

SQLskills Immersion Event

IE2: Performance Tuning

Module 4: SQLOS

Jonathan Kehayias
Jonathan@SQLskills.com



Overview

- Understanding Windows scheduling
- Server hardware and NUMA
- The purpose of SQLOS
 - CPU Scheduling
 - Memory allocation
- Edge cases for tweaking
- DMV monitoring and troubleshooting (basics)



2

© SQLskills, All rights reserved.
<http://www.SQLskills.com>



Windows Scheduling

- **Windows NT+ uses a multi-level feedback queue**
 - Give preference to short jobs
 - Give preference to I/O-bound processes
 - Separate processes into categories based on their need for the processor
 - Threads execute within their quantum
 - Once the quantum expires the thread must wait for Windows to schedule them again
 - All threads with the same priority are treated equal
 - Threads with highest priority are scheduled first
 - If a higher-priority thread becomes available to run, the system stops executing lower-priority threads and assigns a full time-slice to the higher-priority thread

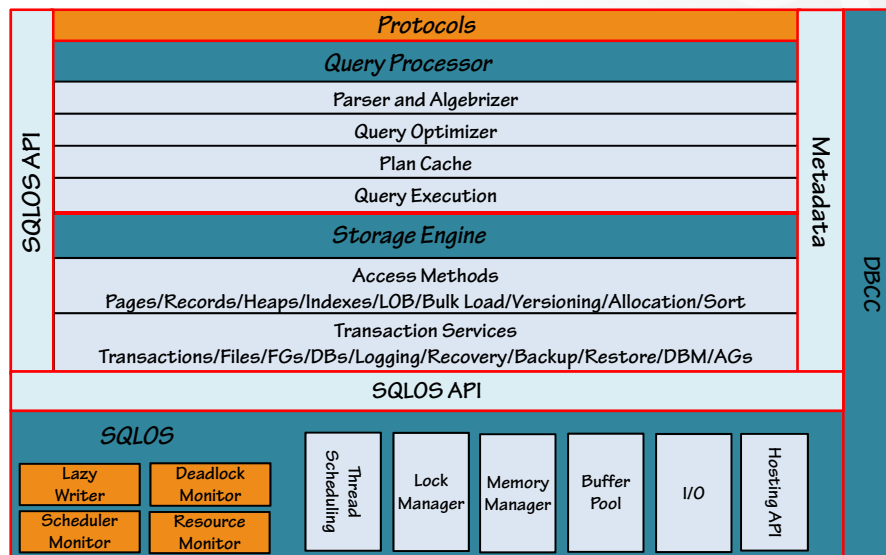
Windows Scheduling

- **Threads priorities are dynamic and can be changed by Windows based on a number of factors:**
 - Normal-priority processes in the foreground are increased so that the priority is greater than any background process's thread priority
 - If a window receives input, such as timer messages, mouse messages, or keyboard input, the scheduler boosts the priority of the thread that owns the window
 - When the wait conditions for a blocked thread are satisfied, the scheduler boosts the priority of the thread
- **Low-priority threads can be randomly boosted to ensure they execute critical code sections and exit appropriately through priority inversion**

Context Switching

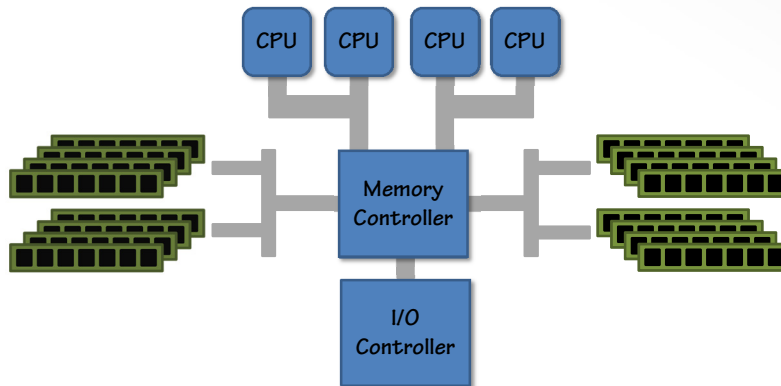
- The scheduler maintains a queue of executable threads for each priority level known as ready threads
- When a processor becomes available, the system performs a context switch by:
 - Saving the context of the thread that just finished executing
 - Moving the thread that just finished executing at the end of the queue for its priority
 - Find the highest-priority queue that contains ready threads
 - Remove the thread at the head of the queue, load its context, and execute it
- The most common reasons for a context switch are:
 - A thread's quantum has elapsed
 - A thread with a higher priority has become ready to run
 - A running thread needs to wait and relinquishes the remainder of its time slice

Server Architecture

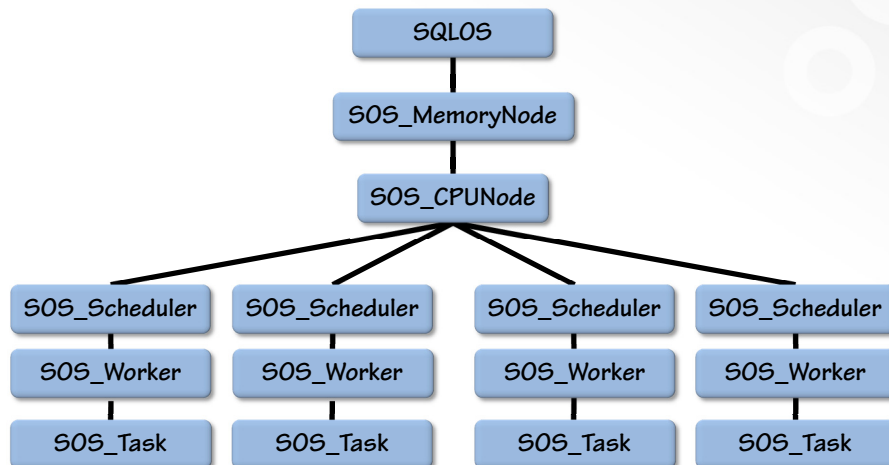


Traditional SMP Systems

- Processors share front-side bus or cross bar
- Memory access uniform across all CPU cores



SQLOS Under SMP

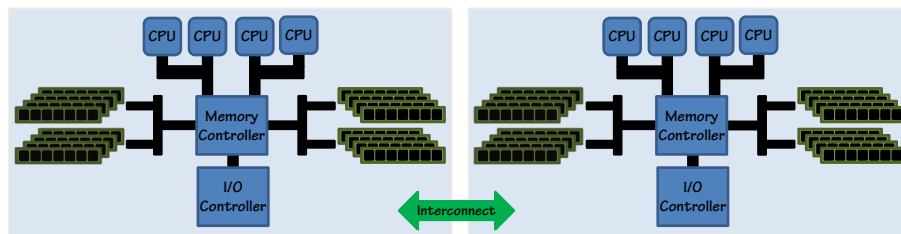


The Case for Non-Uniform Memory Access (NUMA)

- Modern CPUs operate faster than the memory to which they are attached
- Only one processor can access memory at a time for coherency
- As the number of processor cores increases, the cross bar/front-side bus becomes the system bottleneck starving multiple processors for memory access

Traditional NUMA

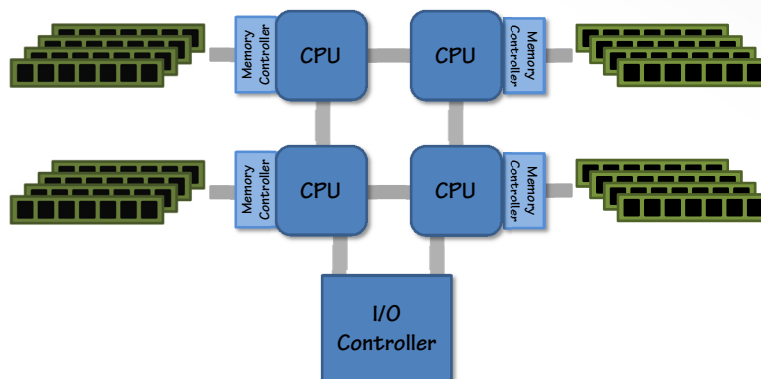
- Traditionally NUMA configurations required special hardware (e.g. IBM xSeries Servers)
- Nodes were physically segregated servers connected by special interconnect
 - Two SMP servers running together as a single system
 - Required consideration for I/O and controller placement

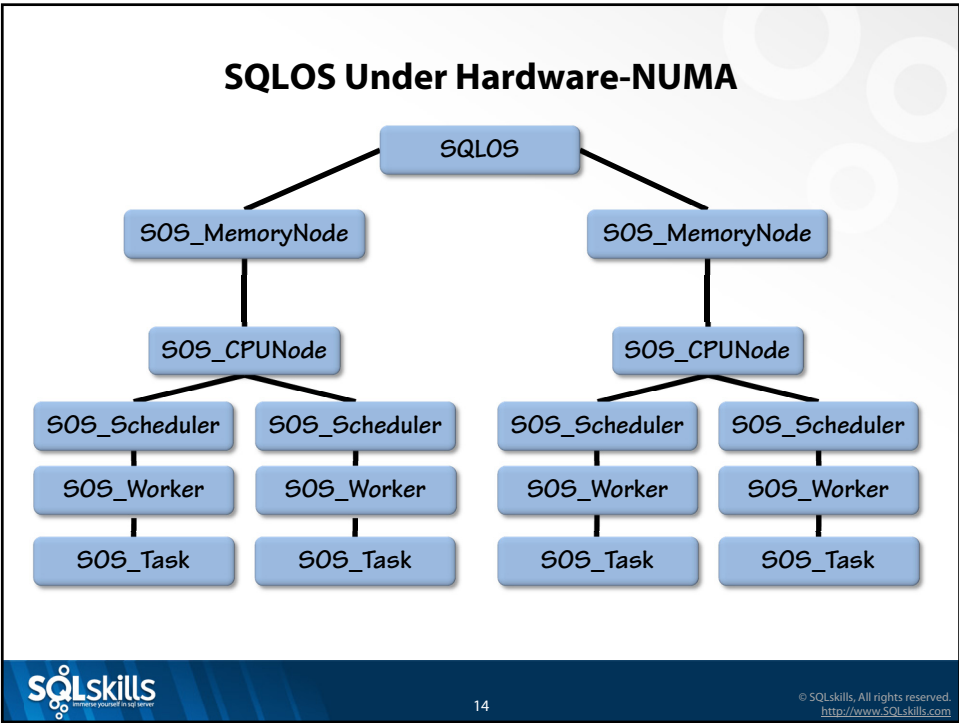
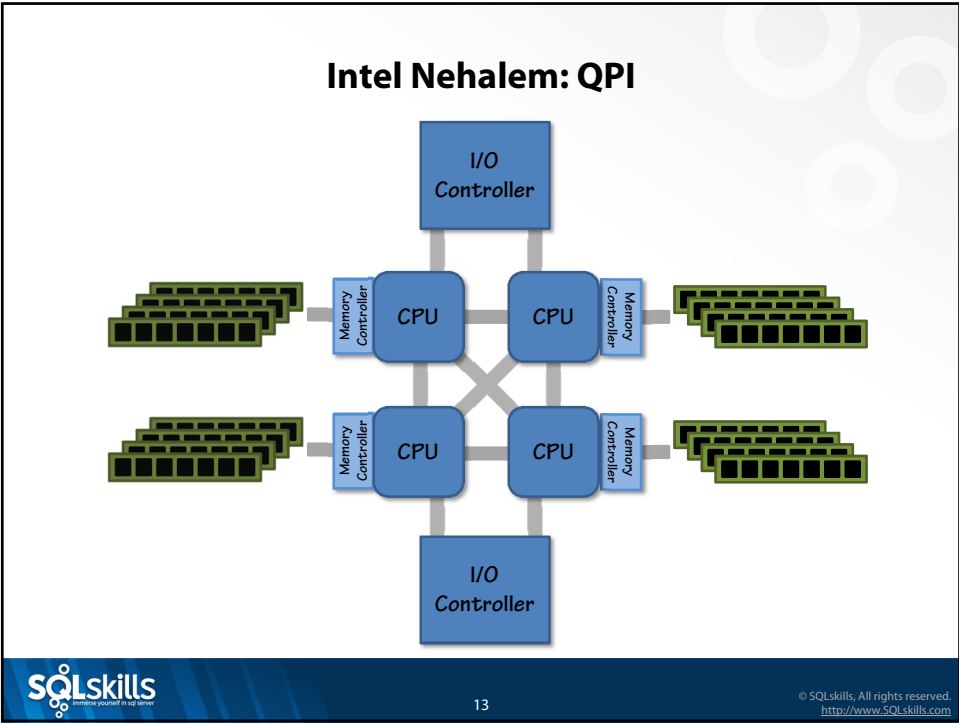


Modern NUMA

- Memory controllers are built into the processor die
- Each processor socket presents itself as an individual NUMA node to the OS unless node interleaving is enable in the BIOS
- Inter-node communication paths are extremely fast and foreign-memory allocations have a lower impact on overall performance

AMD NUMA: HyperTransport





Is NUMA Important Still?

- **Even with enhancements, NUMA still matters**
 - Separate lazywriter process per NUMA node
 - Impact to checkpoint operations
 - Latency with remote memory still matters
- **Max Degree of Parallelism**
 - Number of physical cores per NUMA node
 - Prevent cross-node scheduling of parallel processing
 - Leverage Level-1 cache on die to prevent translation look-aside buffer (TLB) misses
 - The TLB is a CPU cache that memory management hardware uses to improve virtual address space translation speed
 - The TLB maps virtual and physical address spaces to reduce the number of page table lookups performed, which read multiple memory areas to compute the address needed

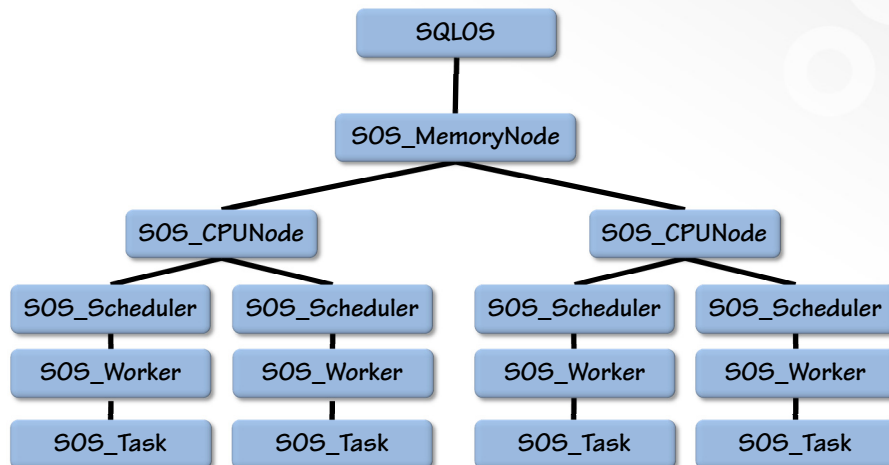
Node Interleaving

- **The option to run the server in an SMP configuration**
- **Sufficiently Uniform Memory Architecture (SUMA) configuration**
 - Memory mapped in 4KB regions to nodes in round-robin even fashion instead of as a single, sequential block under NUMA
- **Beneficial to certain workloads, but should not generally be used with SQL Server**
- **Trace Flag 8015 disables NUMA detection in SQLOS**
 - Trace flag 8048 may be a better solution for certain workloads since this changes memory heap partitioning in SQLOS under NUMA from node based to per-CPU and may be beneficial on servers with 8+ cores per socket
 - <http://blogs.msdn.com/b/psssql/archive/2011/09/01/sql-server-2008-2008-r2-on-newer-machines-with-more-than-8-cpus-presented-per-numa-node-may-need-trace-flag-8048.aspx>

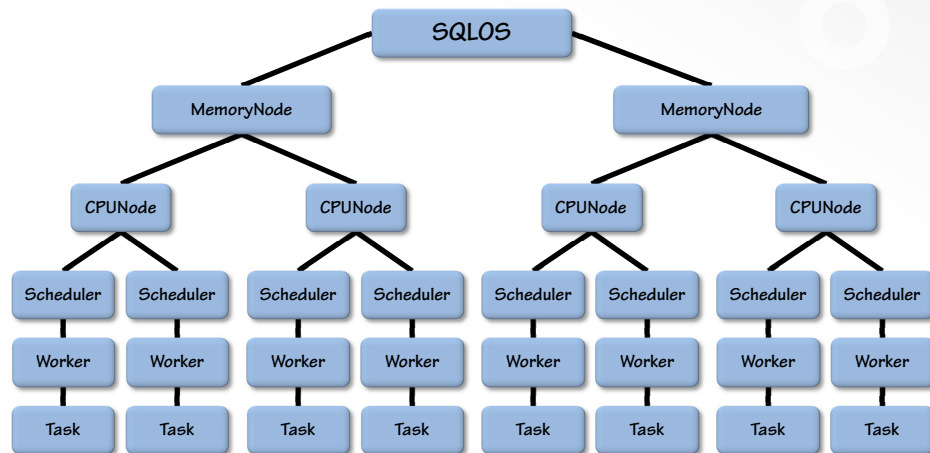
Soft NUMA

- **Creates a custom NUMA configuration independent of hardware NUMA configuration**
 - Registry settings control Soft NUMA configuration
 - Can provide greater performance, scalability, and manageability on SMP as well as on real NUMA hardware for specific workloads
 - ETL world record set using Soft NUMA configuration
 - Allows the ability to mask independent nodes to specific TCP ports for increased performance
- **Soft-NUMA nodes can only include processors from a single hardware-NUMA node**

SQLOS Under SMP with Soft-NUMA



SQLOS Under Hardware-NUMA with Soft-NUMA



SOS Scheduler

- **Provides a thin layer between SQL Server and the OS**
 - Provides cooperative scheduling and I/O processing for SQL Server instead of preemptive scheduling used by Windows
 - Cooperative scheduling: tasks have an execution quantum (duration of time) and voluntarily yield the CPU to other tasks
 - Preemptive scheduling: highest priority task gets the CPU
- **Two modes of execution: thread or fiber**
 - Thread is default
 - Fiber can provide performance boost for specific usage scenarios
 - Rarely used in production environment
 - CLR not supported under fiber mode

SOS Workers (Threads)

- **Default configuration of 0**
 - 32-bit SQL Server
 - Actual number is: $((\text{NumberOfSchedulers} - 4) * 8) + 256$
 - 64-bit SQL Server
 - Actual number is: $((\text{NumberOfSchedulers} - 4) * 16) + 512$
- **SELECT max_workers_count**
- **FROM sys.dm_os_sys_info**
- **Edge cases for changing default**
 - Database mirroring on 32-bit servers
 - Principal server uses 1 global thread, 2 threads per mirrored database
 - Mirror server uses 1 global thread, 2 threads per mirrored database and 1 additional thread per mirrored database for every 4 processors

Scheduler Monitor

- **Per node monitoring for issues every 5 seconds**
 - 17883: non-yielding task, single scheduler
 - 17887: I/O completion stall, single scheduler
 - 17884: no work progressing, all schedulers
 - 17888: 50% of same resource, all schedulers
- **Ring buffer entries using sys.dm_os_ring_buffers and Extended Events**

Scheduling DMVs

- `sys.dm_os_schedulers`
- `sys.dm_io_pending_io_requests`
- `sys.dm_os_workers`
- `sys.dm_os_tasks`
- `sys.dm_os_waiting_tasks`
- `sys.dm_os_wait_stats`

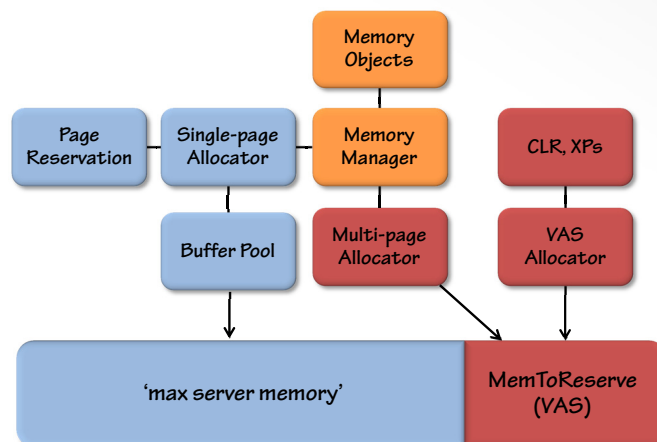
Demo

Scheduling DMVs

Memory Manager SQL Server 2008

- **Single-page allocator**
 - Single 8KB memory-page allocations for the buffer pool
 - Data cache, plan cache, execution memory
- **Multi-page allocator**
 - Memory allocations larger than a single 8KB page that are outside of the buffer pool
 - Thread stack, SQLCLR, linked servers, OLE automation, SQLXML, extended stored procedures
- **VAS allocator**
 - Allocations directly from VirtualAlloc API
 - SQLCLR, extended stored procedures

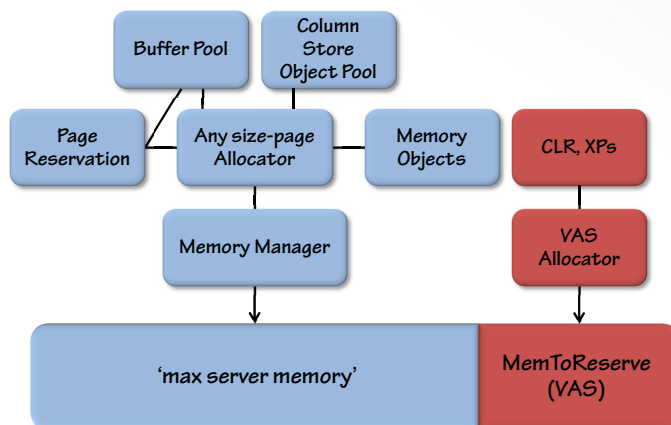
Memory Manager SQL Server 2008



Memory Manager SQL Server 2012

- Memory manager rewritten with a new “any size page” allocator instead of separate single-page and multi-page allocators
 - The ‘max server memory’ sp_configure option is closer to the actual maximum for the instance process
 - Does not account for SQLCLR memory allocations from VAS, or providers that execute out of process
- AWE has been removed from SQL Server
- Consistent out-of-memory handling and management across all the internal components
- New optimized Column Store Object Pool for managing Column Store index memory requirements

Memory Manager SQL Server 2012



Memory Nodes

- Represent the memory attached to all CPUs in SMP configuration or to a single NUMA node
- Supports the memory allocators already discussed as well as:
 - Reserved-page allocator: DAC memory
 - Large-page allocator: used for buffer pool and plan cache when Large Page support is enabled by TF 834
- DMVS and DBCC MEMORYSTATUS provide information about the memory nodes in the system
 - sys.dm_os_memory_nodes

Memory Clerks

- Track memory usage for a specific component
- Receive notifications about memory state changes and respond to pressure
 - SQLCLR garbage collection hooks into this mechanism
- Utilize memory node allocators for memory allocation
- DMVS and DBCC MEMORYSTATUS provide information about the memory clerks in the system
 - sys.dm_os_memory_clerks

Memory Caches

- **Caches are memory clerks**
- **Buffer pool**
 - Data page cache
- **Cache Store**
 - Generic cache framework
 - Procedure cache and system rowset cache
- **User Store**
 - Generic cache framework
 - Schema manager, security, and metadata caches
- **sys.dm_os_memory_cache_counters**

Cache Store

- **Generic cache mechanism with storage**
 - sys.dm_os_memory_cache_hash_tables
 - sys.dm_os_memory_cache_entries
- **Implements size control by utilizing LRU (clock) algorithm**
 - External clock hand controls memory consumed by all caches
 - Internal clock hand controls memory consumed by a given cache
 - Each tick of the clock hand decrements the cost for the cache entries by one
 - Cache entries with a zero cost are removed from the cache
 - When a cache entry is reused its cost value is reset
 - sys.dm_os_memory_cache_clock_hands
- **User Store is exactly the same as Cache Store but doesn't have storage**

Buffer Pool: General

- **Largest Cache Store (and memory consumer) for SQL Server**
- **Single 8K page allocations**
 - Data page cache
 - Procedure cache on 64-bit instances as well
- **Reserves virtual address space upfront**
 - Leaves space for thread stacks plus 256MB (modified by -g) on 32-bit systems
- **Memory is committed and mapped on demand**

Buffer Pool Under NUMA

- **Has a single memory clerk that spans NUMA nodes**
 - DBCC MEMORYSTATUS incorrectly reports memory allocations under NUMA
 - Use Buffer Node performance counters to track NUMA memory allocations
- **Separate lazywriter per NUMA node**
- **Max server memory divided evenly across nodes**
 - 32GB max server memory with 8 NUMA nodes allocates 4GB per node
 - Taking a node offline results in its memory being redistributed to remaining nodes and foreign page allocations

Lazywriter

- Works like a clock to sweep the cache, starting at buffer zero and sequentially sweeping the BUF structures with each tick
- Each tick looks at 16/32 buffers to determine if the Time of Last Access (TLA) is below the current threshold
 - If the page is clean it returns the buffer to the free list
 - If the page is dirty WriteMultiple is called to gather pages in physical order that are also dirty before performing the write to disk and returning the buffer to the free list
 - SQL Server 2005 gathers up to 16 contiguous pages after the current page
 - SQL Server 2008+ gathers up to 32 contiguous pages before or after the current page
- Helper routines (HelpLazyWriter) allow any worker to perform a lazywriter tick while allocating memory if the number of buffers on the free list is low

Memory Management (32-Bit Processes)

- User mode virtual Address Space (VAS) limited to 2GB (2^{32})
 - Amount of memory addressable through calls to VirtualAlloc() to an application in Windows
- PAE: Physical Address Extension
 - Allows access to memory above the 4GB address space (4GB to 64GB)
- AWE: Address Windowing Extensions
 - Switches from a 32-bit pointer to a virtual 36-bit pointer
 - Allows an application to quickly manipulate physical memory greater than 4GB up to 64GB RAM
 - Limits /3GB tuning to under 16GB RAM due to reduced PTE entries to map memory in kernel mode VAS
 - Removed from SQL Server 2012

Memory Management (32-Bit Processes: /3GB and /PAE)

- **/3GB tuning**
 - For servers with < 16GB, the /3GB boot.ini option can be used to increase the user mode VAS by 1GB
 - The /USERVA boot.ini option can be used to tune the amount of Page Table Entries (PTEs) for system stability
- **-g SQL Server startup parameter**
 - Used to reserve additional startup virtual address space (VAS) also known as the MemToLeave (for the internal function used by SQL Server in 32 bit operation) to reserve VAS for use by the sqlservr.exe process for non-buffer pool allocations
 - Linked servers, SQLCLR, OLE automation, SQLXML, large execution plans, extended stored procedures, etc.

Memory Management (64-Bit Processes)

- User mode VAS increased dramatically from 2GB to 16 exabytes (2^{64}) but limited to 8TB virtually by Windows
- Eliminates the need for specialized configuration for SQL Server to allocate the available memory from Windows

Lock Pages in Memory

- Lock Pages in Memory allows SQL Server to allocate memory using `AllocateUserPhysicalPages()` instead of `VirtualAlloc()`
- **Required for AWE on 32-bit SQL Servers**
- **Prevents paging of the buffer pool on 64-bit instances**
 - Does not require configuration of “AWE enabled” in SQL Server
 - TF 845 required for Standard Edition instances
- **Detected at startup only**

Large Pages

- **32-bit: the OS page is 4KB**
- **64-bit: can use large pages (2-16MB)**
 - Helps avoid translation look-aside buffer (TLB) misses
- **Requires**
 - Enterprise Edition
 - Locked pages in memory
 - TF 834
 - <http://support.microsoft.com/kb/920093>
- **All memory allocated at startup and must be contiguous**
- **Recommended for virtual servers implemented on VMware to minimize virtualized-TLB lookups**
 - http://www.vmware.com/files/pdf/sql_server_virt_bp.pdf
 - Based on 3.5 infrastructure, now out-of-date

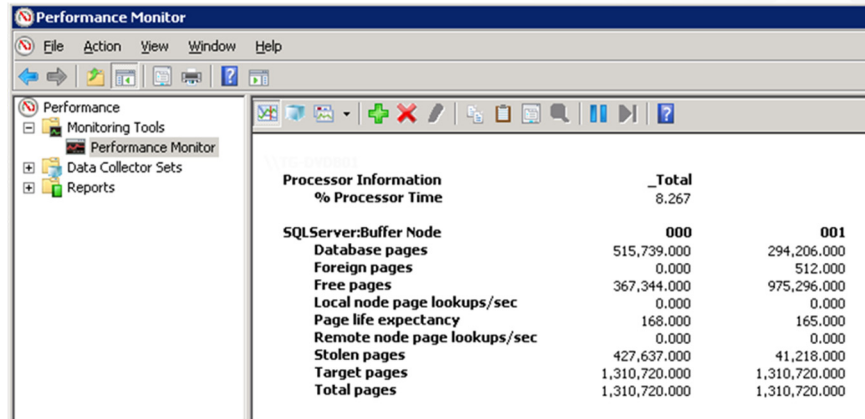
Server Memory Configuration (Max Server Memory)

- Required on 64-bit servers to prevent memory pressure and out of memory conditions
- Only applies to the buffer pool, does not set the total amount of memory used by SQL Server
 - Additional memory for thread stack and multipage allocators
 - Memory for thread stack = (max worker threads) x (stack size)
 - X32 = 512K; x64 = 2MB; ia64 = 4MB
 - (Total system memory) minus (memory for thread stack) minus (OS memory requirements ~ 2-4GB) minus (memory for other applications)
 - How I typically do it: (1GB for the OS + 1GB for each 4GB between 4-16GB + 1GB for each 8GB above 16GB in the server)
- Best to monitor Memory\Available Mbytes > 150-300MB and adjust as necessary

Monitoring Memory Usage

- Performance counters
 - sys.dm_os_performance_counters
 - Performance Monitor
- DMVs
 - Detailed information about memory usage by SQL Server at all levels
- DBCC MEMORYSTATUS
 - <http://support.microsoft.com/kb/907877>
- Ring buffers
 - sys.dm_os_ring_buffers
 - History of memory notifications

Monitoring Memory Usage (PerfMon Buffer Node Under NUMA)



Processor Information		
% Processor Time		_Total
		8.267
SQLServer:Buffer Node	000	001
Database pages	515,739.000	294,206.000
Foreign pages	0.000	512.000
Free pages	367,344.000	975,296.000
Local node page lookups/sec	0.000	0.000
Page life expectancy	168.000	165.000
Remote node page lookups/sec	0.000	0.000
Stolen pages	427,637.000	41,218.000
Target pages	1,310,720.000	1,310,720.000
Total pages	1,310,720.000	1,310,720.000

Resource Monitor

- **Provides a mechanism to respond to memory pressure**
 - Implemented as a regular task in the system using its own dedicated, hidden scheduler
 - Invokes garbage collection in SQLCLR
- **Outputs information to the OS ring buffers**
 - ring_buffer_type="RING_BUFFER_RESOURCE_MONITOR"

Memory DMVs (System Information)

- (Reference slide)
- sys.dm_os_sys_info
- sys.dm_os_sys_memory
- sys.dm_os_loaded_modules

Memory DMVs (SQL Server)

- (Reference slide...)
- 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

Demo

Memory monitoring with DMVs and DBCC MEMORYSTATUS

Review

- Understanding Windows scheduling
- Server hardware and NUMA
- The purpose of SQLOS
 - CPU Scheduling
 - Memory allocation
- Edge cases for tweaking
- DMV monitoring and troubleshooting (basics)

Questions?

