



PUBLIC

SAP Adaptive Server Enterprise 16.1

Document Version: 1.0 – 2026-03-30

Performance and Tuning Series: Monitoring Tables

Content

1	Monitoring Tables in SAP ASE.	5
1.1	Where Does the Monitoring Information Come From?.	5
1.2	Using Transact-SQL to Monitor Performance.	6
1.3	Performance Impact of Using the Monitoring Tables.	6
1.4	Monitoring Threshold-Based Events.	6
1.5	Monitoring Multiple Replication Agent Scanner Threads.	7
2	Remotely Accessing Monitoring Tables.	9
3	Configuring the Monitoring Tables to Collect Data.	10
3.1	Configuration Parameters for Monitoring Tables.	11
3.2	Allocating Memory for Pipe Error Messages.	12
3.3	Error 12036.	13
4	Access Control for Monitoring Tables.	14
5	Wrapping Counter Datatypes.	16
6	Stateful Historical Monitoring Tables.	17
6.1	Transient Monitoring Data.	19
7	Using Monitoring Tables in a Clustered Environment.	21
7.1	Configuring the System View.	21
8	Monitoring Tables for the Statement Cache.	22
8.1	Configuring SAP ASE to Monitor the Statement Cache.	22
8.2	Deleting Statements From the Statement Cache.	22
8.3	Obtaining the Hash Key from the SQL Text.	22
8.4	Displaying Text and Parameter Information for Cached Statements.	23
9	Examples of Querying the Monitoring Tables.	24
10	Wait Events.	26
10.1	Event 19: xact coord: pause during idle loop.	26
10.2	Event 29: waiting for regular buffer read to complete.	27
10.3	Event 30: wait to write MASS while MASS is changing.	27
10.4	Event 31: waiting for buf write to complete before writing.	28
10.5	Event 32: waiting for an APF buffer read to complete.	28
10.6	Event 35: waiting for buffer validation to complete.	29
10.7	Event 36: waiting for MASS to finish writing before changing.	29

10.8	Event 37: wait for MASS to finish changing before changing.	30
10.9	Event 41: wait to acquire latch.	30
10.10	Event 46: wait for buf write to finish getting buf from LRU.	31
10.11	Event 51: waiting for last i/o on MASS to complete.	31
10.12	Event 52: waiting for i/o on MASS initiated by another task.	32
10.13	Event 53: waiting for MASS to finish changing to start i/o.	32
10.14	Event 54: waiting for write of the last log page to complete.	33
10.15	Event 55: wait for i/o to finish after writing last log page.	33
10.16	Event 57: checkpoint process idle loop.	34
10.17	Event 61: hk: pause for some time.	34
10.18	Event 70: waiting for device semaphore.	34
10.19	Event 83: wait for DES state is changing.	35
10.20	Event 84: wait for checkpoint to complete.	35
10.21	Event 85: wait for flusher to queue full DFLPIECE.	35
10.22	Event 91: waiting for disk buffer manager i/o to complete.	36
10.23	Event 99: wait for data from client.	36
10.24	Event 104: wait until an engine has been offlined.	36
10.25	Event 124: wait for mass read to finish when getting page.	37
10.26	Event 142: wait for logical connection to free up.	37
10.27	Event 143: pause to synchronise with site manager.	38
10.28	Event 150: waiting for a lock.	38
10.29	Event 157: wait for object to be returned to pool.	39
10.30	Event 169: wait for message.	39
10.31	Event 171: wait for CTLIB event to complete.	40
10.32	Event 178: waiting while allocating new client socket.	40
10.33	Event 179: waiting while no network read or write is required.	40
10.34	Event 197: waiting for read to complete in parallel dbcc.	41
10.35	Event 200: waiting for page reads in parallel dbcc.	41
10.36	Event 201: waiting for disk read in parallel dbcc.	42
10.37	Event 202: waiting to re-read page in parallel.	42
10.38	Event 203: waiting on MASS_READING bit in parallel dbcc.	42
10.39	Event 205: waiting on TPT lock in parallel dbcc.	43
10.40	Event 207: waiting sending fault msg to parent in PLL dbcc.	43
10.41	Event 209: waiting for a pipe buffer to read.	44
10.42	Event 210: waiting for free buffer in pipe manager.	44
10.43	Event 214: waiting on run queue after yield.	44
10.44	Event 215: waiting on run queue after sleep.	45
10.45	Event 222: replication agent sleeping during flush.	46
10.46	Event 250: waiting for incoming network data.	46
10.47	Event 251: waiting for network send to complete.	47
10.48	Event 259: waiting until last chance threshold is cleared.	47

10.49	Event 260: waiting for date or time in waitfor command.	48
10.50	Event 266: waiting for message in worker thread mailbox.	48
10.51	Event 272: waiting for lock on ULC.	48
10.52	Event 334: waiting for Lava pipe buffer for write.	49
10.53	Event 374: wait for lock pending/data pending to be cleared.	49
10.54	Event 375: OCM wait for finishing BAST handling.	50
10.55	Event 380: lock/data pending to reset when OCM_ERR_DIDNTWAIT.	50
10.56	Event 389: OCM wait for pushing data flag to be cleared.	50
10.57	Event 442: waiting for cluster physical lock acquisition.	51
10.58	Event 444: waiting for cluster physical lock lookup.	51
10.59	Event 449: waiting for cluster object lock lookup.	52
10.60	Event 454: waiting for cluster logical lock lookup.	52
10.61	Event 483: Waiting for ack of a multicast synchronous message.	52
10.62	Event 780: Bucket pool manager consolidator sleep.	53
10.63	Event 801: KPP handler pause event.	53

1 Monitoring Tables in SAP ASE

SAP ASE 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 SAP ASE, 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 SAP ASE 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, which can be run manually automatically during server installation. These proxy tables use an interface to the SAP ASE to collect monitoring data when you perform a query.

Because SAP ASE versions may change the definitions of the monitoring tables, SAP recommends that when you upgrade, you run the `installmaster` script before using the monitoring tables.

Note

You must have the `mon_role` to query these tables.

Related Information

[Access Control for Monitoring Tables \[page 14\]](#)

1.1 Where Does the Monitoring Information Come From?

SAP ASE gathers the information for the monitoring tables from several sources.

Information is gathered from:

- Global monitor counters (for example, `monSysWaits`, which has a limited number of rows).
- Resource-specific monitor counters associated with a single server resource, such as engines or the procedure or data cache (for example, `monCachedProcedures`, for which the number of result rows depends on a snapshot of cached, compiled objects in procedure 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.
- Historical data stored in in-memory 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.

1.2 Using Transact-SQL to Monitor Performance

Providing monitoring information as tables enables you to use Transact-SQL to monitor SAP ASE.

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 SAP ASE 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 SAP ASE to define remote procedures as local tables.

1.3 Performance Impact of Using the Monitoring Tables

Using the monitoring tables typically results in a performance impact on SAP ASE of 5 percent or less.

However, the performance impact of the monitoring tables depends on a number of factors:

- The frequency and other characteristics of the activity on the server
- The complexity, size, and performance characteristics of the queries you are executing on the server
- The configuration of the server
- The monitoring tables you are querying
- The frequency with which you execute the monitoring table queries

1.4 Monitoring Threshold-Based Events

You can configure, record, and list threshold events.

To configure the amount of time a query spends using CPU, use the `sp_add_resource_limit cpu_time` limit type.

To configure the number of locks used, use the `sp_add_resource_limit lock_count` limit type.

The syntax is:

```
sp_add_resource_limit name, appname, rangename, limittype, limitvalue [,
enforced [, action [, scope ]]]
```

In this example, when the CPU time of the query batch takes longer than 120 seconds to execute, SAP ASE records the events to `monThresholdEvent`:

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,cpu_time, 120, 2, 5, 2
```

`monThresholdEvent` lets you record all violations for configured resource limitations, and allows you to determine the thresholds for:

- `tempdb` usage per transaction
- Total runtime of a query or transaction
- Number of rows fetched or written by a query
- Number of rows read by a query
- Estimated plan cost of a query
- Total logical/physical I/Os of a query
- Total CPU time of a query
- Number of locks used

To record events to the `monThresholdEvent` table, enable the `threshold event monitoring` and `threshold event max messages` options. (see the *Reference Manual: Configuration Parameters*).

Note

There is a separate instance of the `monThresholdEvent` table on each instance of an SAP ASE cluster. In order to retrieve the events from all instances in the cluster in a query on the `monThresholdEvent` table, ensure that the `system_view` property is set to "cluster". In order to see only the events on the local instance, set `system_view` to "instance", or use a predicate on the `InstanceID` column in the where clause in your query to specify the ID of a chosen instance.

1.5 Monitoring Multiple Replication Agent Scanner Threads

SAP ASE monitoring tables support multiple scanner threads for Replication Agent. Enabling multiple scanner threads for Replication Agent provides access to available replication paths and improved replication performance.

When you enable multiple scanners, monitoring tables provide information about Replication Agent scanner task performance. The monitoring tables include:

- `monRepSenders` – includes these columns to provide compatibility with the single-task model.
 - `NumberOfTruncPointRequested`
 - `NumberOfTruncPointMoved`
 - `AvgTruncPointInterval`

- **monRepScanners** – the **NumberOfTruncPointRequested** and **NumberOfTruncPointMoved** columns are populated when the single-task scanning of a log model is used. When the Multi-Path Replication (MPR) model is used, these columns values are 0. The related information for the MPR model can be found on the **monRepSenders** table.
- **monRepCoordinator**

Any MPR binding model can be used with multiple scanners. For more information about multiple scanner threads, see:

Multiple Scanners in the Replication Server New Features Guide.

2 Remotely Accessing Monitoring Tables

installmontables is provided 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 to create the monitoring tables).

Run installmontables to view the instructions for editing. For example:

```
isql -Usa -P<sa_password> -S<server 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
```

3 Configuring the Monitoring Tables to Collect Data

Use `sp_configure enable monitoring` to configure SAP ASE to start collecting monitoring information.

Many of the monitoring tables require that you enable one or more of the configuration options before SAP ASE collects their data. Different tables require different options. There are a number of configuration options, listed below, that control the collection of monitoring data in various areas.

By default, SAP ASE does not collect the monitoring information required by the monitoring tables. To start collecting monitoring information, set:

- `enable cistol`
- `enable monitoring to 1`

SAP ASE 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`
- `enable spinlock monitoring`
- `enable stmt cache monitoring`
- `execution time monitoring`
- `errorlog pipe active`
- `errorlog pipe max messages`
- `lock timeout pipe active`
- `lock timeout 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`
- `threshold event max messages`
- `threshold event monitoring`

- wait event timing

Note

See *Reference Manual: Configuration Parameters* for descriptions of the configuration parameters.

3.1 Configuration Parameters for Monitoring Tables

SAP ASE 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 SAP ASE collects data. You must grant the mon_role to users before they can query some monitoring tables.

If you are not using a monitoring table, disable the associated configuration parameters, reducing the load on the SAP ASE caused by collecting monitoring data.

These monitoring tables require enabled configuration parameters or special permissions:

- monCachePool – enable enable monitoring
- monCachedProcedures – enable enable monitoring and statement statistics active
- monCachedStatement – enable enable monitoring, statement cache size, and enable stmt cache monitoring
- mon CLMObjectActivity – enable enable monitoring
- monDataCache – enable enable monitoring
- monDeadLock – enable enable monitoring, deadlock pipe active, deadlock pipe max messages
- monDeviceIO – enable enable monitoring
- monEngine – enable enable monitoring
- monErrorLog – enable enable monitoring, errorlog pipe active, errorlog pipe max messages
- monHADRMembers – Requires the mon_role role.
- monIOQueue – enable enable monitoring
- monLocks – enable enable monitoring
- monLockTimeout – enable enable monitoring, lock timeout pipe active, lock timeout pipe max messages
- monMemoryUsage – enable enable monitoring. Requires the mon_role role.
- monNetworkIO – enable enable monitoring
- monOpenDatabases – enable enable monitoring
- monOpenObjectActivity – enable enable monitoring, object lockwait timing, per object statistics active
- monOpenPartitionActivity – enable enable monitoring, per object statistics active
- monProcedureCache – enable enable monitoring
- monProcess – enable enable monitoring and wait event timing
- monProcessActivity – enable enable monitoring and wait event timing
- monProcessNetIO – enable enable monitoring

- `monProcessObject` – enable enable monitoring and per object statistics active
- `monProcessProcedures` – enable enable monitoring and statement statistics active
- `monProcessSQLText` – enable enable monitoring, SQL batch capture, and max SQL text monitored
- `monProcessStatement` – enable enable monitoring
- `monProcessWaits` – enable enable monitoring, process wait events, and wait event timing
- `monProcessWorkerThread` – enable enable monitoring
- `monServiceTask` – enable enable monitoring
- `monSpinlockActivity` – enable enable monitoring and enable spinlock monitoring
- `monSQLRepActivity` – enable enable monitoring and per object statistics active
- `monSQLRepMisses` – enable enable monitoring and per object statistics active
- `monStatementCache` – enable enable monitoring, statement cache size, and enable stmt cache monitoring
- `monSysPlanText` – enable enable monitoring and execution time monitoring. Requires the `mon_role` role
- `monSysPlanText` – enable enable monitoring, plan text pipe active, and plan text pipe max messages
- `monSysSQLText` – enable enable monitoring, SQL batch capture, SQL text pipe active, SQL text pipe max messages, and max SQL text monitored
- `monSysStatement` – enable enable monitoring, per object statistics active, statement pipe active, statement pipe max messages, and statement statistics active
- `monSysWaits` – enable enable monitoring and wait event timing
- `monSysWorkerThread` – enable enable monitoring
- `monTableCompression` – enable enable monitoring, per object statistics active, and capture compression statistics
- `monTempdbActivity` – enable enable monitoring, object lockwait timing, and per object statistics active
- `monThresholdEvent` – enable enable monitoring, threshold event monitoring, and 'threshold event max messages. Requires the `mon_role` role.

3.2 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`

SAP ASE 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 SAP ASE for the new pipe size to take effect.

These are algorithms for determining the size for the parameters:

- For an individual SAP ASE, 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.

Related Information

[Using Monitoring Tables in a Clustered Environment \[page 21\]](#)

3.3 Error 12036

If you query the monitoring tables, but have not enabled all the configuration parameters the tables require, SAP ASE 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 SAP ASE is not collecting the data required to populate one or more columns in the table.

Consider enabling the required configuration parameters.

4 Access Control for Monitoring Tables

Access to many of the monitoring tables is restricted to users with the `mon_role`. Only users who are granted the role can run queries on the monitoring tables.

However, you need the `sapmon_role` role to access the sensitive monitoring tables `monLicense` and `monSSLCertInfo`.

You can access the following monitoring tables without the `mon_role` or `sapmon_role` roles:

- `monControllerLoad`
- `monEngine`
- `monInmemoryStorage`
- `monIOController`
- `monLogicalCluster`
- `monLogicalClusterAction`
- `monLogicalClusterInstance`
- `monLogicalClusterRoute`
- `monProcessMigration`
- `monRepCoordinator`
- `monRepLogActivity`
- `monRepScanners`
- `monRepScannersTotalTime`
- `monRepSchemaCache`
- `monRepSenders`
- `monServiceTask`
- `monSysLoad`
- `monTableTransfer`
- `monTask`
- `monThread`
- `monThreadPool`
- `monWorkload`
- `monWorkloadPreview`
- `monWorkloadProfile`
- `monWorkQueue`
- `monWorkloadRaw`

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 *System Administration Guide, Volume 1 > Managing User Permissions*.

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 a 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.

5 Wrapping Counter Datatypes

Some columns in the monitoring tables contain integer counter values that are incremented throughout the life of SAP ASE. 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 or Indicators = 3
```

6 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.

SAP ASE 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`

Note

In `monSysPlanText` and `monSysSQLText`, the values of the columns `BatchID`, `ContextID`, `ProcedureID`, and `PlanID` are modified effective SAP ASE versions 15.0.3 and later. For the changes in these columns, see the *Reference Manual: Tables*.

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 SAP ASE writes information to the buffer.

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

Monitoring table	Configuration parameters
<code>monErrorLog</code>	<code>errorlog pipe active</code> <code>errorlog pipe active messages</code>

Monitoring table	Configuration parameters
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.

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 *Configuring the Monitoring Tables to Collect Data* and *Reference Manual: Configuration Parameters* for more information about `sp_configure`.

SAP ASE 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.
- SAP ASE 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. SAP ASE 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.

Related Information

[Configuring the Monitoring Tables to Collect Data \[page 10\]](#)

6.1 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 SAP ASE 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.

SAP 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.

7 Using Monitoring Tables in a Clustered Environment

Monitoring tables in a clustered environment use the value of the `system_view` setting to determine whether to gather data from all instances in the cluster or only from the instance to which the connection that issues the query is connected.

Use `set` command to configure the `system_view` parameter. Query the `<@@system_view>` global variable to determine the current setting for `system_view`.

`system_view` 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.

7.1 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.

8 Monitoring Tables for the Statement 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..

For more information about the statement cache, see *System Administration Guide, Volume 2 > Configuring Memory*.

8.1 Configuring SAP ASE to Monitor the Statement Cache

Use `enable stmt cache monitoring` to configure SAP ASE to collect the monitoring information on the statement cache.

8.2 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 (<SSQLID>)
```

8.3 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*.

SAP ASE 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')
```

Related Information

[Displaying Text and Parameter Information for Cached Statements \[page 23\]](#)

8.4 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
```

9 Examples of Querying the Monitoring Tables

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 SAP ASE, 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:

1. Run all queries in your applications that access the table in question. Ensure that SAP ASE runs long enough so all applications have performed their selects.
2. 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
```

10 Wait Events

SAP ASE task management includes these states for a process: running, runnable, sleeping, and blocked.

ⓘ Note

This section contains a list of the most commonly seen wait events. See the `WaitEventInfo` table for a complete list of wait events.

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. SAP ASE 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.

10.1 Event 19: xact coord: pause during idle loop

The SAP ASE 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.

10.2 Event 29: waiting for regular buffer read to complete

A wait caused by a physical read (most likely a cache miss) which occurs when SAP ASE 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.

10.3 Event 30: wait to write MASS while MASS is changing

SAP ASE is attempting to write to a memory address space segment (MASS). A MASS is one or more contiguous pages SAP ASE 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)

10.4 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.

10.5 Event 32: waiting for an APF buffer read to complete

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

Action

A high value for `waits` may indicate that SAP ASE is using asynchronous prefetch too often. Tuning the local APF limit for cache pools may reduce contention for APF pages.

Since SAP ASE 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.

10.6 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, SAP ASE validates the success of the read operation. Because SAP ASE 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.

10.7 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.

If event 36 occurs because of an update to a page, partitioning the cache has no effect. However, if event 36 occurs when page updates are not taking place, partitioning the cache may expedite the writes.

10.8 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.

10.9 Event 41: wait to acquire latch

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

SAP ASE uses a latch as a transient lock to guarantee a page's contents are not changed while another process reads or writes data. SAP ASE 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 a server resource shortage, resulting from:

- A hash table that is too small for a lock, resulting in very long hash chains that SAP ASE 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 SAP ASE with sp_sysmon*.

10.10 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 SAP ASE 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.

10.11 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.

10.12 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.

10.13 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.

SAP ASE 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.

10.14 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 SAP ASE frequently updates the last page of the transaction log, SAP ASE 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 improving 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.

10.15 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 SAP ASE is making a large number of updates to the transaction log because of committed transactions or other operations that requiring writing the transaction log to disk.

A high value for `WaitTime` for event 55 indicates that writes may be taking a long time to complete.

Action

Typically, the value for `Waits` should be much higher than the value for `WaitTime`.

10.16 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.

10.17 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.

10.18 Event 70: waiting for device semaphore

If you are using SAP ASE 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 SAP ASE 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.

10.19 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 SAP ASE is releasing an allocated descriptor, which typically happens when SAP ASE 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.

10.20 Event 84: wait for checkpoint to complete

SAP ASE is dropping a DES, which typically occurs when SAP ASE 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).

10.21 Event 85: wait for flusher to queue full DFLPIECE

When SAP ASE 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. SAP ASE 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.

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

When SAP ASE 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 SAP ASE 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.

10.23 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.

10.24 Event 104: wait until an engine has been offlined

SAP ASE includes an engine cleanup background process that runs continuously. This service performs clean up tasks after an engine goes offline. This process typically remains in a sleep state, waking up every 30 seconds to check for work to do. Event 104 measures the cumulative amount of time this process slept between tasks.

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.

10.25 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.

10.26 Event 142: wait for logical connection to free up

When SAP ASE executes an RPC on a remote server using the site handler mechanism, it creates logical connections. Event 142 occurs when server must close a logical connection but finds another process using it. 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.

10.27 Event 143: pause to synchronise with site manager

SAP ASE communicates with a remote server using a site manager, but if another process is attempting to connect to that remote server. Event 143 measures the amount of time SAP ASE 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.

10.28 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 SAP ASE 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, SAP ASE 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.

10.29 Event 157: wait for object to be returned to pool

The SAP ASE 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 must 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 SAP ASE is waiting. Use `sp_countmetadata` and `sp_monitorconfig` to identify which structures are using the maximum configuration to determine which resources you should increase.

10.30 Event 169: wait for message

Some 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 SAP ASE 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.

10.31 Event 171: wait for CTLIB event to complete

Indicates that SAP ASE 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.

10.32 Event 178: waiting while allocating new client socket

A network listener is a server process that handles a client's incoming connection requests. Event 178 measures the time SAP ASE 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.

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

The SAP ASE 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.

10.34 Event 197: waiting for read to complete in parallel dbcc

When you run `dbcc checkstorage`, SAP ASE must occasionally perform asynchronous I/O on the workspace to read or write a single reserved buffer. Event 197 measures the time SAP ASE 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.

10.35 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.

10.36 Event 201: waiting for disk read in parallel dbcc

When you run `dbcc checkverify`, SAP ASE 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.

10.37 Event 202: waiting to re-read page in parallel

When you run `dbcc checkstorage`, SAP ASE 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.

10.38 Event 203: waiting on MASS_READING bit in parallel dbcc

When you run `dbcc checkstorage`, SAP ASE 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.

10.39 Event 205: waiting on TPT lock in parallel dbcc

When you run `dbcc checkstorage` to check text and image pages, SAP ASE 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.

10.40 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 SAP ASE 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.

10.41 Event 209: waiting for a pipe buffer to read

When SAP ASE 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 SAP ASE has sufficient CPU and I/O bandwidth.

10.42 Event 210: waiting for free buffer in pipe manager

Event 210 measures the amount of time a process waits for SAP ASE to allocate a free pipe buffer.

When SAP ASE 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.

Action

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

10.43 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-intensive 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 SAP ASE 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 SAP ASE has a large number of spids waiting to execute, or that it has spids running which were heavily CPU bound and are not readily yielding their CPU. Query `monProcessActivity` to identify jobs that have high `CPUTime`.

10.44 Event 215: waiting on run queue after sleep

Event 215 measures the amount of time between when an event wakes a sleeping process and when the process is able to run on an engine.

For example, if a process sleeps on send event 251, event 215 measures the amount of time that passes after the send completes and the process is running again on an engine.

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 SAP ASE 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.

10.45 Event 222: replication agent sleeping during flush

If SAP ASE 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.

10.46 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 SAP ASE but is idle.

Action

Because event 250 occurs before SAP ASE 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.

10.47 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 SAP ASE 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:

$(\text{BytesSent}) / (\text{PacketsSent})$

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

10.48 Event 259: waiting until last chance threshold is cleared

When SAP ASE 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`.

10.49 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, SAP ASE puts it to sleep until the requested time expires. Event 260 measures this amount of sleep time.

10.50 Event 266: waiting for message in worker thread mailbox

SAP ASE worker threads communicate with each other and the parent spid through an internal SAP ASE 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, SAP ASE 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.

10.51 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. SAP ASE 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.

10.52 Event 334: waiting for Lava pipe buffer for write

SAP ASE 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 SAP ASE 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 Technical Support.

10.53 Event 374: wait for lock pending/data pending to be cleared

A lock request is blocked because a required lock is unavailable at the local node and a currently pending request.

Action

A high number of conflicting requests for the same lock across different nodes indicates that the application is issuing conflicting lock requests on the same object from multiple nodes. The lock emanates from the object coherency manager, which manages the metadata describing database objects across all of the instances in the cluster.

You may improve performance by directing most requests for the object in question to a single node.

10.54 Event 375: OCM wait for finishing BAST handling

One or more ocm_lock requests are blocked by a conflicting ocm_lock request from a different node.

Action

A high value for this wait event means that different nodes are sending a high number of conflicting requests for the same ocm_lock. The application is probably issuing these conflicting lock requests from multiple nodes. You may improve performance by directing most requests for the ocm_lock to a single node.

10.55 Event 380: lock/data pending to reset when OCM_ERR_DIDNTWAIT

If the cluster lock manager cannot immediately grant a lock request, it returns a LOCK_DIDNTWAIT message, which SAP ASE translates as OCM_ERR_DIDNT_WAIT. The ocm_lock request goes to sleep until the AST returns from the master database with a response for the lock-request.

Action

A high value for this wait event indicates there are numerous conflicting requests for the same ocm_lock across different nodes. This may be caused by the application issuing conflicting lock requests on the same object from multiple nodes. You may improve performance by directing requests for the object in question to a single node.

10.56 Event 389: OCM wait for pushing data flag to be cleared

One or more ocm_lock requests cannot be processed.

SAP ASE cannot process one or more ocm_lock requests because:

- Another lock request is holding the lock
In this situation, a high value indicates there are numerous exclusive lock requests that need to perform heavy operations without losing the ocm_lock. This is an internal issue.
To improve performance, determine why SAP ASE is holding the lock for so long.
- Another request is pushing the data.

In this situation, a high value may indicate that data resides only in memory and SAP ASE must regularly push this data to other nodes to avoid losing data in the case of node failure. This adversely affects performance because SAP ASE performs no useful work while it is pushing the data to other nodes. This is an internal issue.

To improve performance, determine why SAP ASE is holding the lock or pushing the data for so long.

- A canceled transaction is pending (the requested data is probably not being delivered, and the OCM on this node is taking corrective action).

This wait event is rare (usually when an instance is expecting data from a node that just crashed). A high value for this wait event may result from a high rate of node or hardware failure.

10.57 Event 442: waiting for cluster physical lock acquisition

Action

To fix this situation:

- Check the CIPC send and receive queues – the wait time required for look-ups might increase if messages are stuck in the queues and not immediately processed.
- Check the network – congested networks can slow message transmission.
- Avoid the badly partitioned setting – enable trace flag 15815 for clusters that have more than two instances if the setting is well partitioned. This trace flag forces the affinity mode for master assignment, so the lock requester and the lock master are on the same instance and the lock request is managed by the current instance.

10.58 Event 444: waiting for cluster physical lock lookup

Action

To fix the situation, check the:

- Physical lock hash bucket size – large hash buckets may increase the amount of time required to look up locks.
- CIPC send and receive queues – the wait time required for look-ups might increase if messages are stuck in the queues and not immediately processed.
- Network – congested networks can slow message transmission.

10.59 Event 449: waiting for cluster object lock lookup

Action

To solve the situation, check the:

- Object lock hash bucket size – large hash buckets may increase the amount of time required to look up locks.
- CIPC send and receive queues – the wait time required for look-ups might increase if messages are stuck in the queues and not immediately processed.
- Network – congested networks can slow message transmission.

10.60 Event 454: waiting for cluster logical lock lookup

Action

To solve this situation, check the:

- Logical lock (row, page, and table) hash bucket size – large hash buckets may increase the amount of time required to look up locks.
- CIPC send and receive queues – the wait time required for look-ups might increase if messages are stuck in the queues and not immediately processed.
- Network – congested networks can slow message transmission.

10.61 Event 483: Waiting for ack of a multicast synchronous message

Although this wait event occurs with some frequency, it is a result of normal activity.

Action

No action required.

10.62 Event 780: Bucket pool manager consolidator sleep

The amount of time, in microseconds, the `MEMORY_TUNE` daemon process or `spid` pauses after consolidating, shrinking, deleting, and auto tuning the configured bucket fragment pools.

10.63 Event 801: KPP handler pause event

The amount of time, in microseconds, the `KPP_HANDLER` daemon process or `spid` pauses after scanning the RSA key pair pool entries, but did not find any unrefreshed or used entries.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
 - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
 - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.

