SYBASE®

Performance and Tuning Series: Monitoring Tables

Adaptive Server® Enterprise

15.0.2

DOCUMENT ID: DC00848-01-1502-01

LAST REVISED: December 2008

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

Contents

About This Book

Audience

This book is intended for Adaptive Server™ Enterprise system administrators using the monitoring tables to analyse the performance of their databases.

How to use this book

- Chapter 1, "Introduction to Monitoring Tables" – describes how to install and use the monitoring tables.
- Chapter 2, "Monitoring Tables Descriptions," – lists, in alphabetical order, monitoring tables and columns.
- Chapter 3, "Wait Events," – describes a selection of the more common wait events and actions you can perform to avoid them.

Related documents

The Adaptive Server® Enterprise documentation set consists of:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Product Manuals Web site.
- The Installation Guide for your platform – describes installation, upgrade, and some configuration procedures for all Adaptive Server and related Sybase products.
- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 15.0, the system changes added to support those features, and changes that may affect your existing applications.
- *ASE Replicator Users Guide* – describes how to use the Adaptive Server Replicator feature to implement basic replication from a primary server to one or more remote Adaptive Servers.
- *Component Integration Services Users Guide* – explains how to use the Component Integration Services feature to connect remote Sybase and non-Sybase databases.

- The *Configuration Guide* for your platform – provides instructions for performing specific configuration tasks.
- *Enhanced Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server Users Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes, functions, and stored procedures in the Adaptive Server database.
- *Job Scheduler Users Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).
- *Messaging Service Users Guide* – describes how to use Real Time Messaging Services to integrate TIBCO Java Message Service and IBM WebSphere MQ messaging services with all Adaptive Server database applications.
- *Monitor Client Library Programmers Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server Users Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Performance and Tuning Series* – is a series of books that explain how to tune Adaptive Server for maximum performance:
  - *Basics* – contains the basics for understanding and investigating performance questions in Adaptive Server.
  - *Improving Performance with Statistical Analysis* – describes how Adaptive Server stores and displays statistics, and how to use the set statistics command to analyze server statistics.
  - *Locking and Concurrency Control* – describes how to use locking schemes to improve performance, and how to select indexes to minimize concurrency.
  - *Monitoring Adaptive Server with sp_sysmon* – describes how to use sp_sysmon to monitor performance.

    - *Monitoring Tables* – describes how to query Adaptive Server monitoring tables for statistical and diagnostic information.
    - *Physical Database Tuning* – describes how to manage physical data placement, space allocated for data, and the temporary databases.
    - *Query Processing and Abstract Plans* – describes how the optimizer processes queries and how to use abstract plans to change some of the optimizer plans.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book (regular size when viewed in PDF format).
- *Reference Manual* – is a series of books with detailed Transact-SQL information:
    - *Building Blocks* – discusses datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
    - *Commands* – documents commands.
    - *Procedures* – includes system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.
    - *Tables* – discusses system tables and dbcc tables.
- *System Administration Guide* –
    - *Volume 1* – provides an introduction to the basics of system administration, including a description of configuration parameters, resource issues, character sets, sort orders, and instructions for diagnosing system problems. The second part of this book is an in-depth description of security administration.
    - *Volume 2* – includes instructions and guidelines for managing physical resources, mirroring devices, configuring memory and data caches, managing multiprocessor servers and user databases, mounting and unmounting databases, creating and using segments, using the reorg command, and checking database consistency. The second half of this book describes how to back up and restore system and user databases.
- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Full-size available only in print version; a compact version is available in PDF format.

- *Transact-SQL Users Guide* – documents Transact-SQL, the Sybase-enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the pubs2 and pubs3 sample databases.
- *Troubleshooting Series* –
  - *Troubleshooting: Error Messages Advanced Resolutions* – contains troubleshooting procedures for problems you may encounter. The problems discussed here are the ones the Sybase Technical Support staff hear about most often.
  - *Troubleshooting and Error Messages Guide* – contains detailed instructions on how to resolve the most frequently occurring Adaptive Server error messages. Most of the messages presented here contain error numbers (from the master..sysmessages table), but some error messages do not have error numbers, and occur only in the Adaptive Server error log.
- *Users Guide for Encrypted Columns* – describes how to configure and use encrypted columns with Adaptive Server.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Unified Agent and Agent Management Console* – describes the Unified Agent, which provides runtime services to manage, monitor, and control distributed Sybase resources.
- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Web Services Users Guide* – explains how to configure, use, and troubleshoot Web services for Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.

- *XML Services in Adaptive Server Enterprise* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that are available in XML services.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.
3. In the Certification Report filter select a product, platform, and timeframe and then click Go.

4 Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following sections describe conventions used in this manual.

SQL is a free-form language. There are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and most syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented. Complex commands are formatted using modified Backus Naur Form (BNF) notation.

Table 1 shows the conventions for syntax statements that appear in this manual:

Table 1: Font and syntax conventions for this manual

Element	Example
Command names,procedure names, utility names, and other keywords display in sans serif font.	select sp_configure
Database names and datatypes are in sans serif font.	master database
Book names, file names, variables, and path names are in italics.	*System Administration Guide* *sql.ini* file *column_name* *$SYBASE/ASE* directory

Variables—or words that stand for values that you fill in—when they are part of a query or statement, are in italics in Courier font.	`select column_name`
	`from table_name`
	`where search_conditions`

Type parentheses as part of the command.	compute *row_aggregate* (*column_name*)
Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates “is defined as”.	`::=`
Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces.	`{cash, check, credit}`
Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets.	`[cash | check | credit]`
The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command.	`cash, check, credit`
The pipe or vertical bar (|) means you may select only one of the options shown.	`cash | check | credit`

An ellipsis (...) means that you can *repeat* the last unit as many times as you like.	`buy thing = price [cash | check | credit]`
	`[, thing = price [cash | check | credit]]...`
	You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  sp_dropdevice [*device_name*]

  For a command with more options:

  select *column_name*
  from *table_name*
  where *search_conditions*

  In syntax statements, keywords (commands) are in normal font and identifiers are in lowercase. Italic font shows user-supplied words.

- Examples showing the use of Transact-SQL commands are printed like this:

  ```
  select * from publishers
  ```

- Examples of output from the computer appear as follows:

```
pub_id    pub_name                city          state
-------   ---------------------   -----------   -----
0736      New Age Books           Boston        MA
0877      Binnet & Hardley        Washington    DC
1389      Algodata Infosystems    Berkeley      CA

(3 rows affected)
```

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, SELECT, Select, and select are the same.

Adaptive Server sensitivity to the case of database objects, such as table names, depends on the sort order installed on Adaptive Server. You can change case sensitivity for single-byte character sets by reconfiguring the Adaptive Server sort order. For more information, see the *System Administration Guide*.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Adaptive Server HTML documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Introduction to Monitoring Tables

This chapter describes how to query Adaptive Server monitoring tables for statistical and diagnostic information.

Monitoring tables in Adaptive Server

Adaptive Server includes a set of system tables that contain monitoring and diagnostic information. The information in these tables provides a statistical snapshot of the state of Adaptive Server, which allows you to analyze the server so as to analyze server performance. For example, you can execute queries to report information about the activity of server processes and applications, query performance, usage of database tables, efficiency of data caches, I/O activity on database devices, and many other aspects of the Adaptive Server that affect system performance.

The data in the monitoring tables is not stored on disk. The data is calculated when you execute a query on one of the monitoring tables. Table definitions are contained in proxy table definitions that are created by a server installation script. These proxy tables use an interface to the Adaptive Server to collect monitoring data when you perform a query.

You must create the monitoring tables using a server installation script. See "Installing the monitoring tables" on page 3.

Because Adaptive Server versions may change the definitions of the monitoring tables, Sybase recommends that when you upgrade, you run the appropriate installation script before using the monitoring tables.

Note You must have the mon_role to query these tables. See "The mon_role and additional access controls" on page 9.

Where does the monitoring information come from?

Adaptive Server gathers the information for the monitoring tables from:

- Global monitor counters (for example. monSysWaits, which has a limited number of rows).
- Resource-specific monitor counters (for example, monCachedProcedures, for which the number of result rows depends on a snapshot of cached, compiled objects in procedure cache) associated with a single server resource, such as engines or the procedure or data cache.
- Active process status structures (for example, monProcessWaits and monProcessSQLText). The number of result rows for monProcessWaits or monProcessSQLText depends on the number of active user connections or the number of active process status structures, respectively.
- Circular buffers (for example, monSysStatement and monDeadLock). The number of result rows for monSysStatement or monDeadLock relate to configuration parameter settings and how much data the fast data pipes contain. This buffer is used by all of the historical monitoring tables.

For some large production servers, materializing some monitoring tables may require resources and time.

Using Transact-SQL to monitor performance

Providing monitoring information as tables enables you to use Transact-SQL to monitor Adaptive Server. For example, to identify the processes that have consumed the greatest CPU time or logical I/Os, use:

```
select SPID, Login = suser_name(ServerUserID), CPUTime,
```

```
    LogicalReads
    from master..monProcessActivity
    order by CPUTime desc
```

You can use the same query to find the processes that are using the most physical I/O by substituting PhysicalReads for CPUTime.

The information in each monitoring table can be sorted, selected, joined, inserted into another table, and treated much the same as the information in a regular Adaptive Server table.

The monitoring tables are read-only and do not allow updates because they are in-memory tables that are generated as they are queried. Additionally, you cannot create triggers on monitoring tables.

You can use access control commands such as grant and revoke select to restrict access to the monitoring tables.

The monitoring tables definitions use the Component Integration Services (CIS) proxy table feature, which allows Adaptive Server to define remote procedures as local tables.

Installing the monitoring tables

The installation procedure for monitoring tables for versions of Adaptive Server earlier than 15.0.2 differs from the procedure for version 15.0.2 and later. This section explains the installation procedures for the earlier versions.

Monitoring tables for Adaptive Server version 15.0.2 and later and the Cluster Edition:

- Are installed when you run the *installmaster* script
- Use materialized views
- Do require you to create the loopback server

Versions earlier than 15.0.2, except Cluster Edition

Create monitoring tables using the *installmontables* script located in the *$SYBASE/ASE-15_0/scripts* directory (*%SYBASE%\ASE-15_0\scripts* for Windows).

Run the *installmontables* script using the isql utility. For example:

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

Configuring loopback proxy server for 15.0.1 ESD #2 and earlier

Adaptive Server version 15.0.1 ESD #2 and earlier requires that a server named "loopback" be included in sysservers before you run the *installmontables* script. To create this server, enter:

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

@@servername cannot be NULL. If it currently is NULL, use sp_addserver to define a "local" server name. Restart the server for the change to *@@servername* to take effect.

Remotely accessing and editing monitoring tables

In versions 15.0.2 and later, Sybase provides *installmontables* as a sample script that describes how to remotely access monitoring tables (you need not run the *installmontables* script on a server that is directly monitored for Adaptive Server version 15.0.2 and later to create the monitoring tables). Run *installmontables* to view the instructions for editing. For example:

```
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
```

Configuring the monitoring tables to collect data

By default, Adaptive Server does not collect the monitoring information required by the monitoring tables. Use sp_configure to configure Adaptive Server to start collecting the monitoring information. There are a number of configuration options, listed below, that control the collection of monitoring data in various areas.

Many of the monitoring tables require that you enable one or more of the configuration options before Adaptive Server collects their data. Different tables require different options. Table 1-1 describes which configuration options are required for each monitoring table.

To configure Adaptive Server to collect general monitoring information:

1 By default, the enable cis configuration parameter is enabled (set to a value of 1) when you first configure Adaptive Server. Verify that this parameter is enabled.

2 The enable monitoring configuration parameter determines whether other monitoring options are enable; set enable monitoring to 1.

```
sp_configure "enable monitoring", 1
```

Adaptive Server displays a full list of configuration parameters specifically for monitoring when you enter:

```
sp_configure Monitoring
```

The configuration parameters that control the collection of monitoring information are:

- 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

Note See Chapter 5, "Setting Configuration Parameters," in *System Administration Guide, Volume One* for descriptions of the configuration parameters.

Allocating memory for pipe error messages

A number of monitoring tables use in-memory buffers (called "pipes") to collect monitoring data. These parameters control the amount of memory allocated for each pipe:

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

Adaptive Server can dynamically add memory to a pipe but cannot dynamically remove memory from it, so if you reduce the size of a pipe parameter, you must restart Adaptive Server for the new pipe size to take effect.

These are algorithms for determining the size for the parameters:

- For an individual Adaptive Server, the memory required for the each pipe configuration is:

  configuration_value x *number_of_engines*

- In a clustered environment, each cluster instance allocates the memory required to create the monitoring table pipes. See "Using monitoring tables in a clustered environment" on page 15.

Configuration parameters for the monitoring tables

Adaptive Server uses configuration parameters to control what data is collected for the monitoring tables. Many of the monitoring tables require you to enable configuration parameters before Adaptive Server collects data. If you are not using a monitoring table, disable the associated configuration parameters, reducing the load on the Adaptive Server caused by collecting monitoring data.

Table 1-1 lists all of the monitoring tables and the configuration parameters that apply to them. For many of the tables, even if a required parameter is not enabled, some of the data in the monitoring table is valid.

Table 1-1: Configuration parameters required for some monitoring tables

	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	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored
monCachePool	X																	
monCachedObject																		
monCachedProcedures								X										
monDataCache	X																	
monDeadLock	X		X	X														
monDeviceIO	X																	
monEngine	X																	
monErrorLog	X				X	X												

	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	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored
monIOQueue	X																	
monLicense																		
monLocks	X																	
monNetworkIO	X																	
monOpenDatabases	X																	
monOpenObjectActivity	X						X	X										
monOpenPartitionActivity	X							X										
monProcedureCache	X																	
monProcess	X																X	
monProcessActivity	X						X										X	
monProcessLookup																		
monProcessNetIO	X																	
monProcessObject	X							X										
monProcessProcedures																		
monProcessSQLText	X	X																X
monProcessStatement	X															X	X	
monProcessWaits	X										X						X	
monProcessWorkerThread	X																	
monState																		
monSysPlanText	X								X	X								
monSysSQLText	X											X	X					
monSysStatement	X							X						X	X	X		
monSysWaits	X																X	
monSysWorkerThread	X																	
monTableColumns																		
monTableParameters																		
monTables																		

	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	plan text pipe max messages	process wait events	SQL text pipe active	SQL text pipe max messages	statement pipe active	statement pipe max messages	statement statistics active	wait event timing	max SQL text monitored
monWaitClassInfo																		
monWaitEventInfo																		

Error 12036

If you query the monitoring tables, but have not enabled all the configuration parameters the tables require, Adaptive Server issues error 12036 but still runs the query. Although many of the monitoring tables contain accurate data even if you have not enabled all the configuration parameters, some data is incorrect because Adaptive Server is not collecting the data required to populate one or more columns in the table.

Consider enabling the required configuration parameters. See Table 1-1 for details.

The *mon_role* and additional access controls

Access to the monitoring tables is restricted to users with the mon_role. Only users who are granted this role can execute queries on the monitoring tables. You can grant or revoke select permissions on the monitoring tables from specific logins, roles, or groups to add additional access control to some (or all) of the monitoring tables. For information about acquiring roles, see Chapter 11, "Managing User Permissions," in the *System Administration Guide, Volume 1*.

Some of the monitoring tables may contain sensitive information. For example, the monSysSQLText and monProcessSQLText tables contain all the SQL text that is sent to an Adaptive Server. This text may contain information such as updates to employee salary records. Administrators should consider adding additional access restrictions to these tables, such as limiting access to users with specific roles, to meet the security requirements of their systems.

Note If you are using monitoring tables in a clustered environment, the Workload and LogicalCluster monitoring tables do not require the mon_role.

Wrapping counter datatypes

Some columns in the monitoring tables contain integer counter values that are incremented throughout the life of Adaptive Server. Once a counter reaches the highest value possible (2,147,483,647), it is reset to 0, which is called "wrapping."

Because of the potential for wrapping, the values of some columns in the monitoring tables may not reflect the total accumulated value since the server started. To effectively use this column data, calculate the difference in counter values over specific time periods and use the result of this sample instead of the cumulative value. For example, use the difference between the current value and the value 10 minutes earlier instead of the current value.

The values of different counters tend to increase at different rates. For example, on a busy system, the LogicalReads column in the monDataCache table increases rapidly. Use monTables to identify counters that are likely to wrap; a value of 1 or 3 in the monTableColumns.Indicators specifies columns that are prone to wrapping. A server's behavior depends on load and application activity , and the Indicator column provides a general guideline; review your server's data to identify counters that tend to wrap.

To display a list of columns that are counters, execute:

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

Stateful historical monitoring tables

A number of monitoring tables provide a record of individual events rather than information about the current state. These tables are called "historical" because the events reported in them provide a record of the history of the server over a period of time. Adaptive Server maintains context information for each client connection that accesses the historical tables, and on each successive query on the table returns only rows that the client has not previously received. This "stateful" property of the historical monitoring tables is designed to maximize performance and to avoid duplicate rows when used to populate a repository for historical data.

The historical monitoring tables are:

- monErrorLog
- monDeadLock
- monSysStatement
- monSysSQLText
- monSysPlanText

You can identify historical tables from their monTables.Indicators column:

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

The information returned from historical tables is stored in buffers, one for each historical monitoring table. The sizes of these buffers, which are specified by configuration parameters, affects the length of time data is stored. Use the sp_configure options to configure the size of the buffer and the information to be captured. The sp_configure options you use depend on which monitoring table you are configuring. For example, for the monSysPlanText table, configure:

- plan text pipe max messages – the number of messages to be stored for the particular buffer.
- plan text pipe active – to indicate whether Adaptive Server writes information to the buffer.

The following table lists the configuration parameters that affect the historical monitoring tables:

Monitoring table	Configuration parameters
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 and monSysSQLText	max SQL text monitored

Note Some historical tables require that you set other configuration parameters in addition to those listed above. See Table 1-3 on page 17.

The values of the max messages parameters determine the maximum number of messages per engine. Multiply this value by the number of configured engines to determine the total number of messages that can be stored.

Each message stored adds one row to the monitoring table. Once all entries in the buffer have been used, new messages overwrite old messages in the buffers, so only the most recent messages are returned.

See Chapter 5, "Setting Configuration Parameters" of the *System Administration Guide, Volume 1* and "Configuring the monitoring tables to collect data" on page 5 for more information about sp_configure.

Adaptive Server returns only the data that was added since the previous read, so you may get seemingly inconsistent result sets from queries that attempt to filter results using a where clause because:

- A select from the monitoring table marks all previously unread messages in the table as having been read.
- Adaptive Server language layer performs the filtering, so rows not contained in the result set of the query are still considered as "seen" by the connection.

In this example, the buffer associated with the monErrorLog table contains two messages:

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

(2 rows affected)
```

If you reconnect, the two messages are returned, but you receive the following messages when you filter the result set with a where clause:

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

And:

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

Because the first query moved the client connection's context to include both of the rows for spids 20 and 21, the second query does not return either of these rows. The filter specified in the first query required the server to retrieve and evaluate both rows to return the specified result. Adaptive Server marks the row for spid 21 as "read" even though it did not participate in a result set returned to the client connection.

Note Because of the stateful nature of the historical monitoring tables, do not use them for ad hoc queries. Instead, use a `select * into` or `insert into` to save data into a repository or temporary table and then perform analysis on the saved data.

Transient monitoring data

Because monitoring tables often contain transient data, take care when joining or using aggregates in your queries: results from these operations may be different if a query plan requires a table to be queried multiple times. For example:

```
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)
```

This example queries monProcessStatement twice; first to find the maximum CpuTime, and then to match the maximum. When Adaptive Server performs the second query, there are three potential outcomes returned from monProcessStatement:

- The statement performs more work, consuming more CPU, and having a CpuTime value greater than the previous maximum, so there is no match in the where clause, and the query returns no results.
- The statement finishes executing before the second query executes, yielding no results unless another statement used exactly the same amount of CPU as the previously obtained maximum.
- The statement does not use any additional CPU, and its value of CpuTime still matches the maximum. Only this scenario produces the expected results.

Sybase recommends that you save data from the monitoring tables to a temporary table or repository before you analyze it. Doing so freezes the data and eliminates the potentially undesirable results due to transient data or the stateful nature of the historical monitoring tables.

Using monitoring tables in a clustered environment

In a clustered environment,by default monitoring tables report on a per-instance (that is, a single server in the cluster) basis instead of returning cluster-wide results. This allows you to monitor the activities of processes and queries across the cluster to get a better understanding of the statistics for objects that may be opened on more than one instance, and resource usage on each instance in the cluster. For example, if you query the monitoring tables about a table, the table about which you are querying may be opened or accessed by more than one instance in the cluster, so the descriptors for this table—and the associated statistics—may be in memory on the instance. Statistics are not aggregated for the cluster. The statistical results for all instances are returned as a unioned result set with rows collected from each instance. Each instance is identified in the result set with a row in the InstanceID column.

Configuring the system view

In a clustered server, system_view is a session-specific setting that allows you to control the scope of monitoring data that queries return from the monitoring tables, sysprocesses, sp_who, and other commands. When you set system_view to cluster, queries on the monitoring tables return data from all active instances in the cluster. When you set system_view to instance, queries against the monitoring tables return data only for processes or objects that are active on the instance to which the client is connected.

Use the set command to configure the scope of the session:

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

where:

- instance – returns statistics for only the local instance. Cross-cluster requests are not sent to any other instance in the cluster.
- cluster – returns statistics for all instances in the cluster.
- clear – returns the system view to the configured default.

If you do not specify an InstanceID when you query a monitoring table or call a monitoring table RPC, the instance uses the current system_view configuration.

The session system view is inherited from its host logical cluster. Select the *@@system_view* global variable to determine the current system view.

InstanceID added to monitor instances

Table 1-2 describes monitoring tables to which the Cluster Edition adds the InstanceID column.

Table 1-2: Monitoring tables with InstanceID column

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	

Table 1-3 describes monitoring tables that return identical information for all instances.

Table 1-3: monitoring tables that include the same information for all instances

Table name	Description
monMon	Metadata view is identical on all instances.
monTableColumns	Metadata view is identical on all instances.
monTableParameters	Metadata view is identical on all instances.
monTables	Metadata view is identical on all instances.
monWaitClassInfo	List of descriptions is identical on all instances.
monWaitEventInfo	List of descriptions is identical on all instances.

Monitoring tables for the statement cache

Once enabled, the Adaptive Server statement cache stores the SQL text of ad hoc update, delete, and select commands and other statements likely to be reused. When the statement cache is enabled, the query plans for these statements are saved for reuse. When a new statement is issued, Adaptive Server searches the statement cache for a plan to reuse. If Adaptive Server finds a plan to reuse, the statement does not need to be recompiled, which likely leads to improved performance.

For more information about the statement cache, see Chapter 3, "Configuring Memory," in the *System Adminitration Guide, Volume 2*.

Literal parameterization allows Adaptive Server to recognize queries that are identical except for differeing in literal values in the where clause. In addition to performance benefits, literal parameterization leads to significant space reduction when the metrics and statements are stored in the cache.

The monitoring tables include two tables that can be used to analyze the status and performance of the statement cache: monStatementCache provides a summary snapshot of the statement cache, and monCachedStatement shows detailed information about each cached statement.

Configuring Adaptive Server to monitor the statement cache

Use enable stmt cache monitoring to configure Adaptive Server to collect the monitoring information on the statement cache.

Deleting statements from the statement cache

Use dbcc purgesqlcache to remove statements from the statement cache. When you specify the statement ID, only the corresponding statement is deleted from the cache.

The syntax for dbcc purgesqlcache is:

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

Obtaining the hash key from the SQL text

A hash key is generated based on a statement's text, and acts as an approximate key for the search mechanism in the statement cache. Since other monitoring tables display the statement's text, you can use the hash key as an approximate key to look up and compare SQL text in these tables.

For information about viewing the entire SQL text of a cached statement, see "Displaying text and parameter information for cached statements," below.

Adaptive Server includes two functions you can use to effectively compute the hash key. Use parse_text to verify the validity of the SQL text before computing the hash key. The syntax is:

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

Valid values for prm_opt are:

- 1 – indicates that parse_text will auto-parameterize the output text.
- -1 – indicates that the current session settings for literal parameterization determine whether the input text is parameterized.

If the SQL text is invalid, the parse_text function returns null.

Use hashbytes to compute the hash key over the statement's text. For example:

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

Displaying text and parameter information for cached statements

Use show_cached_text to view the SQL text of a cached statement. show_cached_text uses the statement ID as input and displays the text and parameter information of the corresponding statement.The syntax is:

```
select show_cached_text(SSQLID)
```

Use show_cached_text to obtain text for statements in the statement cache in queries. For example:

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

Examples of querying the monitoring tables

This section provides examples of querying the monitoring tables.

- To return a list of all available monitoring tables:

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

- To list the columns in a specific monitoring table, enter:

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

  You can determine the columns for any of the monitoring tables by substituting its name in the where clause and running the query.

- To determine which currently executing queries are consuming the most CPU, and to list the text of those queries, enter:

```
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
```

- To determine the hit ratio for the procedure cache for the life of Adaptive Server, enter:

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

  The following query also provides the hit ratio for a data cache. In this example, the hit ratio is calculated for a 10-minute interval rather than for the entire life of the server:

```
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
```

To calculate performance metrics for specific sample periods, create a baseline table that stores monitor values at the beginning of the sample period. You can calculate the change in monitor values during the sample period by subtracting the baseline values from the values at the end of the sample period.

Use queries from the following examples to calculate the hit ratio for a data cache, create a baseline, and calculate the amount of activity during the sample period.

- To create a stored procedure that prints the executed SQL and the backtrace for any process currently executing stored procedures, enter:

```
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
```

- To identify any indexes that were used for the table in the database with dbid 5 and object ID 1424005073, enter:

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

To determine if you can remove an index because it is not used by the applications running on your server:

a Run all queries in your applications that access the table in question. Ensure that Adaptive Server runs long enough so all applications have performed their selects.

b To determine whether your application did not use any of the indexes in your database, execute:

```
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
```

CHAPTER 2

Monitoring Tables Descriptions

This chapter describes the monitoring tables in alphabetical order.

The Attributes column provides information about how Adaptive Server manages the column. An Attribute value of:

- "Counter" indicates value in this column may wrap, or become zero and start incrementing again, because the value exceeds the maximum possible value of 2^{31}. Adaptive Server resets the monitor counters when you run sp_sysmon without the noclear option. In Adaptive Server version 15.0.1 and later, the noclear option is, by default, included as a sp_sysmon parameter. In versions earlier than 15.0.1, you must specify noclear to prevent Adaptive Server from resetting the monitor counters.

  Resetting monitor counters may skew your results if you run sp_sysmon on the same Adaptive Server on which you are using the monitoring tables.
- "Null" indicates the column value may be null.
- "Reset" indicates the column is reset when you run sp_sysmon in a manner that causes it to clear the monitoring counters (see *Performance and Tuning Series: Monitoring Adaptive Server with sp_sysmon* for more information).

monCachedObject

Description

Stores statistics for all tables, partitions, and indexes with pages currently in a data cache.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
CacheID	int		Unique identifier for the cache.
DBID	int		Unique identifier for the database.
IndexID	int		Unique identifier for the index.
PartitionID	int		Unique identifier for the partition. This is the same value as ObjectID for nonpartitioned objects.
CachedKB	int		Number of kilobytes of the cache the object is occupying.
CacheName	varchar(30)	Null	Name of the cache.
ObjectID	int	Null	Unique identifier for the object. Null if the descriptor for the object has been removed from the server's metadata cache. In this situation, you can determine the object identifier by querying syspartitions in the specified database for the value of PartitionID.
DBName	varchar(30)	Null	Name of the database (NULL if the descriptor for the object was removed from the server's metadata cache).
OwnerUserID	int	Null	Unique identifier for the object owner.
OwnerName	varchar(30)	Null	Name of the object owner (null if the descriptor for the object was removed from the server's metadata cache).
ObjectName	varchar(30)	Null	Name of the object (null if the descriptor for the object was removed from the server's metadata cache).
PartitionName	varchar(30)	Null	Name of the object partition (null if the descriptor for the object was removed from the server's metadata cache).
ObjectType	varchar(30)	Null	Object type (null if the object is no longer open).

Name	Datatype	Attributes	Description
TotalSizeKB	int	Counter, null	Partition size, in kilobytes.
ProcessesAccessing	int	Counter, null	Number of processes currently accessing pages for this object in the data cache.

monCachePool

Description

Stores statistics for all pools allocated for all data caches.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
CacheID	int		Unique identifier for the cache
IOBufferSize	int		Size (in bytes) of the I/O buffer for the pool
AllocatedKB	int		Number of bytes allocated for the pool
PhysicalReads	int	Counter	Number of buffers read from disk into the pool
Stalls	int	Counter	Number of times I/O operations were delayed because no clean buffers were available in the wash area for this data cache
PagesTouched	int	Counter	Number of pages that are currently being used within the pool
PagesRead	int	Counter	Number of pages read into the pool
BuffersToMRU	int	Counter	Number of buffers fetched and replaced in the most recently used portion of the pool
BuffersToLRU	int	Counter	Number of buffers fetched and replaced in the least recently used portion of the pool: fetch and discard.
CacheName	varchar(30)	Null	Name of the cache

monCachedProcedures

Description

Stores statistics for all stored procedures, triggers, and compiled plans currently stored in the procedure cache.

Note Enable the per object statistics active configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
ObjectID	int		Unique identifier for the procedure
OwnerUID	int		Unique identifier for the object's owner
DBID	int		Unique identifier for the database in which the object exists
PlanID	int		Unique identifier for the query plan for the object in the procedure cache
MemUsageKB	int		Number of kilobytes of memory used by the procedure
CompileDate	datetime		Date that the procedure was compiled
ObjectName	varchar(30)	Null	Name of the procedure
ObjectType	varchar(32)	Null	Type of procedure (for example, stored procedure or trigger)
OwnerName	varchar(30)	Null	Name of the object owner
DBName	varchar(30)	Null	Name of the database

monCachedStatement

Description

monCachedStatement stores detailed monitoring information about the statement cache, including information about resources used during the previous executions of a statement, how frequently a statement is executed, the settings in effect for a particular plan, the number of concurrent uses of a statement, and so on. This information can be helpful when troubleshooting, and when deciding which statements to retain in the cache.

The columns in monCachedStatement allow two attributes: “counter” if the column has a counter value, and “reset” if you can reset the column using sp_sysmon.

Note Enable the enable monitoring and enable statement cache monitoring configuration parameters, and set the statement cache size parameter greater than 0 for this monitoring table to collect data.

Columns

Name	Datatype	Attribute	Description
SSQLID	int		Unique identifier for each cached statement. This value is treated as a primary key for monCachedStatement, and is used in functions. show_cached_plan and show_cached_text both use SSQLID to refer to individual statements in the cache.
HashKey	int		Hash value of the SQL text of the cached statement. A hash key is generated based on a statement’s text, and can be used as an approximate key for searching other monitoring tables.
UserID	int		User ID of the user who initiated the statement that has been cached.
SUserID	int		Server ID of the user who initiated the cached statement.
DBID	int		Database ID of the database from which the statement was cached.
DBName	varchar(30)	Null	Name of database from which the statement was cached.

Name	Datatype	Attribute	Description
CachedDate	datetime		Timestamp of the date and time when the statement was first cached.
LastUsedDate	datetime		Tmestamp of the date and time when the cached statement was last used. Use this information with CachedDate to determine how frequently this statement is used, and whether it is helpful to have it cached.
CurrentUsageCount	int	Counter	Number of concurrent users of the cached statement.
StatementSize	int		Size of the cached statement, in bytes.
MaxUsageCount	int	Counter	Maximum number of times the cached statement's text was simultaneously accessed.
SessionSettings			These session-level settings are associated with each cached statement.
ParallelDegree			Degree of parallelism used by the query that is stored for this statement
QuotedIdentifier			Specifies whether the plan compiled with set quoted_identifier is enabled.
TransactionIsolationLevel			Transaction isolation level for which the statement was compiled.
TransactionMode			Specifies whether "chained transaction mode" is enabled for the statement.
SAAuthorization			Specifies whether the plan was compiled with sa_role authorization.

Name	Datatype	Attribute	Description
SystemCatalogUpdates			Specifies whether allow catalog updates was enabled when the plan was compiled.
ExecutionMetrics			Execution costs are collected when the cached plan is used. These costs are measured in terms of values for logical I/O (LIO) and physical I/O (PIO), execution, and elapsed times. The metrics that ExecutionMetrics reports are the same as those reported by QP metrics, however, the data reported in this table does not require that you enable QP metrics. monCachedStatement captures the metrics independently for the cached statements, regardless of Adaptive Server metrics capture settings.
MetricsCount			Number of times metrics were aggregated for this statement.
MaxElapsedTime	int		Maximum elapsed execution time for this statement.
MinElapsedTime	int		Minimum elapsed execution time for this statement.
AvgElapsedTime	int		Average elapsed execution time for this statement.
MaxLIO	int		Maximum logical I/Os that occurred during any one execution of this statement.

Name	Datatype	Attribute	Description
MinLIO	int		Minimum logical I/Os that occurred during any execution of this statement.
AvgLIO	int		Average number of logical I/Os that occurred during execution of this statement.
MaxPIO	int		Maximum physical I/Os that occurred during any execution of this statement.
MinPIO	int		Maximum physical I/Os that occurred during any execution of this statement.
AvgPIO	int		Average number of physical I/Os that occurred during execution of this statement.
NumRecompilesPlanFlushes	int	Counter	Number of times the cached statement was recompiled because a plan was not found in the cache.
NumRecompilesSchemaChanges	int	Counter	Number of times the statement was recompiled due to schema changes. Running update statistics on a table may result in changes to the best plan. This change is treated as a minor schema change. Recompiling a statement many times indicates that it is not effective to cache this particular statement, and that you may want to delete the statement from the statement cache to make space for some other, more stable, statement.
MaxPlanSize	int		Size of the plan when it is in use, in kilobytes.

Name	Datatype	Attribute	Description
MinPlanSize	int		Size of the plan when it is not in use, in kilobytes.
LastRecompiledDate	datetime		Date when the statement was last recompiled, because of schema changes or because the statement was not found in the statement cache.
UseCount	int		Number of times the statement was accessed after it was cached.
HasAutoParams	boolean		"true" if the statement has any parameterized literals, "false" if it does not.

monCIPC

Description

monCIPC provides summary figures for total messaging within the cluster, as viewed from the current instance or all instances. This monitoring table is available only for Adaptive Server Cluster Edition.

One row is returned in the monCIPC table for each instance in the cluster, if the system view is set to cluster; otherwise, a single row is returned for the instance on which the query is executed.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster
ReceivedCount	int	Number of messages received by this instance
TransmitCount	int	Number of messages sent by this instance
Multicast	int	Number of messages sent that were addressed to all other instances in the cluster
Synchronous	int	Number of those messages sent synchronously

Name	Datatype	Description
ReceiveSoftErrors	int	Number of recoverable errors received on this instance
ReceiveHardErrors	int	Number of unrecoverable errors received on this instance
TransmitsSoftErrors	int	Number of recoverable transmit errors on this instance
TransmitHardErrors	int	Number of unrecoverable transmit errors on this instance
Retransmits	int	Number of retransmissions performed by this instance
Switches	int	Number of switches between the primary interconnect network and the secondary interconnect network
FailedSwitches	int	Number of attempts to switch between primary and secondary interconnect networks that failed

monCIPCEndpoints

Description

monCIPCEndpoints provides a detailed summary, giving traffic data for each subsystem within the cluster instance. This monitoring table is available only for Adaptive Server Cluster Edition.

One row is returned for each logical endpoint in the instance. If the system view is set to cluster, a set of rows is returned for each node in the cluster.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster
ReceivedCount	int	Number of messages received by this logical endpoint within the cluster
TransmitCount	int	Number of messages sent by this logical endpoint within the instance
ReceiveBytes	int	Number of bytes received by this logical endpoint within the instance

Name	Datatype	Description
TransmitBytes	int	Number of bytes sent by this logical endpoint within the instance
ReceiveQ	int	Current number of messages queued for this logical endpoint
MaxReceiveQ	int	Maximum number of messages ever observed queued for this logical endpoint
DoneQ	int	Current number of messages for this logical endpoint that were processed and await further action
MaxDoneQ	int	Maximum number of messages ever observed for this logical endpoint, which have been processed and await further action
EndPoint	varchar	Name of CIPC endpoint

monCIPCLinks

Description

monCIPCLinks monitors the state of the links between instances in the cluster. This monitoring table is available only for Adaptive Server Cluster Edition

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster.
LocalInterface	varchar30	Name of the link's local network endpoint. Same name that appears in the *hosts* file for a server name.
Remote Interface	varchar30	Name of the link's remote end point. Same name that appears in the *hosts* file for a server name.
PassiveState	varchar10	Latest state listed in the traffic on the link.
PassiveStateAge	int	Time since the PassiveState column was updated, in milliseconds.

Name	Datatype	Description
ActiveState	varchar10	Latest state used, as determined by active monitoring (when no traffic was present on the link).
ActiveStateAge	int	Time since the ActiveState column was updated, in milliseconds.

monCIPCMesh

Description

monCIPCMesh gives summary figures for the mesh of connections, from the current instance to all other instances in the cluster, on a per-instance basis. This monitoring table is available only for Adaptive Server Cluster Edition.

One row is returned for each of the four connections to each of the other nodes in the cluster, up to the maximum configured. If the system view is cluster, a set of rows for each instance active in the cluster is returned.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster.
FarInstanceID	tinyint	Instance number of the far-end instance in the cluster.
Received	int	Number of messages received by this instance from the FarInstanceID instance.
Dropped	int	Number of messages from the FarInstanceID instance that were dropped, due to a lack of resources.
Transmitted	int	Number of messages transmitted to the FarInstanceID instance.
Resent	int	Number of messages re-sent to the FarInstanceID instance.
Retry	int	Number of packets retried to the FarInstanceID instance.
SendQ	int	Current number of messages waiting to be sent to the FarInstanceID instance.

Name	Datatype	Description
MaxSetQ	int	Maximum number of messages sent, but notification of sending is not yet processed.
Mesh	varchar	The channel name for the connection. One of: • Out of Band • Message • Large Message • Direct memory access (DMA)
MinRTT	int	Minimum round-trip delay observed for messages (applies only to user datagram protocol (UDP) transport).
MaxRTT	int	Maximum round trip delay observed for messages (applies only to UDP transport).
AverageRTT	int	Average round trip delay observed for messages (applies only to UDP transport).

monClusterCacheManager

Description

monClusterCacheManager stores diagnostic information about the cluster cache manager daemon running on each instance. monClusterCacheManager reports cluster-wide information on a per-instance basis. This monitoring table is available only for Adaptive Server Cluster Edition.

Columns

The columns for monClusterCacheManager are:

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster
DaemonName	varchar	Name of the cluster cache manager daemon
RequestsQueued	int	Number of requests queued to the cluster cache manager daemon
RequestsRequeued	int	Number of requests requeued to the cluster cache manager daemon
RequestsServiced	int	Number of requests serviced by the cluster cache manager daemon
TransfersInitiated	int	Number of transfers initiated by the cluster cache manager daemon
Downgrades	int	Number of downgrades performed by the cluster cache manager daemon

Name	Datatype	Description
Releases	int	Number of releases performed by the cluster cache manager daemon
DiskWrites	int	Number of disk writes initiated by the cluster cache manager daemon
SleepCount	int	Number of times the cluster cache manager daemon went to sleep
AvgServiceTime	int	Average time (in milliseconds) spent servicing a request
MaxQSize	int	Maximum number of requests queued to the cluster cache manager daemon at any time since the instance started

monDataCache

Description

Stores statistics relating to Adaptive Server data caches.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
CacheID	int		Unique identifier for the cache
RelaxedReplacement	int		Specifies whether the cache is using relaxed cache replacement strategy
BufferPools	int		Number of buffer pools within the cache
CacheSearches	int	Counter, reset	Cache searches directed to the cache
PhysicalReads	int	Counter, reset	Number of buffers read into the cache from disk
LogicalReads	int	Counter, reset	Number of buffers retrieved from the cache
PhysicalWrites	int	Counter, reset	Number of buffers written from the cache to disk

Name	Datatype	Attributes	Description
Stalls	int	Counter, reset	Number of times I/O operations were delayed because no clean buffers were available in the wash area
CachePartitions	smallint		Number of partitions currently configured for the cache
CacheName	varchar(30)	Null	Name of cache

monDeadLock

Description

Provides information about deadlocks. Use deadlock pipe max messages to tune the maximum number of messages returned.

monDeadLock is an historical monitoring table. See "Stateful historical monitoring tables" on page 11.

Use sp_monitor 'deadlock' to check current deadlock options. The deadlock parameter provides a number of reports based on monDeadLock, which are useful for analyzing the history of server deadlocks.

Note Enable the enable monitoring, deadlock pipe active, and deadlock max pipe messages configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
DeadLockID	int		Unique identifier for the deadlock
VictimKPID	int		Kernel process ID (kpid) of the victim process for the deadlock
ResolveTime	datetime		Time when the deadlock was resolved
ObjectDBID	int		Unique database identifier for database where the object resides
PageNumber	int		Page number requested for the lock, if applicable
RowNumber	int		Row number requested for the lock, if applicable
HeldFamilyId	smallint		spid of the parent process holding the lock

Name	Datatype	Attributes	Description
HeldSPID	smallint		spid of process holding the lock
HeldKPID	int		kpid of process holding the lock
HeldProcDBID	int		Unique identifier for the database where the stored procedure that caused the lock to be held resides, if applicable
HeldProcedureID	int		Unique object identifier for the stored procedure that caused the lock to be held, if applicable
HeldBatchID	int		Identifier of the SQL batch executed by the process holding the lock when the deadlock occurred
HeldContextID	int		Unique context identifier for the process holding the lock when it was blocked by another process (not when it acquired the lock)
HeldLineNumber	int		Line number within the batch of the statement being executed by the process holding the lock when it was blocked by another process (not when it acquired the lock)
WaitFamilyId	smallint		spid of the parent process waiting for the lock
WaitSPID	smallint		spid of the process waiting for the lock
WaitKPID	int		kpid of the process waiting for the lock
WaitTime	int		Amount of time, in milliseconds, that the waiting process was blocked before the deadlock was resolved
ObjectName	varchar(30)	Null	Name of the object
HeldUserName	varchar(30)	Null	Name of the user for whom the lock is being held
HeldApplName	varchar(30)	Null	Name of the application holding the lock
HeldTranName	varchar(255)	Null	Name of the transaction in which the lock was acquired
HeldLockType	varchar(20)	Null	Type of lock being held

Name	Datatype	Attributes	Description
HeldCommand	varchar(30)		Category of process or command that the process was executing when it was blocked
WaitUserName	varchar(30)	Null	Name of the user for whom the lock is being requested
WaitLockType	varchar(20)	Null	Type of lock requested

monDeviceIO

Description

Returns statistical information relating to activity on database devices.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
Reads	int	Counter, reset	Number of reads from the device
APFReads	int	Counter, reset	Number of asynchronous prefetch (APF) reads from the device
Writes	int	Counter, reset	Number of writes to the device
DevSemaphoreRequests	int	Counter, reset	Number of I/O requests to a mirrored device (if mirrored)
DevSemaphoreWaits	int	Counter, reset	Number of tasks forced to wait for synchronization of an I/O request to a mirrored device (if mirrored)
IOTime	int)	Counter	Total amount of time (in milliseconds) spent waiting for I/O requests to be satisfied
LogicalName	varchar(30)	Null	Logical name of the device
PhysicalName	varchar(128)	Null	Full hierarchic file name of the device

monEngine

Description

Provides statistics regarding Adaptive Server engines.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
EngineNumber	smallint		Number of the engine.
CurrentKPID	smallint		Kernel process identifier (kpid) for the currently executing process.
PreviousKPID	int		kpid for the previously executing process.
CPUTime	int	Counter, reset	Total time, in seconds, the engine has been running.
SystemCPUTime	int	Counter, reset	Time, in seconds, the engine has been executing system database services.
UserCPUTime	int	Counter, reset	Time, in seconds, the engine has been executing user commands.
IdleCPUTime	int	Counter, reset	Time, in seconds, the engine has been in idle spin mode.
Yields	int	Counter, reset	Number of times this engine yielded to the operating system. modify the rate of yielding during idle periods using runnable process search count.
Connections	int	Counter	Number of connections this engine handles.
DiskIOChecks	int	Counter, reset	Number of times the engine checked for asynchronous disk I/O. Modify the frequency of these checks with i/o polling process count.

Name	Datatype	Attributes	Description
DiskIOPolled	int	Counter, reset	Number of times the engine polled for completion of outstanding asynchronous disk I/O. The polling occurs whenever disk I/O checks indicate that asynchronous I/O has been posted and is not yet complete.
DiskIOCompleted	int	Counter, reset	Number of asynchronous disk I/Os completed when the engine polled for outstanding asynchronous disk I/O.
MaxOutstandingIOs	int		Current number of I/O requests initiated by this engine that are not completed.
ProcessesAffinitied	int		Number of processes associated with this engine.
ContextSwitches	int	Counter, reset	Number of context switches.
HkgcMaxQSize	int		Maximum number of items Adaptive Server can queue for housekeeper garbage collection in this engine.
HkgcPendingItems	int		Number of items yet to be collected by housekeeper garbage collector on this engine.
HkgcHWMItems	int		Maximum number of pending items queued for housekeeper garbage collector at any instant since server started.
HkgcOverflows	int		Number of items that could not be queued to housekeeper garbage collector due to queue overflows.
Status	varchar(20)	Null	Status of the engine (online, offline, and so on).

Name	Datatype	Attributes	Description
Starttime	datetime	Null	Date that the engine came online.
StopTime	datetime		Date that the engine went offline.
AffinitiedToCPU	int	Null	Number of the CPU to which the engine is affinitied.
OSPID	int	Null	Identifier for the operating system process executing the engine.

monErrorLog

Description

Returns the most recent error messages from the Adaptive Server error log. Use errorlog pipe max messages to tune the maximum number of messages returned. See "Stateful historical monitoring tables" on page 11.

Note Enable the enable monitoring, errorlog pipe active, and errorlog pipe max messages configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier (spid)
KPID	int		Kernel process identifier (kpid)
FamilyID	smallint		spid of the parent process
EngineNumber	smallint		Engine on which the process was running
ErrorNumber	int		Error message number
Severity	int		Severity of error
State	int		State of error
Time	datetime		Timestamp when error occurred
ErrorMessage	varchar(512)	Null	Text of the error message

monIOQueue

Description

Provides device I/O statistics displayed as data and log I/O for normal and temporary databases on each device.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
IOs	int	Counter	Total number of I/O operations
IOTime	int	Counter	Amount of time (in milliseconds) spent waiting for I/O requests to be satisfied
LogicalName	varchar(30)	Null	Logical name of the device
IOType	varchar(12)	Null	Category for grouping I/O. One of UserData, UserLog, TempdbData, TempdbLog.

monLicense

Description

Provides a list of all licences currently checked out by the Adaptive Server.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
Quantity	int		Quantity of licenses used for this feature.
Name	varchar(30)	Null	Name of the feature license.
Edition	varchar(30)	Null	Edition of Adaptive Server for which this feature is licensed.
Type	varchar(64)	Null	License type.
Version	varchar(16)	Null	Version of the feature license in use
Status	varchar(30)	Null	Status of this feature license (that is, whether the license is withing a grace period expired).

Name	Datatype	Attributes	Description
LicenseExpiry	datetime	Null	Date that the license expires, if this is an expiring license.
GraceExpiry	datetime	Null	Date this license expires, if this license was awarded on grace. Refer to the Status column to determine whether this license was awarded a grace period.
LicenseID	varchar(150)	Null	License identifier. This may not be available if the license has been awarded a grace period.
Filter	varchar(14)	Null	Filter used when selecting this feature license. Use sp_lmconfig to change the filter.
Attributes	varchar(64)	Null	License attributes. These attributes are "*name*=*value*" pairs which, if specified, limit certain characteristics of Adaptive Server. Possible limiters are: • ME = maximum number of engines • MC = maximum number of connections • MS = maximum number of disk space • MM = maximum number of memory • CP = maximum number of CPUs

monLocks

Description

Returns a list of granted locks and pending lock requests.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier of process holding or requesting the lock.
KPID	int		Kernel process identifier
DBID	int		Unique identifier for this database object.
ParentSPID	smallint		Parent process ID.
LockID	int		Lock object ID.
BlockedBy	int		If the lock request is blocked, the BlockedBy column is the session process identifier for the process holding the lock that is blocking this lock request. Null if request is not blocked.
BlockedState	varchar(64)		Lock state if the lock being held is blocking other lock requests or if the lock request is blocked. Values are: • Blocked • Blocking • Demand • Detached • Null (if there is no blocking condition)
Context	int		Lock context (bit field). These values are the same as for those of the context column in syslocks. See the *Reference Manual: Tables* for information about syslocks.
ObjectID	int	Null	Unique identifier for the object
LockState	varchar(20)	Null	Indicates if the lock is granted. Values are: • Granted • Requested
LockType	varchar(20)	Null	Type of lock. Values are: • Exclusive • Shared • Update

Name	Datatype	Attributes	Description
LockLevel	varchar(30)	Null	The type of object for which the lock was requested. Values are: • Row • Page • Table • Address
WaitTIme	int	Null	The time (in seconds) for which the lock request was not granted.
PageNumber	int	Null	Page that is locked when `LockLevel = 'PAGE'`
RowNumber	int	Null	Row that is locked when `LockLevel = 'ROW'`

monLogicalCluster

Description

Displays information about the logical clusters currently configured on the system. This monitoring table is available only for Adaptive Server Cluster Edition.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
LCID	int	Logical cluster ID.
Name	varchar(30)	Logical cluster name.
State	varchar(20)	Current state. One of: • Online • Offline • Failed • Inactive • Time_wait

Name	Datatype	Description
DownRoutingMode	varchar(20)	Down routing-mode setting. One of: • System • Open • Disconnect
FailoverMode	varchar(20)	Failover mode setting, `instance` or `cluster`.
Attributes	int	Bitmask of logical cluster attributes.
StartupMode	varchar(20)	Start-up mode setting, `automatic` or `manual`.
SystemView	varchar(20)	System view setting, `instance` or `cluster`.
Roles	varchar(20)	Comma-delimited list of special roles for this logical cluster. The “system” logical cluster always has the system role. The open logical cluster has the “open” role. If the system logical cluster also has the open role, the value for this column is `system, open`. Logical clusters without any special roles return a null value.
LoadProfile	varchar(30)	Load profile associated with this logical cluster.
ActiveConnections	int	Number of active connections using this logical cluster.
BaseInstances	tinyint	Number of instances configured as base instances for this logical cluster.
ActiveBaseInstances	tinyint	Number of base instances on which this logical cluster is currently active.
FailoverInstances	tinyint	Number of instances configured as failover instances for this logical cluster.
ActiveFailoverInstances	tinyint	Number of failover instances on which this logical cluster is currently active.

monLogicalClusterAction

Description

Shows all administrative actions against local clusters from start-up until these actions are released. This monitoring table is available only for Adaptive Server Cluster Edition.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
Handle	int	Unique handle used to cancel this action.
State	varchar(20)	State of the action: `active`, `complete`, or `canceled`.
LCID	int	Logical cluster ID to which this action applies.
LogicalClusterName	varchar(30)	Logical cluster name of this logical cluster (denormalized to reduce joins).
Action	varchar(15)	Action being performed. A combination of the command running and its scope. For example, offline instance or failover cluster.
FromInstances	varchar(96)	A comma-separated list of `from instances` for this command and action (instance being brought offline).
ToInstances	varchar(96)	A comma-separated list of `to instances` for this command and action (instances being brought online).
InstancesWaiting	int	Number of instances waiting to go offline (this is a count of FromInstances that are in the `time_wait` state).
WaitType	varchar(20)	Current wait state for this action. One of: `wait`, `until`, or `nowait`.
StartTime	datetime	Date and time the command was issued.
Deadline	datetime	Date and time the command must be finished (based on the time value supplied to the `wait` or `until` options).

Name	Datatype	Description
CompleteTime	datetime	Date and time the command and action completed (when InstancesWaiting is zero and the action went from `active` to the `complete` state). Returns NULL for incomplete actions.
ConnectionsRemaining	int	Number of connections remaining to move as a result of this command.
NonMigConnections	int	Number of connections to be terminated because they do not support the migration protocol.
NonAHConnections	int	Number of connections that do not support the high availability failover protocol. These connections are disconnected and cannot fail over when the command finishes.

monLogicalClusterInstance

Description

Stores information about the many-to-many relationship between instances and logical clusters. You need not have the mon_role role to query this monitor table. This monitoring table is available only for Adaptive Server Cluster Edition

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
LCID	int	Logical cluster ID
LogicalClusterName	varchar(30)	Logical cluster name
InstanceID	tinyint	ID of the instance within the cluster
InstanceName	varchar(30)	Instance name
Type	varchar(20)	Instance type
FailoverGroup	tinyint	Failover group to which this instance is a member (failover instances only)
State	varchar(20)	State of this instance with respect to the logical cluster

Name	Datatype	Description
ActiveConnections	int	Number of active connections for this logical cluster on this instance
NonMigConnections	int	Number of active connections that do not support the connection migration protocol
NonHAConnections	int	Number of active connections that do not support the high availability failover protocol
LoadScore	real	Workload score for this instance using the load profile associated with its logical cluster

monLogicalClusterRoute

Description

Displays information about the configured routes (application, login, and alias bindings). You need not have the mon_role role to query this monitor table. This monitoring table is available only for Adaptive Server Cluster Edition

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
LCID	int	Logical cluster ID
LogicalClusterName	varchar(30)	Logical cluster name
RouteType	varchar(20)	Route type. One of: `application`, `login`, or `alias`
RouteKey	varchar(30)	Application, login, or alias name associated with this route.

monNetworkIO

Description

Returns network I/O statistics for all communication between Adaptive Server and client connections.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
PacketsSent	int	Counter, reset	Number of packets sent
PacketsReceived	int	Counter, reset	Number of packets received
BytesSent	int	Counter, reset	Number of bytes sent
BytesReceived	int	Counter, reset	Number of bytes received

monOpenDatabases

Description

Provides state and statistical information pertaining to databases that are currently in the server's metadata cache.

If the value of number of open databases is too low, Adaptive Server may flush database descriptors from the metadata cache. If this occurs, Adaptive Server loses the database statistics, but the statistics are reinitialized the next time the database descriptor is installed in the metadata cache.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
DBID	int		Unique identifier for the database
BackupInProgress	int		Specifies whether a backup is currently in progress for the database
LastBackupFailed	int		Specifies whether the last backup of the database failed

Name	Datatype	Attributes	Description
TransactionLogFull	int		Specifies whether the database transaction log is full
AppendLogRequests	int	Counter	Number of semaphore requests when attempting to append to the database transaction log
AppendLogWaits	int	Counter	Number of times a task had to wait for the append log semaphore to be granted
DBName	varchar(30)	Null	Name of the database
BackupStartTime	datetime	Null	Date the last full database backup started
SuspendedProcesses	int	Null	Number of processes currently suspended due to the database transaction log being full
QuiesceTag	varchar(30)	Null	Tag used in the quiesce database command for this database if the database is in a quiesced state
LastCheckpointTime	datetime	Null	Date and time checkpoint last ran for this database
LastTranLogDumpTime	datetime	Null	Date and time of this database's most recently successful transaction log dump

monOpenObjectActivity

Description

Provides statistics for all open tables and indexes.

Note Enable the enable monitoring, object lockwait timing, and per object statistics active configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
DBID	int		Unique identifier for the database.
ObjectID	int		Unique identifier for the object.
IndexID	int		Unique identifier for the index..
DBName	varchar(30)	Null	Name of the database in which the object resides
ObjectName	varchar(30)	Null	Name of the object.
LogicalReads	int	Counter, null	Total number of times a buffer for this object has been retrieved from a buffer cache without requiring a read from disk.
PhysicalLocks	int		Number of physical locks requested per object. Available only for Adaptive Server Cluster Edition.
PhysicalLocksDeadlocks	int		Number of times a requested physical lock returned a deadlock. The Cluster Physical Locks subsection of sp_sysmon uses this counter to report deadlocks while acquiring physical locks for each object. Available only for Adaptive Server Cluster Edition.
PhysicalLocksPageTransfer	int		Number of page transfers that occurred when an instance requested a physical lock. The Cluster Physical Locks subsection of sp_sysmon uses this counter to report the node-to-node transfer and physical-lock acquisition as a node affinity ratio for this object. Available only for Adaptive Server Cluster Edition.

Name	Datatype	Attributes	Description
PhsycialLocksRetained	int		Number of physical locks retained. Use to identify the lock hit ratio for each object. Good hit ratios imply balanced partitioning for this object. Available only for Adaptive Server Cluster Edition.
PhysicalLocksWaited	int		Number of times an instance waited for a physical lock request. Available only for Adaptive Server Cluster Edition.
PhysicalReads	int	Counter, null	Number of buffers read from disk.
APFReads	int	Counter, null	Number of APF buffers read from disk.
PagesRead	int	Counter, null	Total number of pages read.
PhysicalWrites	int	Counter, null	Total number of buffers written to disk.
PagesWritten	int	Counter, null	Total number of pages written to disk.
RowsInserted	int	Counter, null	Number of rows inserted.
RowsDeleted	int	Counter, null	Number of rows deleted.
RowsUpdated	int	Counter, null	Number of updates.
Operations	int	Counter, null	Number of times the object was accessed.
LockRequests	int	Counter, null	Number of requests for a lock on the object.
LockWaits	int	Counter, null	Number of times a task waited for an object lock.
OptSelectCount	int	Counter, null	Number of times the optimizer selected this index to be used in a query plan.
LastOptSelectDate	datetime	Null	Last date the index was selected for a plan during compilation.
UsedCount	int	Counter, null	Number of times the object was used in a plan during execution.

Name	Datatype	Attributes	Description
LastUsedDate	datetime	Null	Last date the index was used in a plan during execution.

Note Because you can use the plan for a stored procedure or trigger multiple times, the value of the OptSelectCount column may be less than the value of UsedCount. In addition, because Adaptive Server may decide not to execute certain portions of a query plan during execution, the UsedCount may be less than the OptSelectCount.

monOpenPartitionActivity

Description

Provides information about the use of each open partition on the server.

Note Enable the enable monitoring and per object statistics active configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
DBID	int		Unique identifier for the database.
ObjectID	int		Unique identifier for the object.
IndexID	int		Unique identifier for the index.
PartitionID	int		Unique identifier for the partition.
DBName	varchar(30)	Null	Name of the database in which the object resides.
ObjectName	varchar(30)	Null	Name of the object.
PartitionName	varchar(30)	Null	Name of the partition.
LogicalReads	int	Counter, null	Total number of buffers read.
PhysicalLocks	int		Number of physical locks requested per object. Available only for Adaptive Server Cluster Edition.

Name	Datatype	Attributes	Description
PhysicalLocksDeadlocks	int		Number of times a physical lock requested returned a deadlock. The Cluster Physical Locks subsection of sp_sysmon uses this counter to report deadlocks while acquiring physical locks for each object. Available only for Adaptive Server Cluster Edition.
PhysicalLocksPageTransfer	int		Number of page transfers that occurred when an instance requested a physical lock. The Cluster Physical Locks subsection of sp_sysmon uses this counter to report the node-to-node transfer and physical-lock acquisition as a node affinity ratio for this object. Available only for Adaptive Server Cluster Edition.
PhsycialLocksRetained	int		Number of physical locks retained. Use to identify the lock hit ratio for each object. Good hit ratios imply balanced partitioning for this object.
PhysicalLocksWaited	int		Number of times an instance waited for a physical lock request. Available only for Adaptive Server Cluster Edition.
PhysicalReads	int	Counter, null	Number of buffers read from disk.
APFReads	int	Counter, null	Number of asynchronous prefetch (APF) buffers read.
PagesRead	int	Counter, null	Total number of pages read.
PhysicalWrites	int	Counter, null	Total number of buffers written to disk.
PagesWritten	int	Counter, null	Total number of pages written to disk.

Name	Datatype	Attributes	Description
RowsInserted	int	Counter, null	Number of rows inserted.
RowsDeleted	int	Counter, null	Number of rows deleted.
RowsUpdated	int	Counter, null	Number of updates.
OptSelectCount	int	Counter, null	Number of times object was selected for plan during compilation.
LastOptSelectDate	datetime	Null	Last date the index was selected for plan during compilation.
UsedCount	int	Counter, null	Number of times the object was used in a plan during execution.
LastUsedDate	datetime	Null	Last date the index was used in a plan during execution.

Note Because you can use the plan for a stored procedure or trigger multiple times, the value of the OptSelectCount column may be less than the value of UsedCount. In addition, because the Adaptive Server may decide not to execute certain portions of a query plan during execution, the UsedCount may be less than the OptSelectCount.

monProcedureCache

Description

Returns statistics relating to Adaptive Server procedure cache.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
Requests	int	Counter, reset	Number of stored procedures requested
Loads	int	Counter, reset	Number of stored procedures loaded into cache
Writes	int	Counter, reset	Number of times a procedure was normalized and the tree written back to sysprocedures

Name	Datatype	Attributes	Description
Stalls	int	Counter, reset	Number of times a process had to wait for a free procedure cache buffer when installing a stored procedure into cache

monProcedureCacheMemoryUsage

Description

Includes one row for each procedure cache allocator. An allocator is identified by an allocator ID, which is internal to Adaptive Server.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
Allocator ID	int		Allocator ID
ModuleID	int		Module ID (internal to Adaptive Server)
Active	int		Number of memory pages (2KB) currently allocated to this allocator
HWM	int		Maximum number of memory pages allocated since the server was started
ChunkHWM	int		Largest number of contiguous memory pages allocated since the server was started
AllocatorName	varchar(30)		Name of the allocator
NumReuseCaused	int	Null	Number of times this allocator has caused replacement

monProcedureCacheModuleUsage

Description

Includes one row for each module that allocates memory from procedure cache. A module, which is identified with a module ID, is a functional area classification internal to Adaptive Server procedure cache management.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
ModuleID	int		A module ID
Active	int		Number of memory pages (2KB) currently allocated to this module
HWM	int		The maximum number of memory pages allocated since the server was started
ModuleName	int		Name of the module
NumPagesReuse	varchar(30)	Null	Number of pages allocated to this module

monProcess

Description

Provides detailed statistics about processes that are currently executing or waiting.

Note Enable the enable monitoring and wait event timing configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
BatchID	int		Unique identifier for the SQL batch containing the executing statement

Name	Datatype	Attributes	Description
ContextID	int		A unique identifier generated each time an executing query causes a stored procedure, trigger, execute immediate, deferred compilation, or other compiled object execution to occur
LineNumber	int		Line number of the current statement within the SQL batch
SecondsConnected	int		Number of seconds since this connection was established
DBID	int		Unique identifier for the database used by the process
EngineNumber	smallint		Unique identifier of the engine on which the process is executing
Priority	int		Priority at which the process is executing
FamilyID	int	Null	spid of the parent process, if this is a worker process
Login	varchar(30)	Null	Login user name
Application	varchar(30)	Null	Application name. May be blank if the application did not set a name in its login structure.
Command	varchar(30)	Null	Category of process or command the process is currently executing
NumChildren	int	Null	Number of child processes, if executing a parallel query
SecondsWaiting	int	Null	Amount of time, in seconds, the process has been waiting, if the process is currently blocked by a lock held by another process.
WaitEventID	int	Null	Unique identifier for the event for which the process is waiting, if the process is currently in a wait state.

Name	Datatype	Attributes	Description
BlockingSPID	int	Null	Session process identifier of the process holding the lock this process requested, if waiting for a lock
BlockingXLOID	int	Null	Unique lock identifier for the lock that this process has requested, if waiting for a lock
DBName	varchar(30)	Null	Name of the database the process is currently using
EngineGroupName	varchar(30)	Null	Engine group for the process
ExecutionClass	varchar(30)	Null	Execution class for the process
MasterTransactionID	varchar(255)	Null	Name of the transaction the process has open

monProcessActivity

Description

Provides detailed statistics about process activity.

Note Enable the enable monitoring, object lockwait timing, and wait event timing configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier.
KPID	int		Kernel process identifier.
ServerUserID	int		Server user identifier (SUID) of the user running this process. The value in ServerUserID matches the syslogins.suid column. Use the suser_name function to obtain the corresponding name.
CPUTime	int	Counter	CPU time (in milliseconds) used by the process.
WaitTime	int	Counter	Time (in milliseconds) the process spent waiting.

Name	Datatype	Attributes	Description
PhysicalReads	int	Counter	Number of buffers read from disk.
LogicalReads	int	Counter	Number of buffers read from cache.
PagesRead	int	Counter	Number of pages read.
PhysicalWrites	int	Counter	Number of buffers written to disk.
PagesWritten	int	Counter	Number of pages written.
MemUsageKB	int		Amount of memory (in bytes) allocated to the process.
LocksHeld	int		Number of locks process currently holds.
TableAccesses	int	Counter	Number of pages read that Adaptive Server retrieved without using an index.
IndexAccesses	int)	Counter	Number of pages read that Adaptive Server retrieved using an index.
WorkTables	int	Counter	Total number of work tables the process created.
TempDbObjects	int	Counter	Total number of temporary tables the process created.
ULCBytesWritten	int	Counter	Number of bytes written to the user log cache for the process.
ULCFlushes	int	Counter	Total number of times the user log cache was flushed.
ULCFlushFull	int	Counter	Number of times the user log cache was flushed because it was full.
ULCMaxUsage	int		The maximum usage (in bytes) of the user log cache by the process.
ULCCurrentUsage	int		The current usage (in bytes) of the user log cache by the process.
Transactions	int	Counter	Number of transactions started by the process.
Commits	int	Counter	Number of transactions committed by the process.
Rollbacks	int	Counter	Number of transactions rolled back by the process.

monProcessLookup

Description

Provides identifying information about each process on the server. See "monProcessActivity" on page 61 for statistics about the activity of each process.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
Login	varchar(30)	Null	Login user name
Application	varchar(30)	Null	Application name
ClientHost	varchar(30)	Null	Host name of client
ClientIP	varchar(24)	Null	IP address of client
ClientOSPID	varchar(30)	Null	Client application's operating system process identifier

monProcessNetIO

Description

Provides the network I/O activity information for each process.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
NetworkPacketSize	int		Network packet size the session is currently using.
PacketSent	int	Counter	Number of packets sent
PacketsReceived	int	Counter	Number of packets received
BytesSent	int	Counter	Number of bytes sent
BytesRecieved	int	Counter	Number of bytes received

monProcessObject

Description

Provides statistical information regarding objects currently being accessed by processes.

Note Enable the enable monitoring and per object statistics active configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
DBID	int		Unique identifier for the database in which the object resides
ObjectID	int		Unique identifier for the object
PartitionID	int		Unique identifier for the partition
IndexID	int		Unique identifier for the index
OwnerUserID	int		User identifier for the object owner
LogicalReads	int	Counter	Number of buffers read from cache
PhysicalReads	int	Counter	Number of buffers read from disk
PhysicalAPFReads	int	Counter	Number of asynchronous prefetch buffers read from disk
DBName	varchar(30)	Null	Name of database
ObjectName	varchar(30)	Null	Name of the object
PartitionName	varchar(30)	Null	Name of the partition
ObjectType	varchar(30)	Null	Type of object
PartitionSize	int	Counter, null	Partition size in kilobytes

monProcessProcedures

Description

Returns a list of all procedures being executed by processes..

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
DBID	int		Unique identifier for object's database
OwnerUID	int		Unique identifier for the object owner
ObjectID	int		Unique identifier for the procedure
PlanID	int		Unique identifier for the query plan
MemUsageKB	int		Amount of memory, in KB, used by the procedure
CompileDate	datetime		Date that the procedure was compiled
ContextID	int		A unique identifier generated each time an executing query causes a stored procedure, trigger, execute immediate, deferred compilation, or other compiled object execution to occur
LineNumber	int		The line in the procedure currently being executed
DBName	varchar(30)	Null	Name of the database that contains the procedure
OwnerName	varchar(30)	Null	Name of the owner of the object
ObjectName	varchar(30)	Null	Name of the procedure
ObjectType	varchar(32)	Null	The type of procedure (for example, stored procedure or trigger)

monProcessSQLText

Description

Provides the SQL text currently being executed by the process. Use max SQL text monitored to tune the maximum size of the SQL text.

monProcessSQLText returns a row for each row of the SQL text batch a process executes (specified by SPID). That is, if a batch contains three rows, monProcessSQLText returns three rows in its result set. The value for LineNumber indicates the number of the line in the batch. If the length of a single row exceeds 255 bytes, monProcessSQLText returns multiple rows and the value for LineNumber is the same for all rows, but the value for SequenceInLine is different for each row.

Note Enable the enable monitoring, SQL batch capture, and max SQL text monitored configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier.
KPID	int		Kernel process identifier.
ServerUserID	int		Server user identifier (SUID) of the user executing this SQL. The ServerUserID matches the value for the syslogins.suid column. Use the suser_name function to obtain the corresponding name.
BatchID	int		Unique identifier for the SQL batch containing the SQL text.
LineNumber	int		SQL batch line number for the row's SQL text.
SequenceInLine	int		Each row has a unique, and increasing, SequenceInLine value. If the length of the SQL text exceeds 255 bytes, the text is split over multiple rows.
SQLText	varchar(255)	Null	The text being executed.

monProcessStatement

Description

Provides information about the statement currently executing.

Note Enable the enable monitoring, statement statistics active, and wait event timing configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier.
KPID	int		Kernel process identifier.
DBID	int		Unique identifier for the database currently being used by the process.
ProcedureID	int		Unique identifier for the stored procedure.
PlanID	int		Unique identifier for the plan the process is executing.
BatchID	int		The batch number for the process in which the statement is executed.
ContextID	int		The stack frame of the procedure, if a procedure.
LineNumber	int		Line number of the statement within the SQL batch.
CPUTime	int	Counter	CPU time, in milliseconds, used by the statement.
WaitTime	int	Counter	Amount of time, in milliseconds, the task has waited while the statement executes.
MemUsageKB	int		Number of kilobytes of memory used for execution of the statement.
PhysicalReads	int	Counter	Number of buffers read from disk.
LogicalReads	int	Counter	Number of buffers read from cache.
PagesModified	int	Counter	Number of pages modified by the statement.
PacketsSent	int	Counter	Number of network packets sent by Adaptive Server.

Name	Datatype	Attributes	Description
PacketsReceived	int	Counter	Number of network packets received by Adaptive Server.
NetworkPacketSize	int		Size, in bytes, of the network packet currently configured for the session.
PlansAltered	int	Counter	Number of plans altered at execution time.
RowsAffected	int		Number of rows affected by the current statement. Queries using an inefficient query plan likely show a high number of logical I/Os per returned row.
StartTime	datetime	Null	Date when the statement began executing.

monProcessWaits

Description

Provides a list of all wait events for which current processes on the server are waiting. Returns only wait events whose Waits value is greater than zero.

Note Enable the enable monitoring, process wait events, and wait event timing configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attribute	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
WaitEventID	smallint		Unique identifier for the wait event
Waits	int	Counter	Number of times the process has waited for the event
WaitTime	int	Counter	Amount of time, in milliseconds, that the process has waited for the event

WaitEventInfo contains descriptions of each wait event. Join the WaitEventID column from each monitor table to view this data.

See Chapter 3, "Wait Events," for a descriptions of select wait events.

monProcessWorkerThread

Description

Provides statistics for the activity of each currently configured worker process.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attribute	Description
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
ThreadsActive	int		Number of worker threads currently in use by the process
MaxParallelDegree	smallint		The maximum degree of parallelism this task can use, which is set with the set parallel_degree option for the session, or the current Run Value for max parallel degree.
MaxScanParallelDegree	smallint		The maximum degree of parallelism for scans this task can use, which is set with set scan_parallel_degree for the session, or if this is not set, the current Run Value for max scan parallel degree.
ParallelQueries	int	Counter	Total number of parallel queries performed by this process
PlansAltered	int	Counter	Number of plans altered from "optimal" for the process. Plans are altered if Adaptive Server has an insufficient number of worker threads available to execute the query with an optimal degree of parallelism.
FamilyID	int	Null	The spid of the parent process, if this is a worker process

monState

Description

Provides information regarding the overall state of Adaptive Server.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
LockWaitThreshold	int		Time (in seconds) that a process must wait for a lock before it is counted as blocked and reported in the LockWaits column. The default value for LockWaitThreshold is 5 seconds. The default is used if you do not specify a value in the where clause of the query (for example `LockWaitThreshold=30`).
LockWaits	int		Number of process that have waiting for a lock longer than the value of LockWaitThreshold.
DaysRunning	int		Number of days Adaptive Server has been running.
CheckPoints	int		Specifies if any checkpoint is currently running.
NumDeadlocks	int	Counter	Total number of deadlocks that have occurred.
Diagnostic Dumps	int		Specifies if a shared memory dump is currently in progress for this server.
Connections	int		Number of active inbound connections.
MaxRecovery	int		The maximum time (in minutes), per database, that Adaptive Server uses to complete its recovery procedures in case of a system failure; also, the current Run Value for the recovery interval in minutes configuration option.
StartDate	datetime		Date and time Adaptive Server was started.

Name	Datatype	Attributes	Description
CountersCleared	datetime		Date and time the monitor counters were last cleared.

monStatementCache

Description

monStatementCache provides statistical information about the statement cache. You must enable the statement cache before monStatementCache table can collect data.

Note Enable the enable monitoring and enable stmt cache montoring configuration parameters and set the statement cache size parameter greater than 0 for this monitoring table to collect data.

Columns

Name	Type	Attributes	Description
TotalSizeKB	int		Configured size, in KB, of the statement cache.
UsedSizeKB	int		Amount of the statement cache, in KB, currently in use.
NumStatements	int		Number of statements in the statement cache.
NumSearches	int	Counter, reset	Number of times the statement cache was searched.
HitCount	int	Counter, reset	Number of times the statement cache was searched and a match was found.
NumInserts	int	Counter, reset	Number of statements that were inserted into the statement cache.

Name	Type	Attributes	Description
NumRemovals	int	Counter, reset	Number of times statements were removed from the statement cache. This value includes statements that were removed with explicit purges or from a replacement strategy.
NumRecompilesSchemaChanges	int	Counter, reset	Number of recompiles due to schema changes in the tables referred to in the cached statements.
NumRecompilesPlanFlushes	int	Counter, reset	Number of recompiles due to the plan flushes from the cache.

monSysLoad

Description

Provides trended statistics on a per-engine basis. You need not have the mon_role role to query this monitor table. This monitoring table is available only for Adaptive Server Cluster Edition.

There is one row per engine per statistic, with the exception of kernel run queue length, which is reported only for engine number 0.

Averages are computed using an algorithm that eliminates momentary peaks and valleys and provides a an indication of overall trends.

Note You need not enable any monitoring configuration parameters to use monSysLoad. These statics are always available and do not conflict with other monitoring tools, such as sp_sysmon.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster.
EngineNumber	smallint	Engine to which this row belongs.

Name	Datatype	Description
SteadyState	real	Average value for this statistic since Adaptive Server started.
Avg_1min	real	One-minute moving average for this statistic.
Avg_5min	real	Five-minute moving average for this statistic.
Avg_15min	real	Fifteen-minute moving average for this statistic.
Max_1min	real	Maximum one-minute average since start-up.
Max_5min	real	Maximum five-minute average since start-up.
Max_15min	real	Maximum fifteen-minute average since start-up.
Max_1min_Time	datetime	*datetime* at which Max_1min occurred.
Max_5min_Time	datetime	*datetime* at which Max_5min occurred.
Max_15min_Time	datetime	*datetime* at which Max_15min occurred.
Statistic		Name of the statistic this row represents: • Percent CPU busy • Percent I/O busy • Run queue length • Kernel run queue length • Outstanding disk I/Os • Disk I/Os per second • Network I/Os per second
Sample	float	Value of the metric at the last sample interval (that is, the current value of the metric).
Peak	float	The highest Sample value since the instance started (that is, the peak Sample value).
Peak_time	datetime	The date and time the Peak value was achieved.
StatisticID	int	A fixed identifier for this statistic. You may want to write applications to the fixed StatisticID instead of the localized Statistic name.

monSysPlanText

Description

Provides the history of the query plans for recently executed queries.

monSysPlanText returns one row of text from each line of the running query plans (similar to what is returned sp_showplan or by set showplan on). To make sure monSysPlanText reads the query plan text in the correct sequence, order the query result by SequenceNumber. For queries returning data for multiple queries or processes, order the query result by SPID, KPID, BatchID, SequenceNumber.

Note Enable the enable monitoring, plan text pipe active, and plan text pipe max messages configuration parameters for this monitoring table to collect data.

monSysPlanText is a historical monitoring table. See "Stateful historical monitoring tables" on page 11.

Columns

Name	Datatype	Attributes	Description
PlanID	int		Unique identifier for the plan
SPID	smallint		Session process identifier
KPID	int		Kernel process identifier
BatchID	int		Unique identifier for the SQL batch for which the plan was created
ContextID	int		The stack frame of the procedure, if a procedure
SequenceNumber	int		A monotonically increasing number indicating the position of the PlanText column within the entire plan text
DBID	int		Unique identifier for the database where the procedure is stored, if the plan is for a stored procedure
ProcedureID	int		Unique identifier for the procedure, if the plan is for a stored procedure
PlanText	varchar(160)	Null	Plan text output

Note Typically, there are multiple rows in this table for each query plan. Arrange the rows by sorting on the SequenceNumber column in ascending order.

monSysSQLText

Description

Provides the most recently executed SQL text, or the SQL text currently executing. The maximum number of rows returned can be tuned with sql text pipe max messages.

Note Enable the enable monitoring, SQL text pipe active, max SQL text monitored, and SQL text pipe max messeges configuration parameters for this monitoring table to collect data.

monSysSQLText is a historical monitoring table. See "Stateful historical monitoring tables" on page 11.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier.
KPID	int		Kernel process identifier.
ServerUserID	int		Server user identifier (SUID) of the user who executed this SQL text. The ServerUserID matches the value in syslogins.suid. Use the suser_name function to obtain the corresponding name.
BatchID	int		Unique identifier for the SQL batch containing the SQL text.
SequenceInBatch	int		Indicates the position of this portion of SQL text within a batch (the SQL text for a batch may span multiple rows).
SQLText	varchar(255)	Null	SQL text.

Note In many cases the text for a query spans multiple rows in this table. Arrange rows in proper order by sorting on the SequenceInBatch column in ascending order.

monSysStatement

Description

Provides a history of the most recently executed statements on the server. Use statement pipe max messages to tune the maximum number of statement statistics returned.

Note Enable the enable monitoring, per object statistics active, statement pipe active, statement pipe max messages, and statement statistics active configuration parameters for this monitoring table to collect data.

monSysStatement is a historical monitoring table. See "Stateful historical monitoring tables" on page 11.

Columns

Name	Datatype	Attributes	Description
SPID	smallint		Session process identifier.
KPID	int		Kernel process identifier.
DBID	int		Unique identifier for the database.
ProcedureID	int		Unique identifier for the procedure.
PlanID	int		Unique identifier for the stored plan for the procedure.
BatchID	int		Unique identifier for the SQL batch containing the statement.
ContextID	int		The stack frame of the procedure, if a procedure.
ErrorStatus	int		The error return status of the statement.
LineNumber	int		Line number of the statement within the SQL batch.
CPUTime	int	Counter	Number of milliseconds of CPU used by the statement.
WaitTime	int	Counter	Number of milliseconds the task has waited during execution of the statement.
MemUsageKB	int		Number of kilobytes of memory used for execution of the statement.
PhysicalReads	int	Counter	Number of buffers read from disk.
LogicalReads	int	Counter	Number of buffers read from cache.
PagesModified	int	Counter	Number of pages modified by the statement.
PacketSent	int	Counter	Number of network packets sent by Adaptive Server.
PacketsReceived	int	Counter	Number of network packets received by Adaptive Server.
NetworkPacketSize	int		Size (in bytes) of the network packet currently configured for the session.
PlansAltered	int	Counter	The number of plans altered at execution time.

Name	Datatype	Attributes	Description
RowsAffected	int		Number of rows affected by the current statement. Queries using an inefficient query plan likely show a high number of logical I/Os per returned row.
StartTime	datetime	Null	Date the statement began execution.
EndTime	datetime	Null	Date the statement finished execution.
ProcNestLevel	int		Nesting level of the stored procedure in which this statement executed. The value for ProcNestLevel is 0 (zero) for statements within SQL batches.
StatementNumber	int		Number indicating the order in which this statement was executed within the SQL batch for the process.

monSysWaits

Description

Provides a server-wide view of the statistics for events on which processes have waited.

Note Enable the enable monitoring and wait event timing configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
WaitEventID	smallint		Unique identifier for the wait event
WaitTime	int	Counter	Amount of time (in seconds) tasks spent waiting for the event
Waits	int		Number of times tasks waited for the event

See Chapter 3, "Wait Events," for more information

You can join the monSysWaits table with monWaitEventInfo using the WaitEventID columns as the join column to obtain the wait event descriptions. For example:

```
select w.Waits, w.WaitTime, w.WaitEventID, i.Description
from master..monSysWaits w, master..monWaitEventInfo i
where w.WaitEventID = i.WaitEventID
```

monSysWorkerThread

Description

Returns server-wide statistics related to worker thread configuration and execution.

Note Enable the enable monitoring configuration parameter for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
ThreadsActive	int		Number of worker processes currently active
TotalWorkerThreads	int		Maximum number of worker processes (configured by setting number of worker processes)
HighWater	int	reset	The maximum number of worker processes that have ever been in use
ParallelQueries	int	Counter, reset	Number of parallel queries attempted
PlansAltered	int	Counter, reset	Number of plans altered due to unavailable worker processes
WorkerMemory	int		The amount of memory currently in use by worker processes
TotalWorkerMemory	int		The amount of memory configured for use by worker processes

Name	Datatype	Attributes	Description
WorkerMemoryHWM	int	reset	The maximum amount of memory ever used by worker processes
MaxParallelDegree	int		The maximum degree of parallelism that can be used: the current Run Value for max parallel degree configuration option
MaxScanParallelDegree	int		The maximum degree of parallelism that can be used for a scan: the current Run Value for max scan parallel degree configuration option

monTableColumns

Description

Describes all the columns for each monitoring table. monTableColumns helps determine what columns are in the monitoring tables. You can join monTableColumns with monTables to report columns and column attributes for the monitoring tables.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
TableID	int		Unique identifier for the view
ColumnID	int		Position of the column
TypeID	int		Identifier for the datatype of the column
Precision	tinyint		Precision of the column, if numeric
Scale	tinyint		Scale of the column, if numeric
Length	smallint		Maximum length of the column (in bytes)

Name	Datatype	Attributes	Description
Indicators	int		Indicators for specific column properties (for example, if the column is prone to wrapping and should be sampled)[1]
TableName	varchar(30)	Null	Name of the table
ColumnName	varchar(30)	Null	Name of the column
TypeName	varchar(20)	Null	Name of the datatype of the column
Description	varchar(255)	Null	Description of the column

[1]The Indicators column is a bitmap. Use a bit mask to to determine which bits are turned on. Possible values are:

- 1 – the value for Indicators may increase rapidly and lead to counter wrapping if values reach 2^{32}, which can occur to columns that have the number 1 bit in the Indicators column value turned on. To determine whether the 1 bit is turned on, use:

```
select TableName, ColumnName
from Master..monTableColumns
where Indicators & 1 != 0
```

- 2 – the counter is shared with sp_sysmon and is reset if you execute sp_sysmon. . .clear.

  To display all columns sp_sysmon clears with the clear parameter, use:

```
Select TableName, ColumnName
from master..monTableColumns
where Indicators & 2 != 0
```

monTableParameters

Description

Provides a description for all columns in a monitoring table used to optimize query performance for the monitoring tables.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
TableID	int		Unique identifier for the table
ParameterID	int		Position of the parameter
TypeID	int		Identifier of the datatype of the parameter
Precision	tiny_int		Precision of the parameter, if numeric
Scale	tiny_int		Scale of the parameter, if numeric
Length	small_int		Maximum length of the parameter (in bytes)
TableName	varchar(30)	Null	Name of the table
ParameterName	varchar(30)	Null	Name of the parameter
TypeName	varchar(20)	Null	Name of the data type of the parameter
Description	varchar(255)	Null	Description of the parameter

monTables

Description

Provides a description of all monitoring tables. You can join monTables with monTableColumns for a description of each monitoring table and the columns it contains.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
TableID	int		Unique identifier for the table
Columns	tinyint		Total number of columns in the table
Parameters	tinyint		Total number of optional parameters you can specify
Indicators	int		Indicators for specific table properties (for example, if the table retains session context)[1]
Size	int		Maximum row size (in bytes)
TableName	varchar(30)	Null	Table name

Name	Datatype	Attributes	Description
Description	varchar(368)	Null	Table description

[1]The Indicators column is a bit map. Use a bitmask to to determine which bits are turned on. A value of 1 indicates the table is a historical table.

To display all tables that are historical:

```
Select TableName
from master..monTables
where Indicators & 1 != 0
```

monTempdbActivity

Description

Provides statistics for all open local temporary databases, including global system tempdb when the instance is started in tempdb configuration mode. This monitoring table is available only for Adaptive Server Cluster Edition.

Note Set the enable monitoring, per object statistics active, and object lockwait timing configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
DBID	int	Unique identifier for the database
InstanceID	tinyint	ID of the instance within the cluster
DBName	varchar(30)	Name of the database
AppendLogRequest	int	Number of semaphore requests from an instance attempting to append to the database transaction log
AppendLogWaits	int	Number of times a task waits for the append log semaphore to be granted
LogicalReads	int	Total number of buffers read
PhysicalReads	int	Number of buffers read from disk
APFReads	int	Number of asynchronous prefetch (APF) buffers read
PagesRead	int	Total number of pages read
PhysicalWrites	int	Total number of buffers written to disk
PagesWritten	int	Total number of pages written to disk

Name	Datatype	Description
LockRequests	int	Number of requests for a object lock in this temporary database
LockWaits	int	Number of times a task waited for an object lock in this temporary database
CatLockRequests	int	Number of requests for a lock on the system catalog
CatLockWaits	int	Number of times a task waited for a lock for system table
AssignedCnt	int	Number of times this temporary database was assigned to a user task
SharableTabCnt	int	Number of sharable tables created

monWaitClassInfo

Description

Provides a textual description for all of the wait classes (for example, waiting for a disk read to complete). All wait events (see the description for monWaitEventInfo) have been grouped into wait classes that classify the type of event for which a process is waiting.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
WaitClassID	smallint		Unique identifier for the wait event class
Description	varchar(50)	Null	Description of the wait event class

monWaitEventInfo

Description

Provides a textual description for every possible situation where a process is forced to wait within Adaptive Server.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Attributes	Description
WaitEventID	smallint		Unique identifier for the wait event type
WaitClassID	smallint		Unique identifier for the wait event class
Description	varchar(50)	Null	Description of the wait event type

Join monWaitEventInfo with monProcessWaits or monSysWaits on the WaitEventID column to obtain the wait event descriptions listed in those tables.

monWorkload

Description

Displays the workload score for each logical cluster on each instance according to its load profile. This monitoring table is available only for Adaptive Server Cluster Edition.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster
InstanceName	varchar(30)	Instance name
LCID	tinyint	Logical cluster ID
LogicalClusterName	varchar(30)	Logical cluster name
LoadProfile	varchar(30)	Name of the load profile used to generate the load score
LoadScore	int	Load score for this instance or logical cluster

Name	Datatype	Description
ConnectionsScore	float	Weighted value for the `user connections` metric
CpuScore	float	Weighted value for the `cpu utilization` metric
RunQueueScore	float	Weighted value for the `run queue` metric
IoLoadScore	float	Weighted value for the `io load` metric
EngineScore	float	Weighted value for the `engine deficit` metric
UserScore	float	Weighted value for the `user` metric
LoginRedirections	int	Number of load-based login redirections this logical cluster performs on this instance
DynamicMigratoins	int	Number of load-based connection migrations (dynamic load distribution) this logical cluster performs on this instance

monWorkloadPreview

Description

Provides an estimate of how a load profile impacts the workload score without enabling the profile. monWorkload includes one row for each logical cluster and instance on which this logical cluster is running. The load score and components are based on the current profile for that logical cluster. The monWorkloadPreview table has one row for each combination of instance and load profile configured on the system, allowing the administrator to see how workload scoring would be done for each profile. You need not have the mon_role role to query this monitor table.

This monitoring table is available only for Adaptive Server Cluster Edition.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster
InstanceName	varchar(30)	Instance name
LoadProfileID	smallint	Load profile ID

Name	Datatype	Description
LoadProfile	varchar(30)	Name of load profile used to generate the load score
LoadScore	int	Load score for this instance or logical cluster
ConnectionScore	float	Weighted value for the `user connections` metric
CpuScore	float	Weighted value for the `cpu utilization` metric
RunQueueScore	float	Weighted value for the `run queue` metric
IoLoadScore	float	Weighted value for the `io load` metric
EngineScore	float	Weighted value for the `engine deficit` metric
UserScore	float	Weighted value for the `user` metric

monWorkloadProfile

Description

Displays currently configured workload profiles. You need not have the mon_role role to query this monitor table. This monitoring table is available only for Adaptive Server Cluster Edition.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
ProfileID	smallint	Workload profile ID
Name	varchar(30)	Workload profile name
RefreshInterval	varchar(15)	Frequency of load refresh, can be a time in the form *hh:mm:ss* or `DISABLED` or `AUTO`
ConnectionsWeight	tinyint	Weight associated with the `active connections` metric
CpuWeight	tinyint	Weight associated with the `cpu utilization` metric
RunQueueWeight	tinyint	Weight associated with the `run queue` metric

Name	Datatype	Description
IoLoadWeight	tinyint	Weight associated with the `io load` metric
EngineWeight	tinyint	Weight associated with the `engine deficit` metric
UserWeight	tinyint	Weight associated with the `user metric` metric
LoginWeight	smallint	Threshold for load-based login redirection
Dynamicthreshold	smallint	Threshold for dynamic load distribution (that is, post-login migration for load purposes)
Type		System or user profile.
LoginThreshold		Threshold for the login load distribution.
Hysteresis		Minimum load score that enables redirection.

monWorkloadRaw

Description

Provides the raw workload statistics for each instance. You need not have the mon_role role to query this monitor table. This monitoring table is available only for Adaptive Server Cluster Edition.

Note You need not enable any configuration parameters for this monitoring table to collect data.

Columns

Name	Datatype	Description
InstanceID	tinyint	ID of the instance within the cluster
InstanceName	varchar(30)	Instance name
ConnectionsRaw	float	Raw value for the `user connections` metric
CpuRaw	float	Raw value for the `cpu utilization` metric
RunQueueRaw	float	Raw value for the `run queue` metric
IoLoadRaw	float	Raw value for the `io load` metric

Name	Datatype	Description
EngineRaw	float	Raw value for the `engine deficit` metric
UserRaw	float	Raw value for the `user` metric

CHAPTER 3

Wait Events

Adaptive Server task management includes three states for a process: running, runnable, sleeping, and blocked. When a process is not running (executing on the CPU), it is:

- Waiting on the CPU (the “runnable” state)
- Sleeping because of disk or network I/O
- Blocked on a resource (a lock, semaphore, spinlock, and so on)

A wait event occurs when a server process suspends itself, sleeps, and waits for another event to wake it. Adaptive Server includes unique wait event IDs for each of these wait events. Query monSysWaits and monProcessWaits to find the number of times—and the total amount of time—that a process waited for each wait event.

Note The value of WaitTime in the monSysWaits table is in seconds. The value of the WaitTime in the monProcessWaits table is in milliseconds.

This chapter describes a selection of the more common wait events and actions you can perform to avoid them.

Event 19: xact coord: pause during idle loop

The Adaptive Server transaction coordinator (ASTC) sleeps, waiting for an alarm or a server task to wake it (ASTC handles transactions involving multiple database servers). If the server does not perform many distributed transactions, the time per wait for this event is close to 60 seconds.

Action

No action necessary. Even with high values for WaitTime, event 19 does not affect overall performance.

Event 29: waiting for regular buffer read to complete

A wait caused by a physical read (most likely a cache miss) which occurs when Adaptive Server does not find a page in the data cache and must read it from disk. The number of Waits is the number of physical reads that occurred because of a cache miss. Use the monSysWaits.WaitTime value to derive I/O response times

Action

Because this event's value for monSysWaits.WaitTime is measured in seconds, the value for WaitTime for this event should be much less than the value for Waits (an average physical read should be 2–6 milliseconds; more than 10 milliseconds is considered slow). A high average physical read value may indicate poor disk throughput performance. Query monIOQueue and monDeviceIO to identify slow or overloaded disks.

A high value for Waits, regardless of the value for WaitTime, may indicate that query plans are not as effective as they could be. If you encounter a high value for Waits, a table scan or Cartesian product may have occurred, or the optimizer may have selected a bad plan, due to bad, stale, or missing statistics. Consider adding an index on specific columns to the table on which this occurred.

A high value for Waits can also indicate the data caches are too small, with active pages first pushed out and then reread. Query monOpenObjectActivity, monProcessActivity, monDataCache, monCachPool, and monProcessObject to determine how to proceed.

Event 30: wait to write MASS while MASS is changing

Adaptive Server is attempting to write to a memory address space segment (MASS). A MASS is one or more contiguous pages Adaptive Server keeps in a data cache. However, in this event, the status of the MASS is "changing," meaning another spid is updating the MASS. The spid initiating the write cannot write to the MASS until the MASS is no longer in use.

A high value for WaitTime for event 30 may indicate that the data cache is too small, causing pages in the data cache to reach the wash area frequently, forcing the checkpoint process to perform more writes than necessary.

Action

You may be able to reduce high wait times by:

- Increasing the size of the data cache
- Using cache partitions or named caches to separate memory-intensive objects

- Tuning the housekeeper, washmarker position, or schema implications (such as sequential key tables)
- Positioning the washmarker
- Adjusting the schema (such as sequential key tables)

Event 31: waiting for buf write to complete before writing

A server process responsible for writing data pages to the disk (for example, a checkpoint) has determined that it must write a MASS. However, an earlier write operation involving the same page has not finished, so the second process must wait until the first write completes before initiating its write operation.

Action

Generally, the value for WaitTime for event 31 should be less than the value for Waits. High values for WaitTime may indicate disk contention or slow performance. Query monIOQueue and monDeviceIO to identify overloaded or slow disks.

A high value for WaitTime for event 31 may also indicate that the data cache is too small, causing pages in the data cache to reach the wash area frequently and forcing the checkpoint process to perform more writes than necessary.

Event 32: waiting for an APF buffer read to complete

When Adaptive Server issues an asynchronous prefetch (APF) on a page, another process is reading the MASS to which this page belongs. Adaptive Server must wait for the read to complete before continuing.

Action

A high value for Waits may indicate that Adaptive Server is using asynchronous prefetch too often. Tuning the local APF limit for cache pools may reduce contention for APF pages.

Since Adaptive Server often uses APF for table scans, contention involving APF reads may indicate that an application is performing too many table scans because of factors such as missing indexes.

Event 35: waiting for buffer validation to complete

Indicates that a process is attempting to read data in a page that another process has read into cache. After reading a page into a data cache, Adaptive Server validates the success of the read operation. Because Adaptive Server is validating whether the read was successful, the second process must wait for this to complete before accessing the data.

This event commonly occurs during periods of high physical reads.

Action

The value for WaitTime for event 35 should be quite small. If the value is large, many processes are accessing the same page at the same time, or there is CPU contention. Query monEngine to determine if the engines are overloaded, and run system-level utilities to determine if there is overall CPU contention.

Event 36: waiting for MASS to finish writing before changing

A spid must make changes to a MASS, but another spid is currently writing the MASS. The second spid must wait until the write completes.

Action

A high value for WaitTime indicates that some condition may be causing diminished I/O or data cache manager performance. Normally, the value for Waits should be higher than the value for WaitTime. Query monIOQueue and monDeviceIO to determine if a disk device is slow or overloaded.

Note If event 36 occurs because of an update to a page, partitioning th cache has not effect. However, if event 36 occurs when page updates are not taking place, partitioning the cache may expedite the writes.

Event 37: wait for MASS to finish changing before changing

A spid attempts to make changes to the MASS, but another spid is currently changing the MASS. The first spid must wait until the changes are complete before it can make changes to the MASS.

Action

Typically, the values for Waits for event 37 should be much higher than the values for WaitTime. If the values are not higher for Waits, either many processes are accessing the same MASS at once, or there is CPU contention. Query monEngine to determine if the engines are overloaded. Run system-level utilities to determine if there is overall CPU contention.

Event 41: wait to acquire latch

Event 41 often indicates that multiple processes are simultaneously attempting to update rows on a single page.

Adaptive Server uses a latch as a transient lock to guarantee a page's contents are not changed while another process reads or writes data. Adaptive Server typically uses latches in data-only locked tables to protect the contents of the page when multiple processes are simultaneously reading or updating rows on the same page. If one process attempts to acquire a latch while another process already holds the latch, the first process may need to wait. If event 41 occurs frequently, it may indicate a high level of contention for data on a single physical page within an index or table.

Reduce contention by:

- Introducing an index with a sort order that distributes data differently across pages, so it spreads the rows that are causing contention
- Changing your application so that such contention does not occur

Action

Consider reducing contention for pages by changing index definitions in a way that alters the physical distribution of data across the data and index pages within your table, or modifying your application to reduce contention.

If the average value for WaitTime is high, event 41 may occur because of an Adaptive Server resource shortage, resulting from:

- A hash table that is too small for a lock, resulting in very long hash chains that Adaptive Server must search.
- An operating system issue during which calls that should be quick are a bottle neck (for example, starting asynchronous I/O, which should return immediately, blocks because of operating system resource limitations.
- Extremely high inserts and expanding updates. Page allocations take place frequently, and contention for the allocation page latch results in a high number of Waits. Use dbcc tune(des_greedyalloc) to reduce this contention. For information about latch contention, see *Performance and Tuning Series: Monitoring Adaptive Server with sp_sysmon*.

Event 46: wait for buf write to finish getting buf from LRU

A spid attempts to acquire a buffer from the least recently used (LRU) chain. However, the buffer has an outstanding write that must finish before Adaptive Server can use the buffer for a different page.

Action

Event 46 may indicate that:

- A cache is so busy that the buffer at the end of the LRU chain is still processing. Query monDataCache and monCachePool to determine which cache is busy. Possible resolutions include: increasing the size of the cache, using sp_poolconfig to increase the wash size, and increasing the housekeeper activity by retuning enable housekeeper GC.
- Disk writes are taking a long time to complete. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 51: waiting for last i/o on MASS to complete

Occurs when a process is writing a range of pages for an object to disk because of a change to the object or because the object is removed from the metadata cache. Because it is important to complete the I/O operations on some pages before other pages are written, the process must wait until it is notified that the I/O that was initiated has completed its task.

Action

A high value for WaitTime indicates that writes may be taking a long time to complete. Typically, the value for Waits should be much higher than the value for WaitTime. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 52: waiting for i/o on MASS initiated by another task

A process writes a range of pages for an object to disk because of a change to the object, or because the object was removed from the metadata cache. However, another spid has an I/O outstanding on the MASS, and the second process must sleep until the first process's finish writing.

Action

A high value for WaitTime for this event indicates that writes may be taking too long to complete. Typically, the value for Waits should be much higher than the value for WaitTime. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 53: waiting for MASS to finish changing to start i/o

A spid attempts to write to a MASS, but another spid is already changing the MASS, so the first spid must wait until the changes are complete.

Adaptive Server minimizes the number of disk I/O operations it performs. If a process responsible for writing pages (for example, the checkpoint process) needs to modify a page but determines that another process is modifying the page, the second process waits until the first process completes so the page write includes the page modification.

Action

Normally, the value for Waits for event 53 should be higher than the value for WaitTime. If it is not higher, either many processes are simultaneously accessing the same MASS, or there is CPU contention. Query monEngine to determine if the engines are overloaded. Run system-level utilities to determine if there is overall CPU contention.

Event 54: waiting for write of the last log page to complete

Event 54 occurs when a process is about to initiate a write of the last log page but discovers another process is already scheduled to perform write. The second process waits until the first process finishes its I/O, but the second process does not initiate the I/O operation.

Because Adaptive Server frequently updates the last page of the transaction log, Adaptive Server avoids performing physical writes of the last log page. This reduces the amount of I/O the server performs and increases the last log page's availability to other processes that need to perform an update, and thereby impoving performance.

Action

A high average value for WaitTime for event 54 indicates that writes are taking a long time to complete. Typically, the value for Waits should be much higher than the value for WaitTime. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

High values for Waits, regardless of the average time, may indicate contention for the last log page. Increase the size of the user log cache to reduce contention, or group operations for applications to avoid committing every row.

Event 55: wait for i/o to finish after writing last log page

Indicates a process has initiated a write operation on the last page of the transaction log, and must sleep until the I/O completes. A high value for the Waits column for event 55 indicates that Adaptive Server is making a large number of updates to the transaction log because of committed transactions or other operations that requiring writing the transasction log to disk.

Action

A high value for WaitTime for event 55 indicates that writes may be taking a long time to complete. Typically, the value for Waits should be much higher than the value for WaitTime

Event 57: checkpoint process idle loop

The checkpoint process sleeps between runs to prevent the checkpoint from monopolizing CPU time.

Action

Event 57 may accumulate large amounts of time since the checkpoint process starts when the server starts. However, you need not perform any actions based on this event.

Event 61: hk: pause for some time

The housekeeper pauses occasionally to keep housekeeper functions from monopolizing CPU time.

Action

Event 61 is expected, and may show large values on servers that have run for along time. Typically, you need not perform any actions based on this event.

Event 70: waiting for device semaphore

If you are using Adaptive Server mirroring (that is, disable disk mirroring is set to 0), each disk device access must first hold the semaphore for that device. Event 70 measures the time spent waiting for that semaphore and can occur if disk I/O structures are too low.

Action

If you are not using Adaptive Server mirroring, set disable disk mirroring to 1. If you are using mirroring, high values for WaitTime may indicate a loss of performance from device contention. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device. Evaluate the results to determine if you can shift some of the load to other devices.

Event 83: wait for DES state is changing

A object descriptor (called a "DES") is allocated for every open object (temporary tables, cached query plans and statement cache, stored procedures, triggers, defaults, rules, tables, and so on). Event 83 occurs when Adaptive Server is releasing an allocated descriptor, which typically happens when Adaptive Server is dropping an object.

Action

A high value for Waits for event 83 may indicate a shortage of object descriptors. You may need to increase the number of open objects.

Event 84: wait for checkpoint to complete

Adaptive Server is dropping a DES, which typically occurs when Adaptive Server is dropping an object. Event 84 indicates that the drop must wait for a checkpoint to complete on the database.

Action

Although it is unlikely that the Waits value is high for event 84, a high value may indicate that many drops are occurring simultaneously, or that the checkpoint process is taking a long time. If the checkpoints are running for an excessive amount of time, try decreasing the recovery interval (in minutes).

Event 85: wait for flusher to queue full DFLPIECE

When Adaptive Server runs dump database, it uses the "flusher" process to create lists of pages (which includes a structure called DFLPIECE) that are in a data cache and have been changed. Adaptive Server sends the Backup Server a list of pages to include in the dump.

Event 85 measures the time the dump process spends waiting for the flusher process to fill and queue DFLPIECE.

Action

This event is normal during a dump database. If the average value for WaitTime is exceptionally high (higher than 2), check other events to determine what is slowing down the flusher processes.

Event 91: waiting for disk buffer manager i/o to complete

When Adaptive Server runs load database, it may require the load process to verify that a disk I/O has completed before continuing. Event 91 measures the time Adaptive Server spends waiting for verification.

Action

Generally, the value for WaitTime for event 91 should be much lower than the value for Waits. High values for WaitTime indicate possible disk contention or slowness. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 99: wait for data from client

When a process uses a site handler to connect to a remote server, it must occasionally wait for the server to return the data. Event 99 measures the time the process must wait.

A site handler is a method for transmitting RPCs from a local server to a remote server. A site handler establishes a single physical connection between local and remote servers and multiple logical connections, as required by RPCs.

Action

A high average value for WaitTime for event 99 indicates slow communication with the remote server. This may be due to complex RPC calls that take a long time to complete, performance issues in the remote server, or a slow or overloaded network.

Event 104: wait until an engine has been offlined

Adaptive Server has a service process that cleans up issues after an engine goes offline. This process typically sleeps, waking up every 30 seconds to check for work to do. Event 10 4 measures that sleep time.

Action

The average value for WaitTime for event 104 should be very close to 30. If engines are frequently taken offline, this value may be slightly lower. If the average value for WaitTime is significantly higher or lower than 30, contact Sybase Technical Support.

Event 124: wait for mass read to finish when getting page

Event 124 occurs when a process attempts to perform a physical read but another process has already performed the read request (this also counts as a "cache miss").

Action

The value for WaitTime for event 124 should be much lower than the value for Waits. The average value for WaitTime is high if disk performance is poor. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 142: wait for logical connection to free up

When Adaptive Server executes an RPC on a remote serer using the site handler mechanism, it creates logical connections.

Event 142 occurs when Adaptive Server must close a logical connection but finds another process using it. Adaptive Server must wait until the logical connection is no longer in use before closing the logical connection.

Action

Event 142 should normally have a very low average value for WaitTime. A high value for WaitTime may indicate there is a problem communicating with the remote server.

Event 143: pause to synchronise with site manager

Adaptive Server communicates with a remote server using a site manager, but of another process is attempting to connect to that remote server. Event 143 measures the amount of time Adaptive Server waits to establish the connection to the remote server.

Action

A high average value for WaitTime for event 143 may indicate performance issues on the remote server or a slow or overloaded network. Query monProcessWaits for WaitEventID 143 to determine which spids have high wait times.

Event 150: waiting for a lock

A process attempts to obtain a logical lock on an object but another process is already holding a conflicting lock on this object. Event 150 is a common event that occurs when Adaptive Server performs an operation that requires locks to protect data that is being read or updated. The locks involved may be at various levels, including table, page, or row.

After all conflicting locks are released, Adaptive Server wakes the waiting process and grants it access to the object.

Action

The value for WaitTime for this event can be high if there is contention for a particular table or page (such as a high number of heap inserts). Query monLocks and monOpenObjectActivity to identify objects that are experiencing heavy lock contention.

In some situations, you can reduce the amount of lock contention by changing the table's locking scheme from allpages locking to data-only locking. Application or database design typically causes lock contention; evaluate your application design to determine the best method to reduce lock contention, while still considering other application requirements.

Event 157: wait for object to be returned to pool

The Adaptive Server memory manager allocates memory for storing data describing a wide range of internal objects from separate memory "pools." When a pool's available memory is low, requests for additional memory may be delayed until another operation returns memory to the pool. When this occurs, the requesting process must wait until more memory is available.

Event 157 occurs when a process most wait for memory to become available before allocating the object's data.

Action

If the average value for WaitTime for event 157 is low, performance may not noticeably degrade. However, any Waits on this event indicate a condition you can correct by increasing the configured number of structures for which Adaptive Server is waiting. Use sp_countmetadata and sp_monitorconfig to identify which structures are using the maximum configuration to determine which resources you should increase.

Event 169: wait for message

Some Adaptive Server processes (for example, worker threads, auditing, disk mirroring, and so on) use a structure called a "mailbox" to pass messages. Event 169 measures the time Adaptive Server spends waiting for a message in a mailbox.

Action

Typically, the average value for WaitTime for event 169 is very small. However, if the value for WaitTime is large, query monProcessWaits for rows with WaitEventID value of 169 to determine which jobs have long wait times for this event.

Event 171: wait for CTLIB event to complete

Indicates that Adaptive Server is waiting for the remote server to respond. Event 171 appears if you use Component Integration Services (CIS) for proxy tables and RPC calls.

Action

A high average value for WaitTime for this event may indicate remote CIS server performance issues or a slow or overloaded network. Query monProcessWaits for WaitEventID 171 to determine which spids have high wait times for this event.

Event 178: waiting while allocating new client socket

A network listener is an Adaptive Server process that handles a client's incoming connection requests. Event 178 measures the time Adaptive Server spends waiting for new connection requests.

Action

You need not perform any actions based on event 178. However, you can use some of its information for analysis. The value for WaitTime is roughly equivalent to the amount of time the server has been running. The values for Waits is a measure of how many connection attempts have been made since the server started.

Event 179: waiting while no network read or write is required

The Adaptive Server network task sleeps on event 179 if there is no network I/O the server must send or receive. When there is network activity, the server task wakes, handles the requests, and then goes back to sleep.

Action

High values for event 179 indicate high levels of network activity. If the network activity is unexpectedly high, query other monitoring tables—such as monNetworkIO and monProcessNetIO—to determine which jobs are slowing network performance.

A high value for the Waits column for event 179 may indicate that dbcc checkstorage identified a large number of possible consistency faults. Check the reports from dbcc checkstorage for more information.

Event 197: waiting for read to complete in parallel dbcc

When you run dbcc checkstorage, Adaptive Server must occasionally perform asynchronous I/O on the workspace to read or write a single reserved buffer. Event 197 measures the time Adaptive Server waits for those disk I/Os.

Action

Generally, the value for WaitTime for event 197 should be much lower than the value for Waits. A high average value for WaitTime may indicate poor disk throughput performance. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 200: waiting for page reads in parallel dbcc

Event 200 occurs when you run dbcc checkstorage using multiple worker processes. This event measures the time spent waiting for reads to complete on pages that dbcc checks.

Action

Generally, the value for WaitTime for event 200 should be much lower than the value for Waits. A high average value for WaitTime may indicate poor disk throughput performance. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 201: waiting for disk read in parallel dbcc

When you run dbcc checkverify, Adaptive Server performs a disk read to verify whether a potential fault exists in the disk copy of a page; event 201 measures the time spent waiting for those reads to complete.

Action

Generally, the value for WaitTime for event 201 should be much lower than the value for Waits. A high average value for WaitTime may indicate poor disk throughput. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 202: waiting to re-read page in parallel

When you run dbcc checkstorage, Adaptive Server determines whether it needs to perform a disk read to verify whether a potential fault exists in the disk copy of a page; event 202 measures the time spent waiting for those reads to complete.

Action

Generally, the value for WaitTime for event 202 should be much lower than the value for Waits. A high average value for WaitTime may indicate poor disk throughput. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 203: waiting on MASS_READING bit in parallel dbcc

When you run dbcc checkstorage, Adaptive Server determines whether it needs to perform a disk read to verify whether a fault exists in the disk copy of the MASS. However, another process may have already started that read. Event 203 measures the time spent waiting for those reads to complete.

Action

Generally, the value for WaitTime for event 203 should be much lower than the value for Waits. A high average value for WaitTime may indicate poor disk throughput. Query monIOQueue and monDeviceIO to determine if there is a slow or overloaded disk device.

Event 205: waiting on TPT lock in parallel dbcc

When you run dbcc checkstorage to check text and image pages, Adaptive Server must hold a lock to prevent multiple worker threads from accessing the page links at the same time. Event 205 measures the time spent waiting for those locks.

Action

The frequency of event 205 depends on how many text and image columns are contained in the tables you are checking. An exceptionally high average value for WaitTime may indicate some resource contention for the worker thread holding the lock. Check CPU and disk metrics to determine if there is contention.

Event 207: waiting sending fault msg to parent in PLL dbcc

When you run dbcc checkstorage, each worker process reports possible faults to the parent process by queuing messages to the parent spid. If the mailbox of the parent process is full, the worker process must wait for more room in the mailbox before it can queue the next message. Event 207 measures the time the worker process spends waiting.

Action

Event 207 is typically caused by Adaptive Server reporting a large number of faults. You need not take any actions for this event, other than to follow the normal process of running dbcc checkverify to verify and analyze the faults.

Event 209: waiting for a pipe buffer to read

When Adaptive Server performs a sort in parallel (for example, create index that specifies a `consumers` clause), it uses an internal mechanism to send data between the various tasks. Event 209 measures the amount of time the tasks spend waiting for other tasks to add data to a pipe.

Action

The average value for WaitTime for event 209 should be very low. High average values for WaitTime may indicate that the sort manager producer processes cannot generate data fast enough to keep the consumer processes busy. Check the overall system performance to determine if Adaptive Server has sufficient CPU and I\O bandwidth.

Event 210: waiting for free buffer in pipe manager

When Adaptive Server performs a sort in parallel (for example, create index that specifies a `consumers` clause), it uses an internal mechanism, called a pipe, to send data between the various tasks. Event 210 measures the amount of time a process waits for Adaptive Server to allocate a free pipe buffer.

Action

The average value for WaitTime for event 210 should be very low. High average values for WaitTime may indicate that Adaptive Server has some resource contention. Run sp_monitor or sp_sysmon, or query monEngine to determine if Adaptive Server has sufficient CPU resources.

Event 214: waiting on run queue after yield

Event 214 measures the amount of time a process waits on the run queue after yielding to allow other processes to run. This process is "runnable," not waiting on a lock, physical I/O, or any other wait condition. This event may be caused by insufficient CPU (that is, the server is CPU bound) or table scans in memory.

Event 214 differs from event 215 by indicating a process is performing a CPU-intesive task that exceeded the CPU time allocated by time slice: the process yields the CPU voluntarily and is placed in a runnable state while it waits for the Adaptive Server scheduler to allocate more CPU time. When this occurs, the process continues with the activity it was performing before it yielded the CPU.

Event 215 also indicates that a process is in a runnable state, but for event 214, the process entered this state not because it exceeded the CPU time, but because it encountered a condition that required it to wait for a resource, such as disk or network I/O or a logical lock, before it continues performing its task.

Action

Busy servers typically have high values for Waits. However, high values for WaitTime or the time slice setting may indicate that Adaptive Server has a large number of spids waiting to execute, or that is has spids running which were heavily CPU bound and are not readily yielding their CPU. Query monProcessActivity to identify jobs that have high CPUTime.

Event 215: waiting on run queue after sleep

Event 215 occurs when a process is no longer waiting for another wait event (for example, a logical lock, disk I/O, or another wait event) and is placed on the server's runnable queue. The process must wait until the scheduler allocates CPU time before continuing its task.

See the description for event 214 for differences between event 214 and 215.

Action

Event 215 is a common wait event. The value for Waits for event 215 is typically large. Busy servers have high values for WaitTime because processes are waiting for the Adaptive Server runnable queue for a long time. Reduce the value for time slice to allow more processes to access CPU (this also reduces the average time some processes spend in the CPU) or, if there are sufficient CPUs available on the host machine, increase the number of online engines.

Event 222: replication agent sleeping during flush

If Adaptive Server is a primary server performing replication, the RepAgent process sleeps, waiting for work to do (for example, when rows are added to the log for a database). Event 222 measures the amount of time RepAgent spends asleep.

Action

Depending on the level of activity within a replicated database, event 222 may typically have high values for WaitTime. Typically, you need not perform any actions for this event.

Event 250: waiting for incoming network data

This event measures the time that application processes are active, but waiting for the next request from a client (that is, when jobs are in the `AWAITING COMMAND` state).

Event 250 typically occurs when the application remains connected to the Adaptive Server but is idle.

Action

Because event 250 occurs before Adaptive Server processes each command from a client, the number of Waits and WaitTime may typically be high.

You can use event 250 to estimate how many requests the server has handled from clients.

A high WaitTime value for this event can indicate a large number of idle client connections, or that some client connections remain idle for a long period of time. This wait event can occur between batches or commands sent by the client application, so the Waits value may be high if applications submit a large number of separate commands or batches.

Event 251: waiting for network send to complete

Event 251 measures the amount of time a job waits while sending a reply packet back to a client.

Action

Event 251 may indicate that Adaptive Server is sending large reply sets to clients, or it may indicate a slow or overloaded network. Check the average packet size in the monNetworkIO and monProcessNetIO tables. In each of these tables, the average size is:

(BytesSent) / (PacketsSent)

Increasing the client application's network packet size may improve network performance.

Event 259: waiting until last chance threshold is cleared

When Adaptive Server crosses a last-chance threshold for a database log, every process trying to allocate more log space receives message 7415, and is put to sleep, or suspended, while it waits for available log space. Event 259 measures the amount of time the process waits for this space.

Action

A high value for Waits for this event may indicate that some databases need larger log segments. A high value for the average WaitTime may indicate that you have not defined a threshold procedure, or that a procedure is taking a long time to free log space.

Increasing the frequency of transaction dumps on the database or allocating more space to the log segment may reduce the value for WaitTime.

Event 260: waiting for date or time in waitfor command

Event 260 is normal and expected when processes use the waitfor command.

Action

When a process uses a waitfor command, Adaptive Server puts it to sleep until the requested time expires. Event 260 measures this amount of sleep time.

Event 266: waiting for message in worker thread mailbox

Adaptive Server worker threads communicate with each other and the parent spid through an internal Adaptive Server mechanism called a mailbox. Event 266 measures the amount of time a worker process spends waiting for its mailbox to add a message.

Action

To evaluate event 266, determine the number of parallel queries that were run from monSysWorkerThread.ParallelQueries. If the value for WaitTime is high per query, Adaptive Server may have a resource shortage (generally, CPU time). A high WaitTime value may also indicate unbalanced partitions on objects, causing some worker threads to wait for others to complete.

Event 272: waiting for lock on ULC

Each process allocates a user log cache (ULC) area, which is used to reduce contention on the last log page. Adaptive Server uses a lock to protect the ULC since more than one process can access the records of a ULC and force a flush. Event 272 measures the time the ULC spends waiting for that lock.

Action

Typically, the average value for WaitTime for event 272 is quite low. A high value for average WaitTime may indicate high wait times for other events, forcing the ULC lock holder to wait. You can analyze other wait events to determine what is causing these waits.

Event 334: waiting for Lava pipe buffer for write

Adaptive Server version 15.0 introduced the lava query execution engine. When this engine executes a parallel query, it uses an internal structure called a "pipe buffer" to pass data between the worker processes. Event 334 measures the amount of time Adaptive Server spends waiting for a pipe buffer to be available.

Action

The value for WaitTime should be low when processes execute properly. If this is not the case, contact Sybase Technical Support.

Index

Symbols

Numerics

A

B

C

E

F

G

H

I

M

O

P

Q

R

S

T

V

W

X