

What Larry should have told Bill many years ago

Erling Skaale

esk@miracleas.dk

MIRACLE a/s

www.miracleas.dk

Who am I?

- Consultant SQL Server
 - Miracle A/S Denmark...future – who knows 😊
- Microsoft certified...bla, bla bla – who cares
- 15 years DB2
- 7 years SQL Server
- Able to alter size of Oracle table space
- Oaktable prospect
- Oracle wannabe

Know your enemy

- Why?
 - Keep your friends close and your enemies closer.
 - Don't be humiliated.
 - Complex world
 - Fever and fever 'only Oracle' installations

Rumours about SQL Server

- Toy database
- Dangerous to use – no consistency
- No way to look at performance
- No x\$ or v\$ views
- No way to look at events – 10046
- Locking and blocking

Scratch the surface

Be prepared!



Prior and today

- DBCC SQLPerf
 - Waitstats
 - Umsstats
 - Etc.
- Sysprocesses
- Syscacheobjects
- No support in the future...as if there have been any 😊

SQL Server 2005 – Dynamic Management Views (DMV)

- Expose server state in query-able format
 - State is generally in memory (not persisted)
 - Internal structures
 - Statistics
 - Materialized in to a rowset
 - Can be filtered, ordered, joined like a regular table
- Low overhead
 - For many tables, there is no performance penalty
 - Some tables expose statistics that need to be maintained
- Why DMV's
 - Understandable even without intimate knowledge of the source code
 - More efficient and less intrusive than using a debugger

General Server DMV's and DMF's

- dm_exec_*
 - Execution of user code and associated connections
- dm_os_*
 - Low level system (server-wide) info such as memory, locking & scheduling
- dm_tran_*
 - Transactions & isolation
- dm_io_*
 - I/O on network and disks
- dm_db_*
 - Databases and database objects
(sys.dm_db_index_usage_stats)

Component level DMV's and DMF's

- dm_repl_*
 - Replication
- dm_broker_*
 - SQL Service Broker
- dm_fts_*
 - Full Text Search
- dm_qn_*
 - Query Notifications

SQL Server 2005 – SQLOS DMV's

- Scheduler
 - sys.dm_os_schedulers
 - sys.dm_os_tasks
 - sys.dm_os_workers
 - sys.dm_os_worker_local_storage
 - sys.dm_os_threads
 - sys.dm_io_pending_io_requests
- Synchronization
 - sys.dm_tran_locks
 - sys.dm_os_waiting_tasks
 - sys.dm_os_latch_stats
 - sys.dm_os_wait_stats
- General Support
 - sys.dm_os_loaded_modules
- Memory Management
 - sys.dm_os_memory_clerks
 - sys.dm_os_memory_objects
 - sys.dm_os_memory_allocations
 - sys.dm_os_memory_caches
 - sys.dm_os_memory_pools
 - sys.dm_os_memory_heaps
 - sys.dm_os_virtual_addresses
 - sys.dm_os_virtual_address_dump
 - sys.dm_os_stacks
 - sys.dm_os_ring_buffers
 - sys.dm_os_buffer_descriptors
 - sys.dm_os_memory_cache_entries
- Hosting
 - sys.dm_os_hosts

Query DMV's

- `sys.dm_exec_query_stats`
- `sys.dm_exec_cached_plans`
- `sys.dm_exec_connections`
- `sys.dm_exec_query_optimizer_info`
- `sys.dm_exec_query_transformation_stats`
- `sys.dm_exec_requests`
- `sys.dm_exec_sessions`

Query DMV's



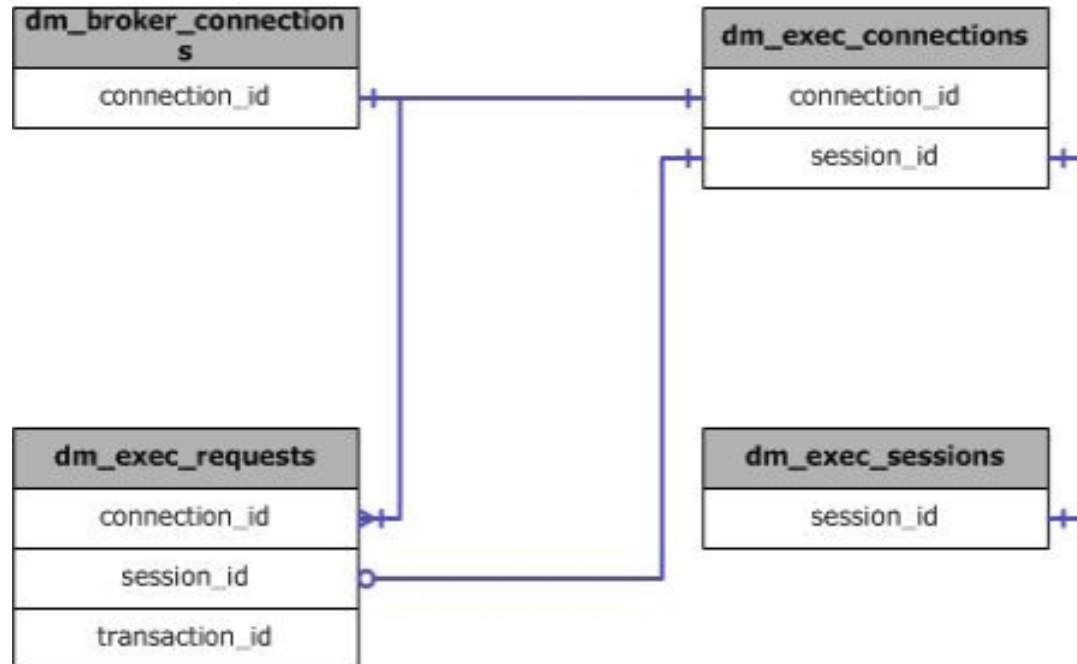
Query DMV's

- `dm_exec_query_stats`
 - Provides aggregate performance statistics for cached query plans. The view contains one row per query plan and the lifetime of the row is tied to the plan itself. When the plan is removed from the cache for whatever reason, the row is eliminated from `dm_exec_query_stats`

Query DMV's

- `dm_exec_cached_plans`
 - Provides information about the query execution plans that are cached by SQL Server for faster query execution

Query DMV's



Query DMV's

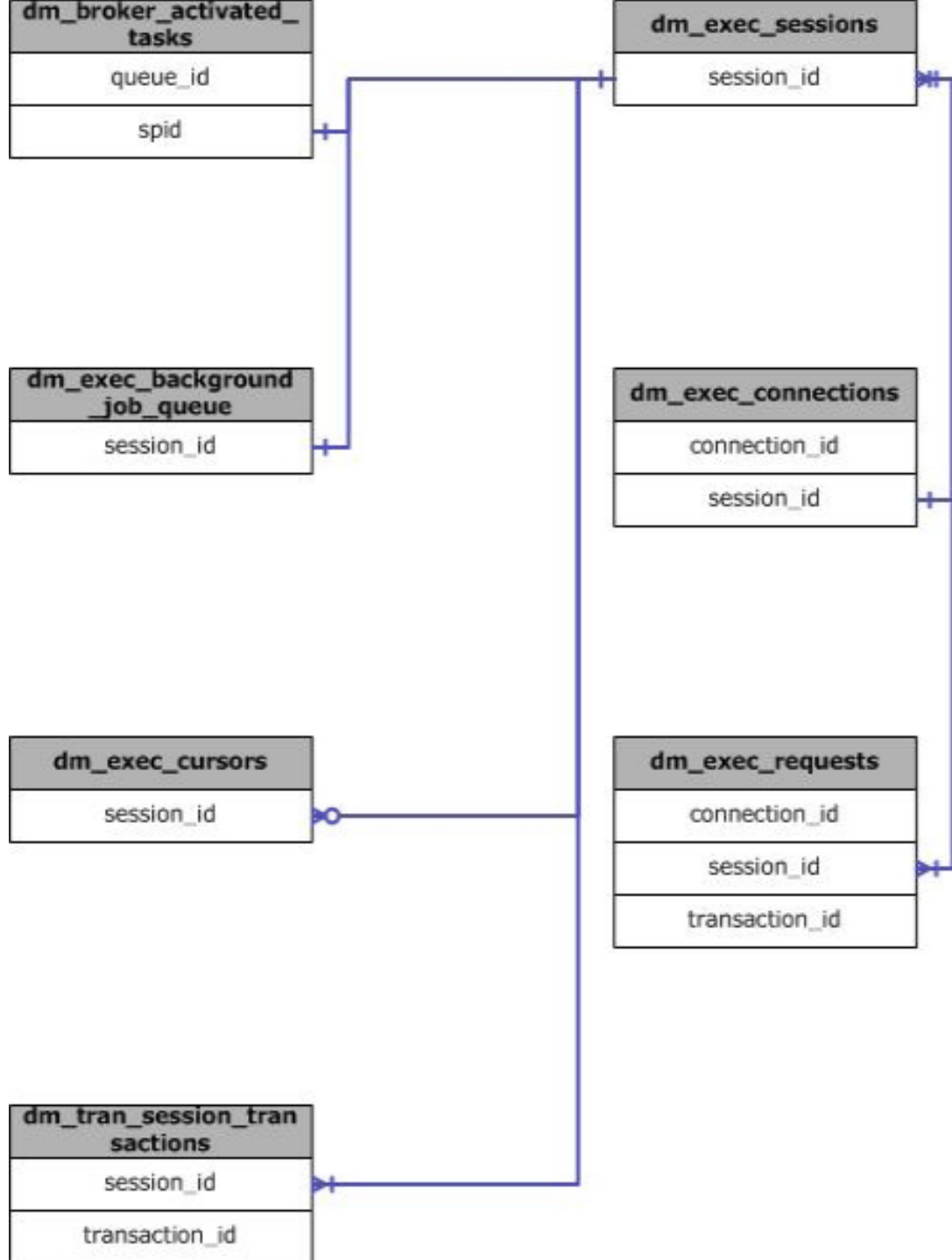
- `dm_exec_connections`
 - Provides information about the connections established to this sql server on various databases by different users local/remote and the details of each connection
 - A physical connection can have up to 10 sessions
 - `ado.net 2.0 SqlConnection Session Pooling`

Query DMV's

- `dm_exec_requests`
 - Provides information about each request executing within SQL Server.
 - Waittime, waittype, read, writes, `cpu_time...`
 - Much like `sysprocesses`

Query DMV's

- Dm_exec_sessions
 - Contains one row per authenticated session on the SQL Server



Query DMV's

- `dm_exec_query_optimizer_info`
 - Provides detailed statistics about the operation of the SQL Server query optimizer

Synchronization DMV's

- `sys.dm_tran_locks`
- `sys.dm_os_waiting_tasks`
- `sys.dm_os_latch_stats`
- `sys.dm_os_wait_stats`

Synchronization DMV's

- `dm_os_wait_stats`
 - Contains a row for each wait type(accumulated)
 - Replaces DBCC SQLPerf(waitstats)
 - DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR)

Synchronization DMV's

- `dm_os_latch_stats`
 - Contains information on latch waits by class
 - `DBCC SQLPERF ('sys.dm_os_latch_stats', CLEAR)`

Synchronization DMV's

- `dm_os_waiting_tasks`
 - Describes the wait queue of tasks, which are waiting on some resource. It is a simultaneous representation of all wait-queues in the system.

Synchronization DMV's



Synchronization DMV's

- `dm_tran_locks`
 - Transaction Locking
 - Contains information about currently active lock manager resources
 - `dbcc traceon (3605, 3604, -1)`
 - `dbcc lock(StallReportThreshold, 200)`

Appendix SQL Waits

• ASYNC_DISKPOOL_LOCK

- RARE During Backup and Restore (e.g. including zeroing out pages) threads written in parallel.
- Possible disk bottleneck. See disk perf counters for confirmation.

ASYNC_IO_COMPLETION

- Waiting for asynchronous IO requests to complete. Identify disk bottlenecks, using PERF Counters, Profiler,::fn_virtualfilestats and SHOWPLAN

- Any of the following will reduce these waits:

- 1. Adding additional IO bandwidth,
- 2. Balancing IO across other drives
- 3. Reducing IO with proper indexing
- 4. Check for bad query plans
- 5. Check for memory pressure

SQL Waits

- CMEMTHREAD - Waiting for thread safe memory objects
- CURSOR - Asynch Cursor thread
- CXPACKET
 - Parallel process waits. One or more parallel processes are complete, waiting for others to complete. Possible skew of data possible lock of a range for this cpu meaning one parallel process is behind, etc.
 - In an OLTP environment, excessive CXPACKET waits can impact the throughput of other OLTP traffic.
 - In a DW environment, CXPACKET waits are expected for multiple proc environments.
 - Check for parallelism – sp_Configure “max degree of parallelism”.
 - If max degree of parallelism = 0, you may want to do one of the following:
 - 1. turn off parallelism entirely: set max degree of parallelism to 1
 - 2. limit parallelism by setting max degree of parallelism to some number less than the total number of CPUs. For example if you have 8 procs, set max degree of parallelism to ≤ 4 .

SQL Waits

- DBTABLE
 - New Checkpoint request that is waiting for outstanding checkpoint request to complete
 - See SQL Buffer Cache perf counters:
 - 1. Page Life Expectancy
 - 2. Checkpoint pages/sec
 - 3. Lazywrites/sec
- DTC
 - Waiting for Distributed Transaction Coordinator
 - Check transaction isolation level

SQL Waits

- EC

- Non-parallel synchronization between parent and child thread

- EXCHANGE

- Waiting on a parallel process to complete, shutdown or startup.
- Check for parallelism – sp_Configure “max degree of parallelism”. If max degree of parallelism = 0, you may want to do one of the following:
 - 1. turn off parallelism entirely: set max degree of parallelism to 1
 - 2. limit parallelism by setting max degree of parallelism to some number less than the total number of CPUs. For example if you have 8 procs, set max degree of parallelism to ≤ 4 .

SQL Waits

- EXECSYNC
 - Query memory and spooling to disk
- IO_COMPLETION
 - Waiting for IO requests to complete.
 - Identify disk bottlenecks, using PERF Counters, Profiler, ::fn_virtualfilestats and SHOWPLAN
 - Any of the following will reduce these waits:
 - 1. Adding additional IO bandwidth,
 - 2. Balancing IO across other drives
 - 3. Reducing IO with proper indexing
 - 4. Check for bad query plans

SQL Waits

- LATCH_x

- Latches are short term light weight synchronization objects. Latches are not held for the duration of a transaction. “Plain” latches are generally not related to IO. These latches can be used for a variety of things, but they are not used to synchronize access to buffer pages (PAGELATCH_x is used for that). Possibly the most common case is contention on internal caches (not the buffer pool pages), especially when using heaps and/or text.
- If high, check PERFMON for
 - 1. memory pressure
 - 2. SQL Latch waits (ms)
- Look for LOG and Pagelatch_UP wait types.
- Latch_x waits can often be alleviated by solving LOG and PAGELATCH_UP contention. In the absence of LOG and/or PAGELATCH_UP contention, the only other option is to partition the table/index in question in order to create multiple caches (the caches are per-index).

SQL Waits

- LATCH_DT
 - Destroy Latch
- LATCH_EX
 - Exclusive Latch
- LATCH_KP
 - Keep Latch
- LATCH_NL
 - Null Latch
- LATCH_SH
 - Shared Latch
- LATCH_UP
 - Update Latch

SQL Waits

- LCK_x
 - Possible transaction management issue.
 - 1. For shared locks, check Isolation level for transaction.
 - 2. Keep transaction as short as possible
 - See SQL Locks perf counters
 - 1. SQLServer Lock wait time (ms)
 - 2. SQLServer Locks: Average Wait Time(ms)
 - 3. SQLServer Locks: Lock requests/sec
 - 4. SQLServer Locks: Lock Waits/sec
 - Hint:
 - 1. check for memory pressure, which causes more physical IO, thus prolonging the duration of transactions and locks.
 - 2. check for appropriate indexing
 - 3. check plan selection – avoid scans where possible

SQL Waits

- LCK_M_BU - Bulk update lock
- LCK_M_IS - Intent Share lock
- LCK_M_IU - Intent Update Lock
- LCK_M_IX - Intent Exclusive lock
- LCK_M_RIn_NL - Range Intent Null Lock
- LCK_M_RIn_S - Range Intent Shared lock
- LCK_M_RIn_U - Range Intent Update lock
- LCK_M_RIn_X - Range Intent Exclusive lock
- LCK_M_RS_S - Range Shared Shared (Key-Range) lock
- LCK_M_RS_U - Range Shared Update (key-range) lock

SQL Waits

- LCK_M_RX_S - Range Exclusive shared (key-range)
- LCK_M_RX_U - Range Exclusive update (key-range) lock
- LCK_M_RX_X - Range Exclusive Exclusive (key-range)
- LCK_M_S - Shared Lock:
- LCK_M_SCH_M - Modify schema lock:
- LCK_M_SCH_S - Shared Schema (Stability) lock
- LCK_M_SIU - Share Intent Update lock
- LCK_M_SIX - Share Intent Exclusive lock
- LCK_M_U - Update lock.
- LCK_M_UIX - Update intent exclusive lock
- LCK_M_X - Exclusive lock

SQL Waits

- LOGMGR

- Waiting for write requests to the transaction log to complete. Identify disk bottlenecks, using PERF Counters, Profiler, ::fn_virtualfilestats and SHOWPLAN. Any of the following will reduce these waits:
 - 1. Adding additional IO bandwidth,
 - 2. Balancing IO across other drives
 - 3. Moving / Isolating the transaction log on its own drive
- See Disk perf counters:
 - 1. Disk sec/read
 - 2. Disk sec/write
 - 3. Disk queues
- See SQL Buffer Cache perf counters:
 - 1. Page Life Expectancy
 - 2. Checkpoint pages/sec
 - 3. Lazywrites/sec
- Check IoStallMS for tranlog1. select * from ::fn_virtualfilestats(dbid,file#)

SQL Waits

- MISCELLANEOUS
 - Catch all wait type
- NETWORKIO
 - Waiting on Network IO Completion.
 - Waiting to read or write to a client on the network
 - This can occur if a client is in the middle of sending packets to SQL Server, or when SQL writes data to a client and is waiting for an ACK.
 - Check NIC bandwidth.
 - 100mbits is preferable to 10mbs.

SQL Waits

OLEDB

- Common causes are SQL Server is waiting for client application to send data. Some examples include:
 - 1. BULK INSERT
 - 2. CONVERT (6.5 to 2000)
 - 3. Full text
 - 4. Linked server calls incl. four part name calls, remote procedure calls, openquery, openrowset etc.
 - 5. Queries that access virtual tables - implemented as OLEDB rowset providers.
 - 6. Heavy use of Profiler
- Check placement of client app including any file input read by the client and SQL Server data and log files. See PERFMON disk secs/read & disk secs/write. If disk secs/read are high, you may add additional IO bandwidth, balance IO across other drives, or move / isolate the database and transaction log on its own drives
- Inspect TSQL code for RPC, Distributed (Linked Server) & Full Text Search. While SQL server supports these type queries, they are sometimes performance bottlenecks.
- To get the SQL Statement involved in OLEDB waits, Select virtual table master..sysprocesses as follows:
 - a. SQL2000 Service Pack 3 Only


```
DECLARE @Handle binary(20)
SELECT @Handle = sql_handle FROM sysprocesses WHERE waittype = 0x0042
SELECT * FROM ::fn_get_sql(@Handle)
```
 - b. SQL2000 RTM, SP1, SP2 – limited to 255 characters dbcc inputbuffer (spid)

SQL Waits

- PAGEIOLATCH_X
 - Latches are short term synchronization objects.
 - used to synchronize access to buffer pages.
 - PageIOLatch is used for disk to memory transfers.
 - If this is significant in percentage, it normally suggests disk IO subsystem issues. Check disk counters.

SQL Waits

- PAGEIOLATCH_DT - Page destroy latch
- PAGEIOLATCH_EX - Page latch exclusive
- PAGEIOLATCH_KP - Page latch keep
- PAGEIOLATCH_NL - Page latch null
- PAGEIOLATCH_SH - Page latch shared
- PAGEIOLATCH_UP - Page latch update

SQL Waits

- PAGELATCH_X
 - Latches are short term light weight synchronization objects.
 - Latches are not held for the duration of a transaction.
 - Typical latching operations during row transfers to memory, controlling modifications to row offset table, etc.
 - Consequently, the duration of latches is normally sensitive to available memory.
 - If this is significant in percentage, it normally indicates cache contention.

SQL Waits

- PAGELATCH_DT - Page latch
- PAGELATCH_EX - Page latch exclusive
 - Contention can be caused by issues other than IO or memory performance, for example, heavy concurrent inserts into the same index range can cause this type of contention.
 - If a lot of inserts need to be placed on the same page they are serialized using the latch.
 - A lot of inserts into the same range can also cause page splits in the index which will hold onto the latch while allocating a new page (this can take a while).
 - Any read accesses to the same range as the inserts would also conflict on the latches.
 - The solution in these cases is to distribute the inserts using a more appropriate

SQL Waits

- PAGELATCH_KP - Page latch keep
- PAGELATCH_NL - Page latch null
- PAGELATCH_SH - Page latch shared
 - Contention can be caused by issues other than IO or memory performance, for example, heavy concurrent inserts into the same index range can cause this type of contention.
 - If a lot of inserts need to be placed on the same page they are serialized using the latch.
 - A lot of inserts into the same range can also cause page splits in the index which will hold onto the latch while allocating a new page (this can take a while).
 - Any read accesses to the same range as the inserts would also conflict on the latches.
 - The solution in these cases is to distribute the inserts using a more appropriate

SQL Waits

- **PAGELATCH_UP**
 - Page latch Update is used only for allocation related pages, and contention on it is often a sign that more files are needed.
 - With multiple files, allocations can be distributed across multiple files thus reducing demand on the per-file data structures stored on these pages.
 - The contention is not IO performance, but rather internal allocation contention to access the pages – adding more spindles to a file or moving the file to a faster disk will not help, nor will adding more memory.

SQL Waits

- PAGESUPP

- Waits for parallel page supplier.
- Possible disk bottleneck; Any of the following will reduce these waits:
 - 1. Adding additional IO bandwidth,
 - 2. Balancing IO across other drives
 - 3. Reducing IO with proper indexing
 - 4. Check for bad query plans

SQL Waits

- PIPELINE_INDEX_STAT
 - PIPELINE waittypes added to allow one user to perform multiple operations such as writes to log cache on behalf of himself as well as other users who are waiting for same operation. It does all log writes in single operation.

SQL Waits

- PSS_CHILD - Waiting on Asynch thread
- RESOURCE_QUEUE Internal Use only
- RESOURCE_SEMAPHORE
 - COMMON for DSS like workload & large queries such as hash joins; must wait for memory quota (grant) prior to execution.
 - See SQL Memory Mgr perf counters
 - 1. Memory Grants Pending
 - 2. Memory Grants Outstanding
- SHUTDOWN
 - Shutdown without specifying NOWAIT,
 - waits for other users to logout before shutdown completes
 - Monitor SQL Statistics: User ConnectionsTo expedite shutdown you can:
 - 1. SHUTDOWN WITH NOWAIT
 - 2. Use SQL Kill command to terminate user connections.
- SLEEP - Internal Use only
- TEMPOBJ - Dropping a global temp object that is being used by others.
- TRAN_MARK_DT - Transaction latch – destroy
- TRAN_MARK_EX - Transaction latch – Exclusive
- TRAN_MARK_KP - Transaction latch - Keep page
- TRAN_MARK_NL - Transaction latch – Null
- TRAN_MARK_SH - Transaction latch – Shared
- TRAN_MARK_UP - Transaction latch - Update

SQL Waits

- UMS_THREAD - Batch waiting on a worker thread to free up (or batch waiting to get a worker thread to run it)
 - If this is a high percentage, you can increase the number of worker threads from the default of 255. The maximum is 1024.
- WAITFOR - BWaitforCheck for waitfor delay in TSQL codeInspect TSQL code for “waitfor delay” statement
- WRITELOG - Waiting for write requests to the transaction log to complete. Identify disk bottlenecks, using PERF Counters, Profiler,::fn_virtualfilestats and SHOWPLA
 - Any of the following will reduce these waits:
 - 1. Adding additional IO bandwidth,
 - 2. Balancing IO across other drives
 - 3. Moving / Isolating the transaction log on its own drive
 - See Disk perf counters:
 - 1. Disk sec/read
 - 2. Disk sec/write
 - 3. Disk queues
 - See SQL Buffer Cache counters:
 - 1. Page Life Expectancy
 - 2. Checkpoint pages/sec
 - 3. Lazywrites/secCheck IoStallMS for tranlog1. select * from ::fn_virtualfilestats(dbid,file#)

SQL Waits

- PIPELINE_LOG

- PIPELINE waittypes added to allow one user to perform multiple operations such as writes to log cache on behalf of himself as well as other users who are waiting for same operation.
- Does in single operation.

- PIPELINE_VLM

- PIPELINE waittypes added to allow one user to perform multiple operations such as writes to log cache on behalf of himself as well as other users who are waiting for same operation.
- Does in single operation.