messages
- max SQL text monitored
- object lockwait timing
- per object statistics active
- plan text pipe active
- process wait events
- sql text pipe active
- sql text pipe max messages

- statement pipe active
- statement pipe max messages
- statement statistics active
- SQL batch capture
- wait event timing

注意 設定パラメータの詳細については、『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

パイプ・エラー・メッセージへのメモリ割り当て

多くのモニタリング・テーブルが、メモリ内バッファ (「パイプ」と呼ばれる) を使用してモニタリング・データを収集します。以下のパラメータは各パイプに割り当てるメモリ量を制御します。

- deadlock pipe max messages
- errorlog pipe max messages
- sql text pipe max messages
- plan text pipe max messages
- statement pipe max messages

Adaptive Server は、パイプへのメモリ追加は動的に実行できますが、パイプからのメモリ削除は動的にはできません。したがって、パイプ・パラメータのサイズを縮小する場合には、Adaptive Server を再起動して新しいパイプサイズが適用されるようにする必要があります。

これらのパラメータのサイズを決定するために、以下のようなアルゴリズムが存在します。

- 個別の Adaptive Server で、各パイプ設定に必要なメモリは次のとおりです。

  設定値 x エンジンの数

- クラスタ環境では、各クラスタ・インスタンスによってモニタリング・テーブル・パイプの作成に必要なメモリが割り当てられます。詳細については、「クラスタ環境でのモニタリング・テーブルの使用」(15 ページ) を参照してください。

モニタリング・テーブルの設定パラメータ

Adaptive Server では、設定パラメータによって、モニタリング・テーブル用にどのデータを収集するかを制御します。モニタリング・テーブルの多くでは、ユーザが 1 つ以上の設定パラメータを有効にして、Adaptive Server によるデータ収集ができるように設定する必要があります。ユーザがモニタリング・テーブルをクエリできるようにするためには、mon_role を付与する必要があります。

モニタリング・テーブルを使用していない場合は、関連付けられた設定パラメータを無効にして、モニタリング・データの収集によって Adaptive Server にかかる負荷を軽減してください。

表 1-1 は、すべてのモニタリング・テーブル、およびそれらに適用される設定パラメータを示します。

表 *1-1:* 一部のモニタリング・テーブルに必須の設定パラメータ

	enable monitoring	SQL batch capture	deadlock pipe active	deadlock pipe max messages	errorlog pipe active	errorlog pipe max messages	object lockwait timing	per object statistics active	plan text pipe active	lock timeout pipe active	lock timeout pipe max messages	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement cache size	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored	enable stmt cache monitoring	capture compression statistics
monCachePool	X																						
monCachedObject																							
monCachedProcedures	X																		X				
monCachedStatement	X															X						X	
monCIPC																							
monCIPCEndpoints																							
monCIPCLinks																							
monCIPCMesh																							
monCLMObjectActivity	X																						
monClusterCacheManager																							
monCMSFailover																							
monDataCache	X																						
monDBRecovery																							
monDBRecoveryLRTypes																							
monDeadLock	X		X	X																			
monDeviceIO	X																						
monDeviceSpaceUsage																							

	enable monitoring	SQL batch capture	deadlock pipe active	deadlock pipe max messages	errorlog pipe active	errorlog pipe max messages	object lockwait timing	per object statistics active	plan text pipe active	lock timeout pipe active	lock timeout pipe max messages	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement cache size	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored	enable stmt cache monitoring	capture compression statistics
monEngine	X																						
monErrorLog	X				X	X																	
monFailoverRecovery																							
monInmemoryStorage																							
monIOController																							
monIOQueue	X																						
monLicense																							
monLocks	X																						
monLockTimeout	X									X	X												
monLogicalCluster																							
monLogicalClusterAction																							
monLogicalClusterInstance																							
monLogicalClusterRoute																							
monNetworkIO	X																						
monOpenDatabases	X																						
monOpenObjectActivity	X						X	X															
monOpenPartitionActivity	X							X															
monPCIBridge																							
monPCIEngine																							
monPCISlots																							
monPCM																							
monProcedureCache	X																						
monProcedureCacheMemoryUsage																							
monProcedureCacheModuleUsage																							
monProcess	X																			X			
monProcessActivity	X																			X			
monProcessLookup																							
monProcessMigration																							

	enable monitoring	SQL batch capture	deadlock pipe active	deadlock pipe max messages	errorlog pipe active	errorlog pipe max messages	object lockwait timing	per object statistics active	plan text pipe active	lock timeout pipe active	lock timeout pipe max messages	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement cache size	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored	enable stmt cache monitoring	capture compression statistics
monProcessNetIO	X																						
monProcessObject	X							X															
monProcessProcedures	X																		X				
monProcessSQLText	X	X																			X		
monProcessStatement	X																						
monProcessWaits	X												X							X			
monProcessWorkerThread	X																						
monRepLogActivity																							
monRepScanners																							
monRepScannersTotalTime																							
monRepSenders																							
monSQLRepActivity	X							X															
monSQLRepMisses	X							X															
monState																							
monStatementCache	X															X						X	
monSysLoad																							
monSysPlanText	X								X			X											
monSysSQLText	X	X												X	X						X		
monSysStatement	X							X									X	X	X				
monSysWaits	X																			X			
monSysWorkerThread	X																						
monTableColumns																							
monTableCompression	X							X															X
monTableParameters																							
monTables																							
monTableTransfer																							
monTask																							
monTempdbActivity	X						X	X															
monThread																							
monThreadPool																							

	enable monitoring	SQL batch capture	deadlock pipe active	deadlock pipe max messages	errorlog pipe active	errorlog pipe max messages	object lockwait timing	per object statistics active	plan text pipe active	lock timeout pipe active	lock timeout pipe max messages	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement cache size	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored	enable stmt cache monitoring	capture compression statistics
monWaitClassInfo																							
monWaitEventInfo																							
monWorkload																							
monWorkloadPreview																							
monWorkloadProfile																							
monWorkloadRaw																							
monWorkQueue																							

エラー 12036

モニタリング・テーブルを問い合わせる場合、テーブルで必要なすべての設定パラメータが有効になっていない場合には、クエリは実行されますが、Adaptive Server によってエラー 12036 が発行されます。すべての設定パラメータが有効になっていない場合でも、多くのモニタリング・テーブルには正確なデータが保存されています。ただし、Adaptive Server がテーブルのカラムに移植するためのデータを収集していないため、一部に不正確なデータがある場合もあります。

必要な設定パラメータを有効にすることを検討してください。詳細については、表 1-1 を参照してください。

mon_role と追加のアクセス制御

モニタリング・テーブルへのアクセスは、mon_role を持つユーザのみに制限されています。このロールを与えられているユーザのみが、モニタリング・テーブルでクエリを実行できます。特定のログイン、ロール、またはグループからモニタリング・テーブルに対する select パーミッションを付与または取り消すと、一部 (またはすべて) のモニタリング・テーブルへのアクセスをさらに厳密に制御することができます。ロールの取得については、『システム管理ガイド第 1 巻』の「第 11 章 ユーザ・パーミッションの管理」を参照してください。

一部のモニタリング・テーブルには機密情報が含まれている場合があります。たとえば、monSysSQLText テーブルおよび monProcessSQLText テーブルには、Adaptive Server に送付されたすべての SQL テキストが格納されています。このテキストに従業員の給与レコードの更新のような情報が含まれている可能性があります。管理者は、特定のロールのあるユーザのみにアクセスを制限するなど、これらのテーブルへの追加のアクセス制限を行って、システムのセキュリティ要件を満たす必要があります。

注意　クラスタ環境でモニタリング・テーブルを使用している場合、Workload および LogicalCluster モニタリング・テーブルでは mon_role は不要です。

カウンタ・データ型の循環

モニタリング・テーブルの一部のカラムには、Adaptive Server の稼働中に増分する整数のカウンタ値があります。カウンタは最大値 (2,147,483,647) に達すると、0 にリセットされます。これを「循環」といいます。

モニタリング・テーブル内のカラムの値の中には、循環が実行されたため、サーバが起動してからの総累計値を反映していない値が存在する可能性があります。このカラム・データを効果的に使用するには、一定の時間が経過するたびにカウンタ値の差分を計算し、このサンプルの結果を累計値の代わりに使用します。たとえば、現在値を使用する代わりに、現在値と 10 分前の値の差分を使用します。

異なるカウンタの値は、異なる速度で増大する傾向があります。たとえば、処理量が多いシステムでは、monDataCache テーブルにある LogicalReads カラムの値は急速に増加します。monTableColumns を使用すると、循環する可能性のあるカウンタを特定できます。monTableColumns.Indicators の値が 1 または 3 のカラムは循環する可能性があることを示します。サーバの動作は、負荷とアプリケーション・アクティビティによって変化します。また、Indicator カラムには、一般的なガイドラインが示されます。このため、循環する可能性のあるカウンタを特定するためにサーバのデータをレビューする必要があります。

カウンタであるカラムのリストを表示するには、次のクエリを実行します。

```
select TableName, ColumnName
from master..monTableColumns
where (Indicators & 1) = 1
```

ステートフルな履歴モニタリング・テーブル

多くのモニタリング・テーブルでは、現在のステータスに関する情報ではなく、個別のイベントのレコードが提供されます。これらのテーブルでレポートされているイベントによって、一定の期間におけるサーバの履歴レコードが提供されているため、これらのテーブルは、「履歴」と呼ばれます。Adaptive Server は、履歴テーブルにアクセスする各クライアント接続のコンテキスト情報を維持し、テーブルに連続してクエリを発行すると、そのクライアントが前回受信していないローだけを返します。こうした履歴モニタリング・テーブルの「ステートフル」な特性は、パフォーマンスを最大化し、履歴データの退避・保管を実施する際にローの重複を回避できるよう、設計されています。

履歴モニタリング・テーブルには、次のようなテーブルがあります。

- monErrorLog
- monDeadLock
- monSysStatement
- monSysSQLText
- monSysPlanText

注意 monSysPlan Text および monSysSQLText では、カラム BatchID、ContextID、ProcedureID、PlanID の値が Adaptive Server バージョン 15.0.3 以降からは変更されました。これらのカラムの変更内容については、『リファレンス・マニュアル：テーブル』を参照してください。

次のように、monTables.Indicators カラムで履歴モニタリング・テーブルを識別できます。

```
select TableName
from master..monTables
where Indicators & 1=1
```

履歴テーブルから返される情報は、履歴モニタリング・テーブルごとのバッファに格納されます。これらのバッファのサイズは、設定パラメータで指定され、データが保存される期間に影響を及ぼします。バッファのサイズや取得する情報を設定するには、sp_configure のオプションを使用します。使用する sp_configure オプションは、設定するモニタリング・テーブルによって異なります。たとえば、monSysPlanText テーブルでは、以下のような設定ができます。

- plan text pipe max messages － 特定のバッファに保存されるメッセージ数の指定
- plan text pipe active － Adaptive Server がそのバッファに対して情報を書き込むかどうかの指定

次の表に、履歴モニタリング・テーブルに影響を及ぼす設定パラメータのリストを示します。

モニタリング・テーブル	設定パラメータ
monErrorLog	errorlog pipe active errorlog pipe active messages
monDeadLock	deadlock pipe active deadlock pipe max messages
monSysStatement	statement pipe active statement pipe max messages
monSysSQLText	sql text pipe active sql text pipe max messages
monSysPlanText	plan text pipe active plan text pipe max messages
monProcessSQLText および monSysSQLText	max SQL text monitored

注意　履歴テーブルの中には、上記の表にある設定パラメータ以外の必須の設定パラメータを持つテーブルもあります。表 1-3 (17 ページ) を参照してください。

max messages パラメータは、エンジンごとの最大メッセージ数を指定します。この値に設定済みのエンジン数を掛けると、保存できるメッセージの総数がわかります。

モニタリング・テーブルには、保存されるメッセージごとに 1 つのローが追加されます。バッファ内のすべてのエントリを使用し終わると、新しいメッセージによって古いメッセージの上書きが行われます。したがって、最新のメッセージのみが返されます。

sp_configure の詳細については、『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」および「データ収集のためのモニタリング・テーブルの設定」(5 ページ) を参照してください。

Adaptive Server は前回の読み込みより後に追加されたデータのみを返します。したがって、where 句を使用して結果のフィルタを試みるクエリでは、以下の理由から、明らかに一貫性がない結果セットが返されます。

- モニタリング・テーブルの select は、テーブル内にある前回読み取られなかったすべてのメッセージを読み取り済みとしマークします。

- Adaptive Server 言語層によってフィルタが実行されます。したがってクエリの結果セットに含まれないローも、その接続によって「表示済み」と判断されます。

この例では、monErrorLog テーブルに関連付けられたバッファに次の 2 つのメッセージが含まれています。

```
select SPID, ErrorMessage
from master..monErrorLog
SPID      ErrorMessage
------    -------------------------------------
20        An error from SPID 20
21        An error from SPID 21

(2 rows affected)
```

ユーザが再接続すると、これらの 2 つのメッセージが返されますが、where 句を使用して結果セットのフィルタを実行すると、次のメッセージが表示されます。

```
select SPID, ErrorMessage
from master..monErrorLog
where SPID=20
SPID         ErrorMessage
------      -------------------------------------
20           An error from SPID 20
(1 row affected)
```

続けて、以下のクエリを実行します。

```
select SPID, ErrorMessage
from master..monErrorLog
where SPID=21
SPID         ErrorMessage
------      -------------------------------------
(0 rows affected)
```

最初のクエリによってクライアント接続のコンテキストが移動され、spids 20 および 21 の両方のローを含むようになります。このため、2 番目のクエリでは、これらの 2 つのローはいずれも返されなくなります。最初に実行されるクエリでは、サーバが両方のローを取得してから、フィルタに指定された条件を評価し、結果を返します。Adaptive Server は、spid 21 が、クライアント接続に返される結果セットに含まれていないにも関わらず、これを「読み取り済み」としてマークします。

注意 履歴モニタリング・テーブルにはステートフルな性質があるため、アドホック・クエリには使用しないでください。代わりに `select * into` または `insert into` を使用して、リポジトリまたはテンポラリ・テーブルにデータを保存し、その保存されたデータ上で分析を実行します。

一時モニタリング・データ

モニタリング・テーブルには、一時的なデータがあることが多いため、クエリでジョインしたり集計を使用したりするときには注意が必要です。テーブルに複数回問い合わせることが必要なプランでは、これらの操作の結果が異なる可能性があります。次に例を示します。

```
select s.SPID, s.CpuTime, s.LineNumber, t.SQLText
from master..monProcessStatement s, monProcessSQLText t
where s.SPID=t.SPID
and s.CpuTime = (select max(CpuTime) from master..monProcessStatement)
```

この例では、monProcessStatement を 2 回問い合わせます。最初の問い合わせで最大の CpuTime を特定し、最大値に一致させます。Adaptive Server による 2 回目のクエリの実行時に、monProcessStatement によって返される結果には、次の 3 とおりの可能性があります。

- 文がさらに処理を実行し、CPU をより多く消費し、CpuTime 値が前回の最大値を超えた場合。この場合、where 句に一致する結果がないため、クエリは結果を返さない。
- 2 回目の問い合わせが実行される前に文の実行が終了する場合。この場合、結果は返されない。ただし、別の文が前回取得された最大値とまったく同じ CPU 時間を使用している場合は、結果を返す。
- 文がそれ以上 CPU を使用せず、CpuTime 値が依然として最大値と一致する場合。このシナリオのみが、予期された結果を返す。

データ分析の前に、モニタリング・テーブルのデータをテンポラリ・テーブルまたはレポジトリに保存することをおすすめします。この操作によってデータを凍結することで、一時データ、または履歴モニタリング・テーブルのステートフルな性質に起因する望ましくない結果を回避できます。

クラスタ環境でのモニタリング・テーブルの使用

クラスタ環境でのモニタリング・テーブルの結果のレポートは、デフォルトでは、クラスタ全体での結果ではなく、インスタンスごとに (クラスタ内の 1 サーバごとに) 行われます。これによって、クラスタ全体でのプロセスのアクティビティやクエリを監視し、そのクラスタ内の複数のインスタンスで開かれた可能性のあるオブジェクトの統計や各インスタンスのリソース利用状況をよりよく理解できる可能性があります。たとえば、あるテーブルに関してモニタリング・テーブルでクエリを実行した場合、クエリの対象であるテーブルは、そのクラスタ内で複数のインスタンスによって開かれたり、アクセスされたりした可能性があるため、このテーブルの記述子と関連する統計は、そのインスタンスのメモリ内にある可能性があります。統計値は、クラスタ全体としての集計はされません。全インスタンスの統計結果は、各インスタンスから収集されたローを結合した結果セットとして返されます。各インスタンスのローは、結果セット内の、InstanceID カラムの値によって識別されます。

システム・ビューの設定

クラスタ・サーバでは、system_view は、セッション固有の設定です。この設定を使用すると、モニタリング・テーブル、sysprocesses、sp_who、およびその他のコマンドからクエリによって返されるモニタリング・データのスコープをユーザが制御できるようになります。system_view を cluster に設定すると、モニタリング・テーブルについてのクエリは、クラスタ内のアクティブな全インスタンスからデータを返します。system_view を instance に設定し、モニタリング・テーブルに対してクエリを実行すると、クライアントが接続しているインスタンス上でアクティブなプロセスまたはオブジェクトのデータのみが返されます。

set コマンドを使用して、セッションの範囲を設定します。

```
set system_view {instance | cluster | clear}
```

各要素の意味は次のとおりです。

- instance − ローカルなインスタンスのみに対して統計を返します。クラスタ間要求は、クラスタ内の他のインスタンスには送信されません。
- cluster − クラスタ内の全インスタンスについての統計を返します。
- clear − システム・ビューをデフォルトの状態に設定し直します。

モニタリング・テーブルに問い合わせるとき、またはモニタリング・テーブル RPC を呼び出すときに InstanceID を指定しないと、インスタンスは、現在の system_view 設定を使用します。

セッション・システム・ビューはホスト論理クラスタから継承されます。*@@system_view* グローバル変数を選択して、現在のシステム・ビューを指定します。

InstanceID のモニタ・インスタンスへの追加

表 1-2 は、Cluster Edition によって InstanceID カラムが追加されるモニタリング・テーブルを示します。

表 1-2: InstanceID カラムのあるモニタリング・テーブル

monCachePool	monDataCache
monCachedProcedures	monDeviceIO
monDeadLock	monErrorLog
monEngine	monIOQueue
monLicense	monLocks
monOpenDatabases	monNetworkIO
monOpenPartitionActivity	monOpenObjectActivity
monProcess	monProcedureCache
moProcessLookup	monProcessActivity
monProcessObject	monProcessNetIO
monProcessSQLText	monProcessProcedures
monProcessWaits	monProcessStatement
monResourceUsage	monProcessWorkerThread
monSysPlanText	monState
monSysStatement	monSysSQLText
monSysWorkerThread	monSysWaits
monCachedObject	

表 1-3 に、すべてのインスタンスに対してまったく同じ情報を返すモニタリング・テーブルを示します。

表 1-3: 全インスタンスについての同一の情報を含むモニタリング・テーブル

テーブル名	説明
monMon	メタデータ・ビューはすべてのインスタンスについて同一です。
monTableColumns	メタデータ・ビューはすべてのインスタンスについて同一です。
monTableParameters	メタデータ・ビューはすべてのインスタンスについて同一です。
monTables	メタデータ・ビューはすべてのインスタンスについて同一です。
monWaitClassInfo	説明のリストはすべてのインスタンスについて同一です。
monWaitEventInfo	説明のリストはすべてのインスタンスについて同一です。

ステートメント・キャッシュ用のモニタリング・テーブル

Adaptive Server のステートメント・キャッシュを有効にすると、update、delete、および select の各アドホック・コマンドに加え、再利用される可能性のあるそれ以外の文の SQL テキストが格納されます。ステートメント・キャッシュが有効になると、これらの文のクエリ・プランは再利用のために保存されます。新しい文が発行されると、再使用するプランが Adaptive Server によってステートメント・キャッシュで検索されます。Adaptive Server によって再利用するプランが検出されると、その文は再度コンパイルされる必要がないため、パフォーマンスの向上につながる可能性があります。

ステートメント・キャッシュの詳細については、『システム管理ガイド 第 2 巻』の「第 3 章 メモリの設定」を参照してください。

リテラルのパラメータ化によって、Adaptive Server は、where 句内のリテラル値のみ異なる、まったく同一のクエリを認識できます。リテラルのパラメータ化は、パフォーマンスを高めるだけでなく、測定基準や文をキャッシュに格納するときの領域を大幅に縮小するというメリットももたらします。

モニタリング・テーブルには、ステートメント・キャッシュのステータスおよびパフォーマンスの分析に使用される 2 つのテーブルが含まれています。monStatementCache には、ステートメント・キャッシュの概要スナップショットがあります。また、monCachedStatement には、キャッシュされた各文についての詳細な情報が表示されます。

ステートメント。キャッシュをモニタするための Adaptive Server の設定

enable stmt cache monitoring を使用すると、ステートメント・キャッシュに関するモニタリング情報を収集するように Adaptive Server を設定できます。

ステートメント・キャッシュからの文の削除

dbcc purgesqlcache を使用すると、ステートメント・キャッシュから文を削除できます。文 ID を指定すると、それに対応する文のみがキャッシュから削除されます。

dbcc purgesqlcache の構文は、次のとおりです。

```
dbcc purgesqlcache (int SSQLID)
```

SQL テキストからのハッシュ・キーの取得

ハッシュ・キーは、文のテキストに基づいて生成され、ステートメント・キャッシュの検索メカニズムに対する近似キーとしての役割を果たします。他のモニタリング・テーブルに文のテキストが表示されるため、ハッシュ・キーは、それらのテーブルで SQL テキストを検索および比較するための近似キーとして使用できます。

キャッシュされた文の SQL テキスト全体を表示する方法の詳細については、後述の「キャッシュされた文のテキストとパラメータに関する情報の表示」を参照してください。

Adaptive Server には、ハッシュ・キーを効果的に計算するために使用できる 2 つの関数が装備されています。ハッシュ・キーを計算する前に parse_text を使用して、SQL テキストの有効性を検証します。構文は次のとおりです。

```
select parse_text(text, prm_opt)
```

prm_opt の有効値は、次のとおりです。

- 1 － parse_text によって出力テキストが自動的にパラメータ化されることを示す。
- -1 － リテラルのパラメータ化の現在のセッション設定によって、入力テキストがパラメータ化されるかどうかが判定される。

SQL テキストが無効な場合、parse-text 関数は null を返します。

hashbytes を使用して、文のテキスト上でハッシュ・キーを計算します。次に例を示します。

```
select hashbytes('xor32', 'select * from syskeys')
```

キャッシュされた文のテキストとパラメータに関する情報の表示

show_cached_text を使用すると、キャッシュされた文の SQL テキストを表示できます。show_cached_text では、入力として文 ID を使用し、該当する文のテキストおよびパラメータ情報を表示します。構文は次のとおりです。

```
select show_cached_text(SSQLID)
```

show_cached_text を使用すると、クエリ内のステートメント・キャッシュにある文のテキストを取得できます。次に例を示します。

```
select SSQLID, show_cached_text(SSQLID)
from master..monCachedStatement
```

モニタリング・テーブルの問い合わせ例

この項では、モニタリング・テーブルのクエリ例を示します。

- 使用可能なすべてのモニタリング・テーブルのリストを返すには、次のクエリを実行します。

```
select TableName
from master..monTables
```

- 特定のモニタリング・テーブル内にあるカラムをリストするには、次のクエリを実行します。

```
select ColumnName, TypeName, Length, Description
from master..monTableColumns
where TableName="monProcessSQLText"
```

  where 句のテーブル名を置き換えてクエリを実行すれば、任意のモニタリング・テーブルのカラムを調べることができます。

- 現在実行中のクエリのうちどのクエリが一番 CPU を消費しているかを特定し、それらのクエリのテキストをリストするには、次のように入力します。

```
select s.SPID, s.CpuTime, t.LineNumber, t.SQLText
from master..monProcessStatement s, master..monProcessSQLText t
where s.SPID = t.SPID
order by s.CpuTime DESC
```

- Adaptive Server の稼働時間全体のプロシージャ・キャッシュのヒット率を調べるには、次のクエリを実行します。

```
select "Procedure Cache Hit Ratio" = (Requests-Loads)*100/Requests
from master..monProcedureCache
```

次のクエリでは、データ・キャッシュのヒット率も取得できます。この例では、ヒット率はサーバの稼働時間全体ではなく、10 分間隔で計算されます。

```
select * into #moncache_prev
from master..monDataCache
waitfor delay "00:10:00"
select * into #moncache_cur
from master..monDataCache
select p.CacheName,
"Hit Ratio"=((c.LogicalReads-p.LogicalReads) - (c.PhysicalReads -
p.PhysicalReads))*100 / (c.LogicalReads - p.LogicalReads)
from #moncache_prev p, #moncache_cur c
where p.CacheName = c.CacheName
```

特定のサンプル期間でパフォーマンス測定基準を計算するには、サンプル期間の開始時のモニタ値を保存するベースライン・テーブルを作成します。サンプル期間のモニタ値の変化は、サンプル期間の終了時の値からベースライン値を引いて計算します。

次の例のクエリを使用して、データ・キャッシュのヒット率の計算、ベースラインの作成、サンプル期間中のアクティビティの量の計算ができます。

- 実行済みの SQL、および現在ストアド・プロシージャを実行しているすべてのプロセスのバックトレースを出力するストアド・プロシージャを作成するには、次のように入力します。

```
create procedure sp_backtrace @spid int as
begin
select SQLText
from master..monProcessSQLText
where SPID=@spid
print "Stacktrace:"
select ContextID, DBName, OwnerName, ObjectName
from master..monProcessProcedures
where SPID=@spid
end
```

- dbid 5 およびオブジェクト ID 1424005073 で、データベース内のテーブルに対して使用されたインデックスを特定するには、次のように入力します。

```
select DBID, ObjectID, LastUsedDate, UsedCount
from master..monOpenObjectActivity
where dbid=5 and ObjectID=1424005073 and IndexID > 1
```

サーバで実行中のアプリケーションが使用しないインデックスを削除することができるかどうかを調べるには、次の手順に従います。

a　問題のテーブルにアクセスするアプリケーションにあるすべてのクエリを実行します。すべてのアプリケーションによってそれぞれ select が実行されるために必要な期間中、Adaptive Server が実行され続けていることを確認します。

b　アプリケーションがデータベース内のインデックスをまったく使用しなかったかどうかを調べるには、次のクエリを実行します。

```
select DB = convert(char(20), db_name()),
TableName = convert(char(20), object_name(i.id, db_id())),
IndexName = convert(char(20),i.name),
IndID = i.indid
from master..monOpenObjectActivity a,
sysindexes i
where a.ObjectID =* i.id
and a.IndexID =* i.indid
and (a.UsedCount = 0 or a.UsedCount is NULL)
and i.indid > 0
and i.id > 99 -- No system tables
order by 2, 4 asc
```

第 2 章

待機イベント

Adaptive Server タスク管理には、実行中、実行可能、スリープ中、およびブロック中というプロセスの状態があります。プロセスが実行中 (CPU 上で実行中) でない場合には、次の 3 つの状態があります。

- CPU 上で待機中 (「実行可能な」状態)
- ディスクまたはネットワーク I/O のためにスリープ中
- リソース上でブロック中 (ロック、セマフォ、スピンロックなど)

待機イベントは、サーバ・プロセスがプロセス自体によって中断され、別のイベントを待機する間スリープ状態になる場合に発生します。Adaptive Server には、これらの待機イベントごとにユニークな待機イベント ID があります。monSysWaits および monProcessWaits を問い合わせると各待機イベントでプロセスが待機した回数、および合計時間が返されます。

注意　monSysWaits テーブルの WaitTime の値は、秒単位です。monProcessWaits テーブルの WaitTime の値は、ミリ秒単位です。

この章では、一般的な待機イベントとそれらを回避するために行うアクションについて説明します。

イベント 19：xact coord：アイドル・ループ中の一時停止

Adaptive Server トランザクション・コーディネータ (ASTC) は、ウェイクアップするアラームまたはサーバ・タスクを待機中にスリープ状態になります (ASTC は複数のデータベース・サーバが関係するトランザクションを処理します)。サーバが多数の分散トランザクションを実行しない場合は、このイベントの time per wait は 60 秒近くになります。

対処法

アクションは不要です。WaitTime の値が大きな場合でも、イベント 19 は全般的なパフォーマンスに影響を及ぼしません。

イベント 29：通常バッファ読み込みの完了を待機

Adaptive Server がデータ・キャッシュ内でページを見つけられずディスクから読み込む必要がある場合に発生する物理読み込み (ほとんどの場合キャッシュ・ミス) を原因とする待機。Waits の数値は、キャッシュ・ミスによって発生した物理読み込みの回数を示します。monSysWaits.WaitTime 値を使用すると I/O 応答時間が導出できます。

対処法

monSysWaits.WaitTime のこのイベントの値は秒単位で測定されるため、このイベントの WaitTime の値は、Waits の値よりずっと小さな値である必要があります (平均的な物理的な読み込みは、2 ～ 6 ミリ秒である必要があります。10 ミリ秒を超えた場合、低速であると判断されます)。物理的な読み込みの平均値が高い場合は、ディスクのスループット・パフォーマンスが低下している可能性があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスクまたはオーバーロードしているディスクを特定できます。

Waits の値が高い場合、WaitTime の値に関係なく、クエリ・プランがその効果を発揮できないことを示している場合があります。Waits の値が高い場合は、不良統計、アクティブでない統計、または欠落している統計が原因で、テーブル・スキャンまたは直積が発生したか、オプティマイザが適切でないプランを選択した可能性があります。この現象が発生したテーブルの特定のカラム上にインデックスを追加することを検討してください。

Waits の値が高い場合、押し出された後再度読み込まれるアクティブなページがあるために、データ・キャッシュが小さすぎることを示している場合があります。monOpenObjectActivity、monProcessActivity、monDataCache、monCachPool、および monProcessObject を問い合わせて、次に進む方法を判断してください。

イベント 30：MASS の変更中の MASS の書き込みを待機

Adaptive Server がメモリ・アドレス領域セグメント (MASS) への書き込みを試行しています。MASS は、Adaptive Server がデータ・キャッシュ内で維持している 1 ページまたは複数の連続するページです。ただし、このイベントで MASS のステータスが「変更中」の場合は、別の spid が MASS を更新していることを意味します。書き込みを開始している spid は、MASS が使用状態でなくなるまでその MASS への書き込みはできません。

イベント 30 の WaitTime の高い値は、データ・キャッシュが小さすぎることを示している場合があります。この場合、データ・キャッシュ内のページが頻繁にウォッシュ・エリアに達するため checkpoint プロセスによって必要以上の書き込みが行われます。

対処法

次のアクションによって待機時間を削減できる場合があります。

- データ・キャッシュのサイズを増加する
- キャッシュ・パーティションまたは名前付きキャッシュを使用してメモリ集約的なオブジェクトを分離する
- ハウスキーピング、ウォッシュ・マーカ位置、スキーマの影響 (シーケンシャル・キー・テーブルなど) を調整する
- ウォッシュ・マーカを配置する
- スキーマ (シーケンシャル・キー・テーブルなど) を調整する

イベント 31：書き込み前にバッファ書き込みを待機

チェックポイントなどのデータ・ページのディスクへの書き込みを行うサーバ・プロセスによって MASS の書き込みの必要性が検出された場合で、その同じページが関係する以前の write オペレーションが終了していない場合、最初の write オペレーションが完了するまで、その write オペレーションの開始を待機する必要があります。

対処法

一般的に、イベント 31 の WaitTime 値は、Waits の値より低い必要があります。WaitTime の高い値は、ディスク競合またはパフォーマンスの低下を示している場合があります。monIOQueue および monDeviceIO を問い合わせると、オーバーロードしているディスクや低速なディスクを特定できます。

イベント 31 WaitTime の高い値は、データ・キャッシュが小さすぎることを示している場合もあります。この場合、データ・キャッシュ内のページが頻繁にウォッシュ・エリアに達するため checkpoint プロセスによって必要以上の書き込みが行われます。

イベント 32：APF バッファ読み込みの完了を待機

Adaptive Server がページ上で非同期プリフェッチ (APF) を発行したときに別のプロセスがこのページが属している MASS を読み込んでいる場合、Adaptive Server は、その読み込みの完了を待機する必要があります。

対処法

Waits の高い値は、Adaptive Server による頻繁すぎる非同期プリフェッチの実行を示している可能性があります。キャッシュ・プール用のローカル APF の制限を調整すると、APF ページに対する競合を低減できる場合があります。

Adaptive Server はテーブル・スキャン用に APF を頻繁に使用します。このため、APF 読み込みに関連する競合が、欠落しているインデックスなどを要因とする、アプリケーションによる頻繁すぎるテーブル・スキャンの実行を示している場合もありま。

イベント 35：バッファ検証完了を待機

あるプロセスが別のプロセスによってキャッシュに読み込まれたページ内のデータの読み込みを試行していることを示します。データ・キャッシュへの読み込み後、Adaptive Server によって read オペレーションが問題なく完了したかどうかが検証されます。Adaptive Server が read の実行が成功したかどうかを検証している間は、次のプロセスによるそのデータへのアクセスは待機する必要があります。

このイベントは、通常、物理的な読み込みが大量に実行される期間に発生します。

対処法

イベント 35 の WaitTime 値は、非常に低い値である必要があります。この値が高い場合、多数のプロセスが同じページに同時にアクセスしているか、CPU 競合が存在します。monEngine に問い合わせると、エンジンがオーバーロードしているかどうかを調べられます。また、システムレベルのユーティリティを実行すると、全体的な CPU 競合があるかどうかを調べられます。

イベント 36：変更前に MASS の書き込み完了を待機

spid が MASS に変更を加える必要がある場合に別の spid がその MASS に書き込みを行っている場合は、2 番目の spid は最初の書き込みが完了するまで待機する必要があります。

対処法

WaitTime の高い値は、I/O またはデータ・キャッシュ・マネージャのパフォーマンスの低下の原因となる条件があることを示しています。通常、Waits の値は、WaitTime の値より高い値である必要があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスや、オーバーロードしているディスク・デバイスを特定できます。

注意　ページの更新によってイベント 36 が発生した場合、キャッシュのパーティション分割を行っても効果はありません。ただし、更新が行われていないときにイベント 36 が発生した場合、キャッシュのパーティション分割を行うと書き込みが効率よく実行される場合があります。

イベント 37：変更前に MASS の変更完了を待機

spid が MASS への変更を試行している場合に別の spid がその MASS の変更を行っている場合は、最初の spid による変更が完了するまで次の spid による変更は待機する必要があります。

対処法

通常の場合、イベント 37 の Waits の値は、WaitTime の値よりずっと高い値である必要があります。Waits の値が高くない場合には、多数のプロセスが同じ MASS に同時にアクセスしているか、CPU の競合が存在します。monEngine を問い合わせると、エンジンがオーバーロードしているかどうかを調べられます。また、システムレベルのユーティリティを実行すると、全体的な CPU の競合が存在するかどうかを調べられます。

イベント 41：ラッチ取得の待機

イベント 41 は、複数のプロセスが 1 つのページ上のローの更新を同時に試行していることを示している場合がしばしばあります。

Adaptive Server は、プロセスによってデータの読み込みや書き込みが行われている間に別のプロセスによってそのページのコンテンツが変更されないことを保証するための一時的なロックとしてラッチを使用します。Adaptive Server は、通常、データのみがロックされたテーブルでラッチを使用して、複数プロセスが同じページ上でローの読み込みや更新を同時に実行している場合のページ・コンテンツの保護を行います。すでにラッチを保持しているプロセスがある場合に別のプロセスがラッチの取得を試みた場合、そのプロセスは待機する必要があります。イベント 41 は頻繁に発生する場合は、インデックスまたはテーブル内にある 1 つの物理的なページ上で高いレベルのデータ競合が存在する可能性を示しています。

次の方法で競合を低減してください。

- 競合しているローが異なるページに分散されるソート順でインデックスを作成する
- アプリケーションを変更し、競合の発生を防止する

対処法

テーブル内のデータ・ページおよびインデックス・ページ全体でデータの物理的な分散を変更するようにインデックス定義を変更することによって競合を低減すること、またはアプリケーションを変更して競合を低減することを検討してください。

WaitTime の平均値が高い場合は、次のような要因による Adaptive Server のリソース不足が原因でイベント 41 が発生している可能性があります。

- Adaptive Server が非常に長いハッシュ・チェーンを検索する結果となる、ロックに対して小さすぎるハッシュ・テーブル。
- すばやく実行されるはずの呼び出しがボトルネックを発生するオペレーティング・システムの問題 (たとえば、すぐに返されるはずの非同期 I/O がオペレーティング・システムのリソース制限によってブロックされるような場合)。
- 極端に高い挿入および拡張更新。ページ割り当てが頻繁に発生し、割り当てページ・ラッチの競合によって Waits の高い値が発生する。dbcc tune(des_greedyalloc) を使用するとこの競合を低減できる。ラッチ競合の詳細については、『パフォーマンス＆チューニング・シリーズ：sp_sysmon による Adaptive Server の監視』を参照してください。

イベント 46：バッファ書き込みによる LRU からのバッファ取得完了を待機

spid によって、最も長い間使用されていない (LRU) チェーンからのバッファの取得が試行されています。Adaptive Server がそのバッファを別のページで使用できるようになるには、そのバッファにある未処理の書き込みが終了する必要があります。

対処法

イベント 46 は、次のような状態を示します。

- キャッシュがビジーであるため、LRU チェーンの終端のバッファがまだ処理中です。monDataCache および monCachePool を問い合わせると、ビジー状態のキャッシュを特定できます。解消する手段としては、キャッシュのサイズの増加、ウォッシュ・サイズの増加のための sp_poolconfig の使用、enable housekeeper GC を返すことによるハウスキーピング・アクティビティの増加などが考えられます。
- ディスク書き込みの完了に長時間かかっています。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 51：MASS での最終 I/O の完了を待機

オブジェクトに変更が加えられたこと、またはオブジェクトがメタデータ・キャッシュから削除されたことによって、プロセスがオブジェクトの一定範囲のページをディスクに書き込んでいる場合に発生します。新たなページへの書き込みが行われる前にページ上での I/O オペレーションが完了していることが重要です。このため、プロセスは開始済みの I/O の終了が通知されるまでそのタスクを待機する必要があります。

対処法

WaitTime の高い値は、書き込みの完了までに長い時間かかっていることを示します。通常、Waits の値は、WaitTime の値よりずっと高い値である必要があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 52：別のタスクによって開始された MASS 上での I/O を待機

オブジェクトに変更が加えられたこと、またはオブジェクトがメタデータ・キャッシュから削除されたことによって、プロセスによってオブジェクトの一定範囲のページのディスクへの書き込みが試行されていますが、別の spid に MASS 上で未処理の I/O があるため、このプロセスは未処理の I/O のあるプロセスの書き込みが終了するまでスリープ状態になります。

対処法

このイベントの WaitTime の値が高い場合、書き込みの完了にかかっている時間が長すぎることを示します。通常、Waits の値は、WaitTime の値よりずっと高い値である必要があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 53：I/O 開始前に MASS による変更の完了を待機

spid による MASS への書き込みを試行したときに別の spid による MASS への変更が実行中であった場合、書き込みを試行した spid は、すでに開始されている変更が終了するまで待機する必要があります。

Adaptive Server は、実行するディスク I/O オペレーションの数を最小化します。ページ書き込みを行うプロセス (チェックポイント・プロセスなど) がページの変更を行う必要がある場合に別のプロセスがそのページの変更を行っていることが検出されると、ページの書き込みにそのページへの変更が含まれるように 2 番目のプロセスは最初のプロセスが終了するまで待機します。

対処法

通常の場合、イベント 53 の Waits の値は、WaitTime の値より高い値である必要があります。高くない場合には、多数のプロセスが同じ MASS に同時にアクセスを行っているか、CPU の競合が存在します。monEngine を問い合わせると、エンジンがオーバーロードしているかどうかを調べられます。また、システムレベルのユーティリティを実行すると、全体的な CPU の競合が存在するかどうかを調べられます。

イベント 54：最終ログ・ページの書き込み完了を待機

イベント 54 は、プロセスが最終ログ・ページの書き込みを開始しようするときに、別のプロセスによる write の実行がすでにスケジュールされていることを検出したした場合に発生します。2 番目のプロセスは、最初のプロセスの I/O が終了するまで I/O オペレーションを開始せずに待機します。

Adaptive Server は、トランザクション・ログの最終ページを頻繁に更新するため、Adaptive Server は最終ログ・ページの物理的な書き込みの実行を回避します。これによって、サーバが実行する I/O の量が減少し、更新を実行する必要がある他のプロセスに対する最終ログ・ページの可用性が増加するため、パフォーマンスが向上します。

対処法

イベント 54 の WaitTime の平均値が高い場合、書き込みに長い時間がかかっていることを示します。通常、Waits の値は、WaitTime の値よりずっと高い値である必要があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

Waits の値が高い場合は、平均時間に関係なく、最終ログ・ページでの競合を示している可能性があります。ユーザ・ログ・キャッシュのサイズを増加して競合を低減するか、アプリケーションのオペレーションをグループ化してローごとのコミットを回避してください。

イベント 55：最終ログ・ページを書き込み後の I/O 終了を待機

プロセスが、トランザクション・ログの最終ページ上で write オペレーションを開始し、I/O が完了するまでスリープ状態になることを示します。イベント 55 の Waits カラムの値が高い場合、トランザクションをディスクに書き込む必要があるコミット済みのトランザクションまたはその他のオペレーションのために Adaptive Server がトランザクション・ログに大規模な更新を行っていることを示します。

対処法

イベント 55 WaitTime の値が高い場合、書き込みに長い時間かかっている可能性があることを示します。通常、Waits の値は、WaitTime の値よりずっと高い値である必要があります。

イベント 57：チェックポイント・プロセス・アイドル・ループ

チェックポイント・プロセスは、チェックポイントが CPU 時間を占有しないように実行と実行の間にスリープします。

対処法

イベント 57 には、サーバ起動時にチェックポイント・プロセスが開始されてから長い時間が蓄積される可能性があります。ただし、このイベントに基づきアクションを実行する必要はありません。

イベント 61：hk：一時停止

ハウスキーピング機能が CPU 時間を占有しないように、ハウスキーピングの一時停止が発生することがあります。

対処法

イベント 61 は、長時間実行されたサーバ上で発生することが予想され、多数発生する可能性があります。通常は、このイベントに基づきアクションを実行する必要はありません。

イベント 70：デバイス・セマフォの待機

Adaptive Server のミラーリングを使用している場合 (つまり disable disk mirroring が 0 に設定されている場合)、各ディスク・デバイス・アクセスはそのデバイスのセマフォを事前に保持している必要があります。イベント 70 は、セマフォを待機した時間を測定し、ディスク I/O 構造が低すぎる場合に発生します。

対処法

Adaptive Server のミラーリング機能を使用していない場合、disable disk mirroring を 1 に設定します。ミラーリング機能を使用している場合に WaitTime の値が高い場合は、デバイス競合によるパフォーマンスの低下を示している場合があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。結果を評価し、ロードの一部を他のデバイスに移すことを検討してください。

イベント 83：DES 状態の変化を待機

オブジェクト記述子 (“DES” と呼ばれる) は、オープンなオブジェクトごとに割り当てられています。オープンなオブジェクトには、テンポラリ・テーブル、キャッシュ・クエリ・プランとステートメント・キャッシュ、ストアド・プロシージャ、トリガ、デフォルト、ルール、テーブルなどがあります。イベント 83 は、Adaptive Server が割り当て済みの記述子を解放したときに発生します。この解放は、通常、Adaptive Server がオブジェクトをドロップしたときに発生します。

対処法

イベント 83 の Waits の高い値は、オブジェクト記述子の不足を示している場合があります。オープンなオブジェクトの数を増加する必要がある場合もあります。

イベント 84：チェックポイントの完了を待機

Adaptive Server が DES をドロップしています。このドロップは、通常、Adaptive Server がオブジェクトをドロップするときに発生します。イベント 84 は、このドロップの動作がデータベース上でチェックポイントが完了するまで待機する必要があることを示します。

対処法

イベント 84 の Waits 値が高くなる可能性はほとんどありませんが、高くなった場合、多数のドロップが同時に発生している可能性、またはチェックポイント・プロセスに長い時間がかかっていることを示します。チェックポイントの実行時間が長すぎる場合は、リカバリ間隔 (分単位) を減少してください。

イベント 85：充填済みの DFLPIECE へのフラッシャのキューを待機

Adaptive Server による dump database の実行時に、データ・キャッシュ内にある変更が加えられたページのリストで変更されたページのリスト (DFLPIECE と呼ばれる構造を持つ) を作成するために「フラッシャ」プロセスが使用されます。Adaptive Server はダンプ内にあるページのリストを Backup Server に送付します。

イベント 85 は、フラッシャ・プロセスが DFLPIECE の充填とキューを待機するためにダンプ・プロセスが待機する時間を測定します。

対処法

このイベントは、通常、dump database 中に発生します。WaitTime の平均値が予想外に高い場合 (2 より高い場合) は、フラッシャ・プロセスの速度が低下している原因を調べるために他のイベントをチェックしてください。

イベント 91：ディスク・バッファ・マネージャ I/O の完了を待機

Adaptive Server が load database を実行した場合に、実行を継続する前にロード・プロセスによってディスク I/O が完了していることを検証する必要がある場合もあります。イベント 91 は、Adaptive Server が検証を待機した時間を測定します。

対処法

一般的にイベント 91 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。WaitTime の高い値は、ディスクの競合または速度低下の可能性を示します。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 99：クライアントからのデータを待機

プロセスが、サイト・ハンドラを使用してリモート・サーバに接続に、サーバからデータが返されるのを待機しなければならないことがあります。イベント 99 は、このプロセスが待機する時間を測定します。

サイト・ハンドラは、ローカル・サーバからリモート・サーバへの RPC 転送のメソッドです。サイト・ハンドラは、RPC が必要とする、ローカル・サーバとリモート・サーバ間の 1 つの物理的な接続および複数の論理接続を確立します。

対処法

イベント 99 の WaitTime の高い平均値は、リモート・サーバとの通信の速度が低いことを示します。これは、複雑な RPC 呼び出しが完了するまでに長い時間がかかっている、リモート・サーバのパフォーマンスに問題がある、またはネットワークの速度が低下またはオーバーロードしていることが原因である可能性があります。

イベント 104：エンジンがオフラインになるのを待機

Adaptive Server には、継続的に実行されるエンジン・クリーンアップ・バックグラウンド・プロセスが含まれます。このサービスは、エンジンがオフラインになった後でクリーンアップ・タスクを実行します。このプロセスは、通常、30 秒ごとにスリープとウェイクアップを繰り返し、行う作業があるかどうかをチェックします。イベント 104 は、このプロセススリープ時間を測定します。

対処法

イベント 104 の WaitTime の平均値は、30 にかぎりなく近い値である必要があります。エンジンが頻繁にオフラインになる場合は、この値は若干低めになります。WaitTime の平均値が、30 を大きく上回ったり下回ったりする場合は、Sybase サポート・センタにご連絡ください。

イベント 124：ページ取得時の大量の読み込みの終了を待機

イベント 124 は、プロセスの物理的な読み込み試行時に、別のプロセスが読み込み要求の実行をすでに開始している場合に発生します (「キャッシュ・ミス」としてもカウントされます)。

対処法

イベント 124 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。ディスク•パフォーマンスが低い場合は、WaitTime の平均値が高くなります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク･デバイスやオーバーロードしているディスク･デバイスを特定できます。

イベント 142：論理接続の解放を待機

Adaptive Server がサイト・ハンドラ・メカニズムを使用して RPC をリモート・サーバ上で実行すると、論理接続が作成されます。

イベント 142 は、Adaptive Server が論理接続を切断する必要がある場合に別のプロセスがその接続を使用していることを検出した場合に発生します。Adaptive Server がその論理接続を切断するには、その論理接続の使用が終了するのを待つ必要があります。

対処法

イベント 142 は、通常、WaitTime の平均値に非常に低い値を取ります。WaitTime の高い値は、リモート・サーバとの通信に問題がある可能性を示します。

イベント 143：サイト・マネージャと同期をとるための一時停止

Adaptive Server がサイト・マネージャを使用してリモート・サーバとの通信を試みたときに、別のプロセスがそのリモート・サーバに接続を試みています。イベント 143 は、Adaptive Server がリモート・サーバへの接続を確立するまで待機する時間を測定します。

対処法

イベント 143 の WaitTime の高い平均値は、リモート・サーバにパフォーマンスの問題がある可能性、またはネットワークの速度が低下またはオーバーロードしている可能性を示します。WaitEventID 143 に対して monProcessWaits を問い合わせると、待機時間の値が高い spid を特定できます。

イベント 150：ロックを待機

オブジェクト上で論理ロックの取得を試行しているプロセスがある場合に、別のプロセスがこのオブジェクト上で競合するロックをすでに保持している場合に発生します。イベント 150 は、Adaptive Server がデータの読み込みや更新ができないように保護することを必要とするオペレーションを実行する場合によく発生するイベントです。関係するロックは、テーブル、ページ、ローなどさまざまなレベルのロックである可能性があります。

すべての競合するロックが解放されると、Adaptive Server は待機していたプロセスをウェイクアップし、そのオブジェクトへのアクセスを許可します。

対処法

特定のテーブルまたはページへの競合が存在する場合 (ヒープへの挿入数が多い場合など)、このイベントの WaitTime の値が高くなる可能性があります。monLocks および monOpenObjectActivity を問い合わせると、ロックの競合が多発しているオブジェクトを特定できます。

場合によっては、テーブルのロッキング・スキームを全ページロックからデータオンリーロックに変更することで、ロックの競合を減少できることもあります。ロック競合は、通常、アプリケーションまたはデータベースの設計が原因で発生します。このため、使用しているアプリケーション設計を評価して、それ以外のアプリケーション要件を満たしつつロック競合を低減するための最適な方法を検討する必要があります。

イベント 157：オブジェクトのプールへの返還を待機

Adaptive Server のメモリ・マネージャは、多種の内部オブジェクトに関する記述情報を格納するために、個別の「プール」からメモリを割り当てます。プールで使用可能なメモリが少ないと、別のオペレーションがそのプールにメモリを返すまで追加メモリの要求が遅れる場合があります。この遅延が発生した場合、要求プロセスは、追加のメモリが使用できるようになるまで待機する必要があります。

イベント 157 は、オブジェクトのデータを割り当てるためにメモリが使用可能になるのをプロセスが待機する必要がある場合に発生します。

対処法

イベント 157 の WaitTime の平均値が低い場合、パフォーマンスは著しくは低下しません。ただし、このイベントのすべての Waits は、Adaptive Server が待機する構造の設定数を増加することによって修正可能な条件であることを示しています。sp_countmetadata および sp_monitorconfig を使用して、どの構造が最大の設定を使用しているかを識別することで、どのリソースを増加する必要があるかを判断できます。

イベント 169：メッセージを待機

Adaptive Server プロセスの中には、ワーカー・スレッド、監査、ディスク・ミラーリングなど、メッセージの伝達に「メールボックス」と呼ばれる構造を使用するプロセスがあります。イベント 169 は、Adaptive Server がメールボックス内のメッセージを待機するために費やした時間を測定します。

対処法

通常、イベント 169 の WaitTime の平均値は、非常に小さい値です。WaitTime の値が大きい場合は、169 の WaitEventID 値を使用して、ローに対して monProcessWaits を問い合わせ、このイベント待機時間が長いジョブを特定します。

イベント 171：CTLIB イベントの完了を待機

Adaptive Server が、リモート・サーバの応答を待機していることを示します。イベント 171 は、プロキシ・テーブルおよび RPC 呼び出しでコンポーネント統合サービス (CIS) を使用している場合に発生します。

対処法

このイベントの WaitTime の高い平均値は、リモート CIS サーバのパフォーマンスに問題があるか、ネットワークの速度が低下またはオーバーロードしている可能性を示します。WaitEventID 171 に対して monProcessWaits を問い合わせて、このイベントの待機時間の値が高い spid を特定してください。

イベント 178：新しいクライアント・ソケットの割り当て中に待機

ネットワーク・リスナは、クライアントの受信接続要求を処理する Adaptive Server のプロセスです。イベント 178 は、Adaptive Server が新規接続要求を待機するために費やした時間を測定します。

対処法

イベント 178 に基づいてアクションを実行する必要はありません。ただし、このイベントの情報の一部を分析用に使用できます。WaitTime の値は、サーバが実行された時間とほぼ同等です。Waits の値は、サーバを起動してから接続試行が何度実行されたかを測定します。

イベント179：ネットワーク読み込みまたは書き込み不要期間中に待機

Adaptive Server ネットワーク・タスクは、サーバが送受信する必要があるネットワーク I/O がない場合、イベント 179 上でスリープします。ネットワーク・アクティビティがあると、サーバ・タスクがウェイクアップし、要求を処理した後再びスリープします。

対処法

イベント 179 の高い値は、ネットワーク・アクティビティのレベルが高いことを示します。ネットワーク・アクティビティが予想外に高い場合、monNetworkIO や monProcessNetIO などの他のモニタリング・テーブルを問い合わせて、ネットワーク・パフォーマンスを低下しているジョブを特定してください。

イベント 179 の Waits カラムの高い値は、dbcc checkstorage が多数の一貫性フォールトを識別したことを示している可能性があります。詳細については、dbcc checkstorage のレポートをチェックしてください。

イベント 197：並列 dbcc で読み込みの完了を待機

dbcc checkstorage の実行時に、予約済みバッファへの読み込みまたは書き込みのために Adaptive Server が作業領域上で非同期 I/O を実行する必要がある場合があります。イベント 197 は、Adaptive Server がそれらのディスク I/O を待機した時間を測定します。

対処法

通常、イベント 197 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。WaitTime の高い平均値は、ディスクのスループット・パフォーマンスの低下を示している可能性があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 200：並列 dbcc でページ読み込みを待機

イベント 200 は、複数のワーカー・プロセスを使用して dbcc checkstorage を実行すると発生します。このイベントは、dbcc がチェックするページ上で読み込みが完了するまでにかかった時間を測定します。

対処法

一般的にイベント 200 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。WaitTime の高い平均値は、ディスクのスループット・パフォーマンスの低下を示している可能性があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 201：並列 dbcc でのディスク読み込みを待機

dbcc checkverify の実行時に、Adaptive Server によってディスクの読み込みが実行され、ページのディスク・コピー内に潜在的なフォールトが存在するかどうかを検証します。イベント 201 は、これらの読み込みが完了までに待機した時間を測定します。

対処法

一般的にイベント 201 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。WaitTime の高い平均値は、ディスクのスループットの低下を示している可能性があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 202：並列 dbcc でのページ再読み込みを待機

dbcc checkstorage の実行時に、Adaptive Server によってディスクの読み込みを実行して、ページのディスク・コピー内の潜在的なフォールトの存在を検証する必要があるかどうかが検討されます。イベント 202 は、これらの読み込みが完了するまで待機した時間を測定します。

対処法

通常、イベント 202 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。WaitTime の高い平均値は、ディスクのスループットの低下を示している可能性があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 203：並列 dbcc の MASS_READING ビット上で待機

dbcc checkstorage の実行時に、Adaptive Server によってディスク読み込みを実行して MASS のディスク・コピー内のフォールトの存在を検証する必要があるかどうかが検討されます。ただし、別のプロセスが、すでにその読み取りを開始している可能性があります。イベント 203 は、これらの読み込みが完了するまでに待機した時間を測定します。

対処法

一般的にイベント 203 の WaitTime の値は、Waits の値よりずっと低い値である必要があります。WaitTime の高い平均値は、ディスクのスループットの低下を示している可能性があります。monIOQueue および monDeviceIO を問い合わせると、低速なディスク・デバイスやオーバーロードしているディスク・デバイスを特定できます。

イベント 205：並列 dbcc の TPT ロック上で待機

dbcc checkstorage を実行して text ページおよび image ページをチェックするときに、Adaptive Server は、同時に複数のワーカー・スレッドがそのページ・リンクにアクセスしないようにロックを保持する必要があります。イベント 205 は、これらのロックを待機するために費やした時間を測定します。

対処法

イベント 205 の発生頻度は、チェック対象のテーブルに含まれる text カラムと image カラムの数によって決まります。WaitTime の極端に高い平均値は、ロックを保持しているワーカー・スレッドに対するリソース競合を示している場合があります。CPU およびディスクの測定基準をチェックして、競合の有無を検証してください。

イベント 207：PLL dbcc 内の親へのフォールト・メッセージ送付を待機

dbcc checkstorage の実行時に、親 spid にメッセージをキューすることで、各ワーカー・プロセスによって潜在的なフォールトが親プロセスにレポートされます。親プロセスのメールボックスが満杯の場合は、ワーカー・プロセスは、メールボックスに空きができて新しいメッセージのキューができるようになるまで待機する必要があります。イベント 207 は、ワーカー・プロセスが待機に費やした時間を測定します。

対処法

イベント 207 は、通常、Adaptive Server が多数のフォールトをレポートすることによって発生します。このイベントに対しては、dbcc checkverify を実行してフォールトの検証と分析を行う通常のプロセス以外に、特別なアクションを実行する必要はありません。

イベント 209：パイプ・バッファの読み込みを待機

Adaptive Server による並列でのソート実行時には (create index で `consumers` 句を指定するなど)、内部メカニズムを使用して複数のタスク間でデータの送付が行われます。イベント 209 は、別のタスクによるパイプへのデータ追加をタスクが待機するために費やした時間を測定します。

対処法

イベント 209 の WaitTime の平均値は、非常に小さい値である必要があります。WaitTime の高い平均値は、ソート・マネージャ・プロデューサ・プロセスによるデータ生成速度が、コンシューマ・プロセスをビジーな状態に維持するために必要な速度に達していないことを示している可能性があります。全体的なシステム・パフォーマンスをチェックして、Adaptive Server に十分な CPU および I/O 帯域幅があるかどうかを検討してください。

イベント 210：パイプ・マネージャ内の空きバッファを待機

Adaptive Server による並列でのソート実行時には (create index で `consumers` 句を指定するなど)、パイプと呼ばれる内部メカニズムを使用して、複数のタスク間でデータの送付が行われます。イベント 210 は、プロセスが Adaptive Server によって空きパイプ・バッファが割り当てられるのを待機した時間を測定します。

対処法

イベント 210 の WaitTime の平均値は、非常に小さい値である必要があります。WaitTime の高い平均値は、Adaptive Server にリソース競合があることを示している可能性があります。sp_monitor または sp_sysmon を実行するか、monEngine を問い合わせて、Adaptive Server に十分な CPU リソースがあるかどうかを検討してください。

イベント 214：解放後に、実行キュー上で待機

イベント 214 は、プロセスが他のプロセスの実行ができるように解放を行った後、実行キュー上で待機した時間を測定します。このプロセスは、「実行可能」であり、ロック、物理 I/O、またはその他の待機条件によって待機しているプロセスではありません。このイベントは、不十分な CPU (つまり CPU 集約的なサーバ)、またはメモリ内のテーブル・スキャンが原因となって発生した可能性があります。

イベント 214 がイベント 215 と異なる点は、time slice によって割り当てられている CPU 時間を超過した CPU 集約的なタスクを実行中のプロセスに、自主的に CPU を解放し、Adaptive Server のスケジューラが追加の CPU 時間を割り当てるまで実行可能な状態で待機するように指示することです。追加の CPU 時間が割り当てられると、プロセスは、CPU を解放する前に実行していたアクティビティを再開します。

イベント 215 も、実行可能な状態にあるプロセスを示しますが、イベント 214 はプロセスが実行可能な状態に移行する理由が割り当て済みの CPU 時間の超過ではなく、タスクの実行に必要なディスクまたはネットワーク I/O や論理ロックなど別の理由があるために待機します。

対処法

ビジーなサーバは、通常、Waits の値が高くなります。ただし、WaitTime の高い値または time slice の高い値での設定は、Adaptive Server に実行を待機している spid が多数存在すること、または、実行中の極端にCPU 集約的な spid が使用している CPU を容易に解放しないことを示している可能性があります。monProcessActivity を問い合わせて、高い CPUTime 値のあるジョブを特定してください。

イベント 215：スリープ後に実行キューで待機

イベント 215 は、プロセスが別の待機イベント (論理ロック、ディスク I/O、その他の待機イベントなど) を待機する必要がなくなった場合に発生し、そのプロセスはサーバの実行可能キューに配置されます。このプロセスは、スケジューラによって CPU 時間が割り当てられるまでタスクの継続を待機します。

イベント 214 と 215 の差異については、イベント 214 の説明を参照してください。

対処法

イベント 215 は、一般的な待機イベントです。イベント 215 の Waits の値は、通常は大きな値となります。ビジーな状態にあるサーバでは、プロセスが Adaptive Server の実行可能キューを長時間待機するため、WaitTime の値が高くなります。追加のプロセスが CPU にアクセスできるようにするには、time slice の値を低く設定して、さらに数多くのプロセスが CPU にアクセスできるようにするか (この場合、プロセスの CPU 時間の平均も減少します)、ホスト・マシン上に十分な CPU が使用可能な場合にはオンライン・エンジンの数を増加してください。

イベント 222：フラッシュ中のレプリケーション・エージェントのスリープ

Adaptive Server がレプリケーションを実行するプライマリ・サーバの場合、RepAgent プロセスは実行する必要がある作業 (たとえば、ローがデータベースのログに追加される場合など) を待機する間スリープします。イベント 222 は、RepAgent がスリープしている時間を測定します。

対処法

レプリケーションされたデータベース内でのアクティビティのレベルによっては、イベント 222 の WaitTime の値が高くなる場合もあります。通常、このイベントに対してアクションを実行する必要はありません。

イベント 250：受信ネットワーク・データを待機

このイベントは、アプリケーション・プロセスが、クライアントからの次の要求を待機中にアクティブである時間を測定します (つまりジョブが AWAITING COMMAND 状態にある場合)。

イベント 250 は、通常、アプリケーションが Adaptive Server に接続されたままアイドル状態になったときに発生します。

対処法

イベント 250 は 、Adaptive Server がクライアントからの各コマンドを処理する前に発生するため、Waits および WaitTime の値は通常高くなります。

イベント 250 を使用すると、サーバが処理したクライアントからの要求数を見積もることができます。

このイベントの高い WaitTime 値は、アイドル状態にある多数のクライアント接続、または長期間にわたりアイドル状態にあるクライアント接続を示している可能性があります。この待機イベントは、クライアント・アプリケーションから送付されるバッチとバッチの間、コマンドとコマンドの間に発生する場合があります。このため、アプリケーションが多数の独立したコマンドまたはバッチを送信した場合、Waits 値が高くなる場合があります。

イベント 251：ネットワーク送信の完了を待機

イベント 251 は、クライアントへの返信パケット送信中にジョブが待機する時間を測定します。

対処法

イベント 251 は、Adaptive Server がクライアントに大きな応答パケットを送信していることを示す場合があります。また、ネットワークの速度低下やオーバーロードを示すこともあります。monNetworkIO テーブルおよび monProcessNetIO テーブルで平均のパケット・サイズをチェックしてください。これらのテーブルの平均サイズは、次のように求めます。

(BytesSent) / (PacketsSent)

クライアント・アプリケーションのネットワーク・パケット・サイズを増加すると、ネットワーク・パフォーマンスが改善される場合もあります。

イベント 259：スレッショルド・クリアの最終機会まで待機

Adaptive Server がデータベース・ログの最後のスレッショルドをクロスすると、追加のログ領域の割り当てを試行しているプロセスは、メッセージ 7415 を受信し、使用可能なログ領域を待機する間スリープまたはサスペンド状態に移行します。イベント 259 は、プロセスがこの領域を待機する時間を測定します。

対処法

このイベントの Waits の高い値は、現在の大きさよりさらに大きなログ・セグメントが必要なデータベースがあることを示している可能性があります。WaitTime の高い平均値は、スレッショルドのプロシージャが定義されていないか、プロシージャがログ領域を解放するまでに長い時間がかかっていることを示している可能性があります。

データベースでのトランザクション・ダンプの頻度を増加するか、ログ・セグメントに追加の領域を割り当てると、WaitTime の値を低くすることができる場合があります。

イベント 260：waitfor コマンドの日付または時刻を待機

イベント 260 は、プロセスによって waitfor コマンドが使用されるときに発生する一般的なイベントです。

対処法

プロセスが waitfor コマンドを使用すると、Adaptive Server はその要求された期限が切れるまでそのプロセスをスリープ状態にします。イベント 260 は、このスリープ時間を測定します。

イベント266：ワーカー・スレッド・メールボックスのメッセージを待機

Adaptive Server のワーカー・スレッドは、メールボックスと呼ばれる Adaptive Server の内部メカニズムを使用して相互に通信するとともに、および親 spid と通信します。イベント 266 は、ワーカー・プロセスがメールボックスによるメッセージの追加を待機するために費やした時間を測定します。

対処法

イベント 266 を評価するには、monSysWorkerThread.ParallelQueries から実行された並列クエリの数を調べます。クエリごとの WaitTime の値が高い場合は、Adaptive Server のリソースが不足している可能性があります (一般的には CPU 時間)。WaitTime の高い値は、ワーカー・スレッドが別のワーカー・スレッドの完了を待機する原因となるオブジェクト上のパーティションのバランスの悪さを示す場合もあります。

イベント 272：ULC 上のロックを待機

各プロセスは、最終ログ・ページ上での競合を低減するためにユーザ・ログ・キャッシュ (ULC) 領域を割り当てます。2 つ以上のプロセスによる ULC のレコードへのアクセスとフラッシュの強制の可能性があるため、Adaptive Server はロックを使用して ULC を保護します。イベント 272 は、ULC がそのロックを待機するために費やした時間を測定します。

対処法

通常、イベント 272 の WaitTime の平均値は、非常に小さい値となります。WaitTime の高い平均値は、ULC ロックを保持しているプロセスに待機を強制し、他のイベントを待機する時間が長いことを示している可能性があります。これらの待機の原因を調べるために、他の待機イベントを分析してください。

イベント 334：Lava パイプ・バッファの書き込みを待機

Adaptive Server のバージョン 15.0 では、lava クエリ実行エンジンが導入されました。このエンジンによる並列クエリの実行時には、「パイプ・バッファ」と呼ばれる内部構造を使用してワーカー・プロセス間でデータを送受信します。イベント 334 は、パイプ・バッファが使用可能になるまで Adaptive Server が待機した時間を測定します。

対処法

このプロセスが適切に実行された場合、WaitTime は低い値を取ります。この値が高い場合には、サポート・センタに問い合わせてください。

イベント 374：ロック保留／データ保留のクリアを待機

ロック要求は、必要なロックがローカル・ノードで使用可能でないため、および現在未処理の要求があるためにブロックされています。

対処法

異なるノード間で同じロックに対して競合する要求が多数ある場合、アプリケーションが複数のノードから同じオブジェクトに対して競合するロック要求を発行していることを意味します。ロックは、クラスタ内のすべてのインスタンスのデータベース・オブジェクトを記述するメタデータを管理するオブジェクト一貫性マネージャから発行されます。

問題のオブジェクトのほとんどの要求を単一ノードにダイレクトすることにより、パフォーマンスが向上する場合があります。

イベント 375：OCM による BAST 処理の終了の待機

別のノードからの競合する ocm_lock 要求により、1 つまたは複数の ocm_lock 要求がブロックされています。

対処法

この待機イベントの値が高い場合、同じ ocm_lock に対して異なるノードが競合する多数の要求を送信していることを意味します。アプリケーションは、複数のノードから競合するこれらのロック要求を発行している可能性があります。ocm_lock のほとんどの要求を単一ノードにダイレクトすることにより、パフォーマンスが向上する場合があります。

イベント 389：OCM によるデータのプッシュ・フラグのクリアの待機

Adaptive Server は、以下の理由で 1 つまたは複数の ocm_lock 要求を処理できません。

- 別のロック要求がロックを保持している

  この場合、大きい値は、ocm_lock を失わずに、多量のオペレーションを実行する必要のある排他的ロック要求が多数あることを意味します。これは内部問題です。

  パフォーマンスを向上させるには、Adaptive Server が長時間ロックを保持している理由を特定してください。

- 別の要求がデータをプッシュしている

  この場合、大きい値は、データがメモリ内にのみ存在し、Adaptive Server はこのデータを定期的に他のノードにプッシュして、ノード障害の差異にデータが失われることを防ぐ必要があることを意味します。Adaptive Server では、データを他のノードにプッシュしている間は、有用な作業が行われないため、パフォーマンスに悪影響が及びます。これは内部問題です。

  パフォーマンスを向上させるには、Adaptive Server が長時間ロックを保持したり、データをプッシュしたりしている理由を特定してください。

- キャンセルされたトランザクションが未処理 (要求されたデータが配布されていない可能性があり、このノードの OCM が是正アクションをとっている)

  この待機イベントはまれです (通常、クラッシュしたばかりのノードからのデータをインスタンスが必要とする場合)。この待機イベントの値が大きくなるのは、ノードまたはハードウェア障害の率が高いためである可能性があります。

イベント 380：OCM_ERR_DIDNTWAIT 時にリセットするロック／データ保留

クラスタ・ロック・マネージャがロック要求にすぐに対応できない場合、LOCK_DIDNTWAIT メッセージが返されます。Adaptive Server はこのメッセージを OCM_ERR_DIDNT_WAIT に変換します。ocm_lock 要求は、AST がロック要求への応答と共に master データベースから返されるまでスリープ状態になります。

対処法

この待機イベントの値が大きい場合、同じ ocm_lock に対して異なるノード間で競合する多数の要求があることを意味します。これはアプリケーションが複数のノードから同じオブジェクトに対して競合するロック要求を発行していることが原因である可能性があります。問題のオブジェクトの要求を単一ノードにダイレクトすることにより、パフォーマンスが向上する場合があります。

イベント 483：マルチキャスト同期メッセージの確認応答の待機

この待機イベントは頻繁に発生しますが、通常のアクティビティの結果です。

対処法

対処する必要はありません。

索引

い

え

お

か

く

こ

さ

し

す

せ

た

ち

て

は

ひ

ふ

め

も

や

り