SQLskills Immersion Event

IEPTO2: Performance Tuning and Optimization

Module 10: SQLOS Memory Management and Memory Performance Tuning

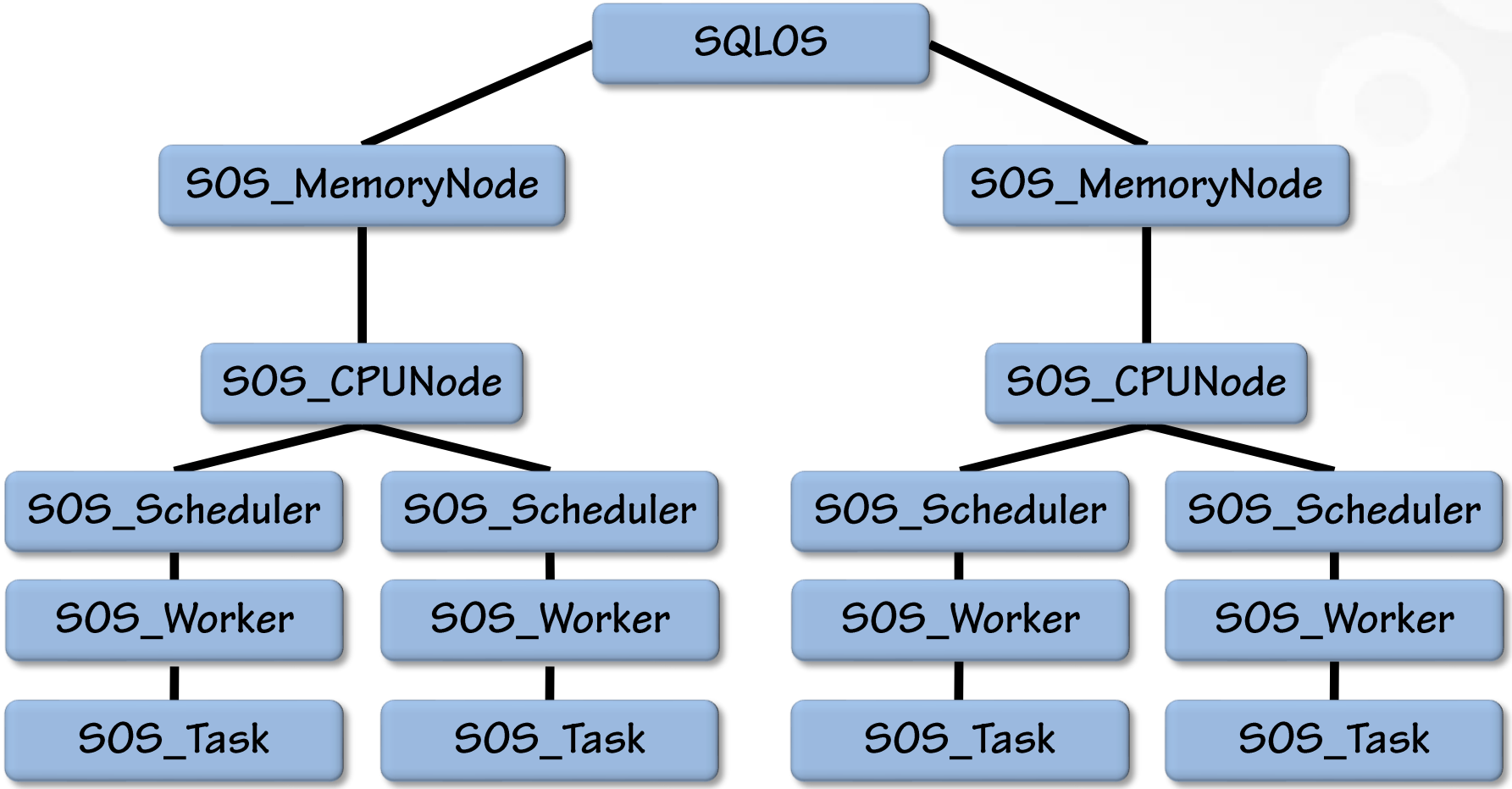
Jonathan Kehayias

Jonathan@SQLskills.com

Overview

- SQLOS Memory Manager and components
- Configuration options for SQL Server
- How to identify memory issues
- Internal memory pressure
- External memory pressure
- Virtual address space issues
- Resource semaphores
- DBCC commands

SQLOS Under Hardware-NUMA

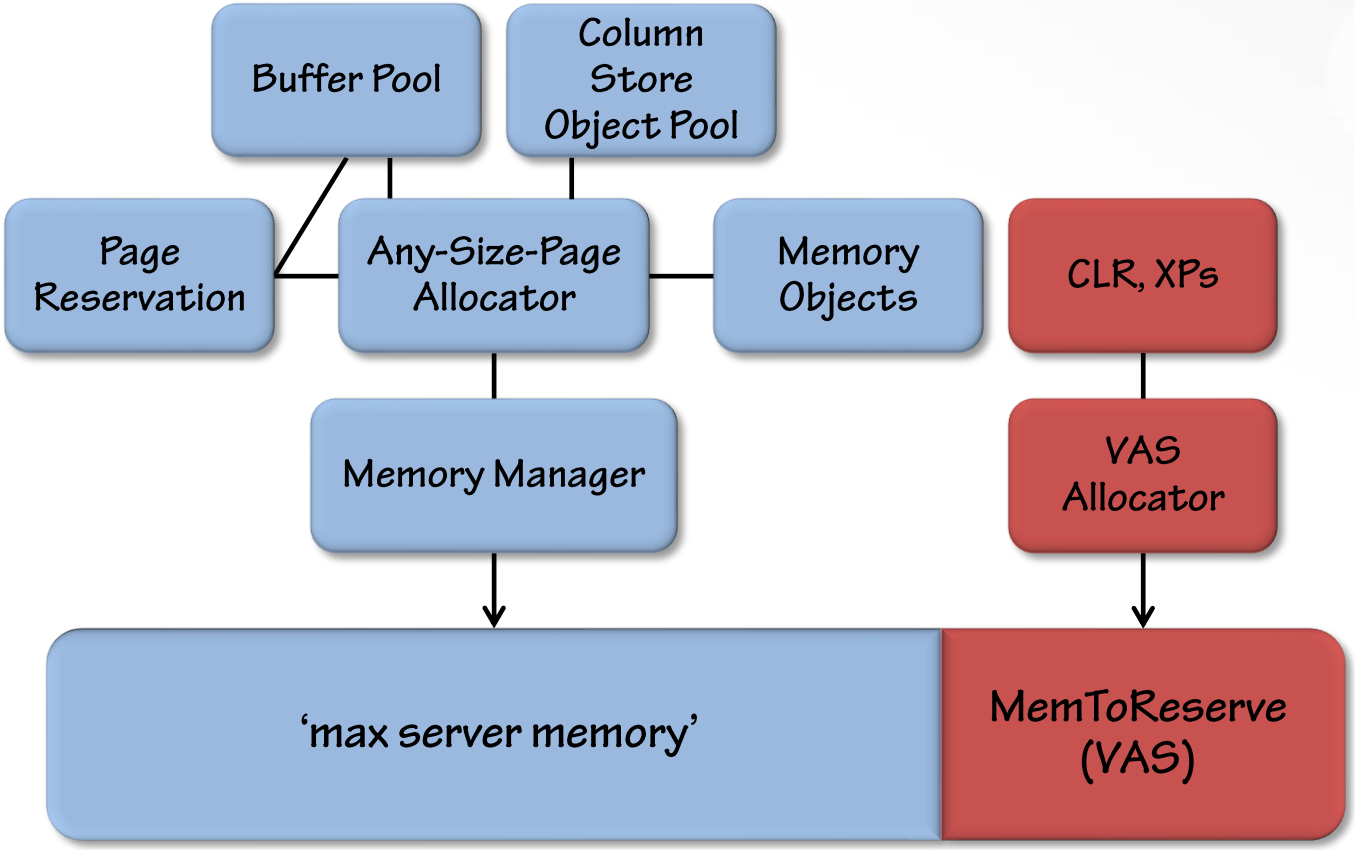


SQLOS Memory Manager

- **Memory manager rewritten in SQL Server 2012 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 2012 and higher**
- **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**

SQLOS Memory Manager

SQL Server 2012 Onwards



Memory Limits by Edition

- **64GB RAM limit for SQL Server 2008 R2 and 2012 Standard Edition**
- **128GB RAM limit for SQL Server 2014 Standard Edition**
- **128GB RAM for SQL Server 2016 Standard Edition SP1 buffer pool**
  - 32GB additional for In-Memory OLTP
  - 32GB additional for Column Store
- **OS maximum for Enterprise Edition**
  - Windows Server 2016 – up to 24TB
  - Windows Server 2012R2 – up to 4TB

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
  - 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 8KB page allocations**
  - Data page cache
  - Procedure cache on 64-bit instances as well
- **Memory is committed and mapped on demand**
- **Reserves virtual address space during SQLOS startup on 32-bit**
  - Leaves space for thread stacks plus 256MB (modified by –g startup parameter)

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 lazy writer per NUMA node**
- **Max server memory divided evenly across nodes**
  - 32GB max server memory with 8 NUMA nodes allocates ~4GB per node
    - There are scenarios where this may not occur
  - Taking a node offline results in its memory being redistributed to remaining nodes and foreign page allocations

Lazy Writer

- **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 (per NUMA node)
  - 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
    - SQL Server 2016+ gathers up to 128 contiguous pages (http://bit.ly/1SV1Tct)
- **Helper routines (HelpLazyWriter) allow any worker to perform a lazy writer tick while allocating memory if the number of buffers on the free list is low**

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 Server instances**
- **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 (SQL 2008 R2 and earlier)
- **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 enabled
  - TF 834 – Buffer Pool (unsupported for Columnstore)
  - http://support.microsoft.com/kb/920093
  - TF 876 – Columnstore Object Pool only
- **All memory allocated at startup and must be contiguous**
- **May lead to high Kernel mode CPU issues for some workloads**

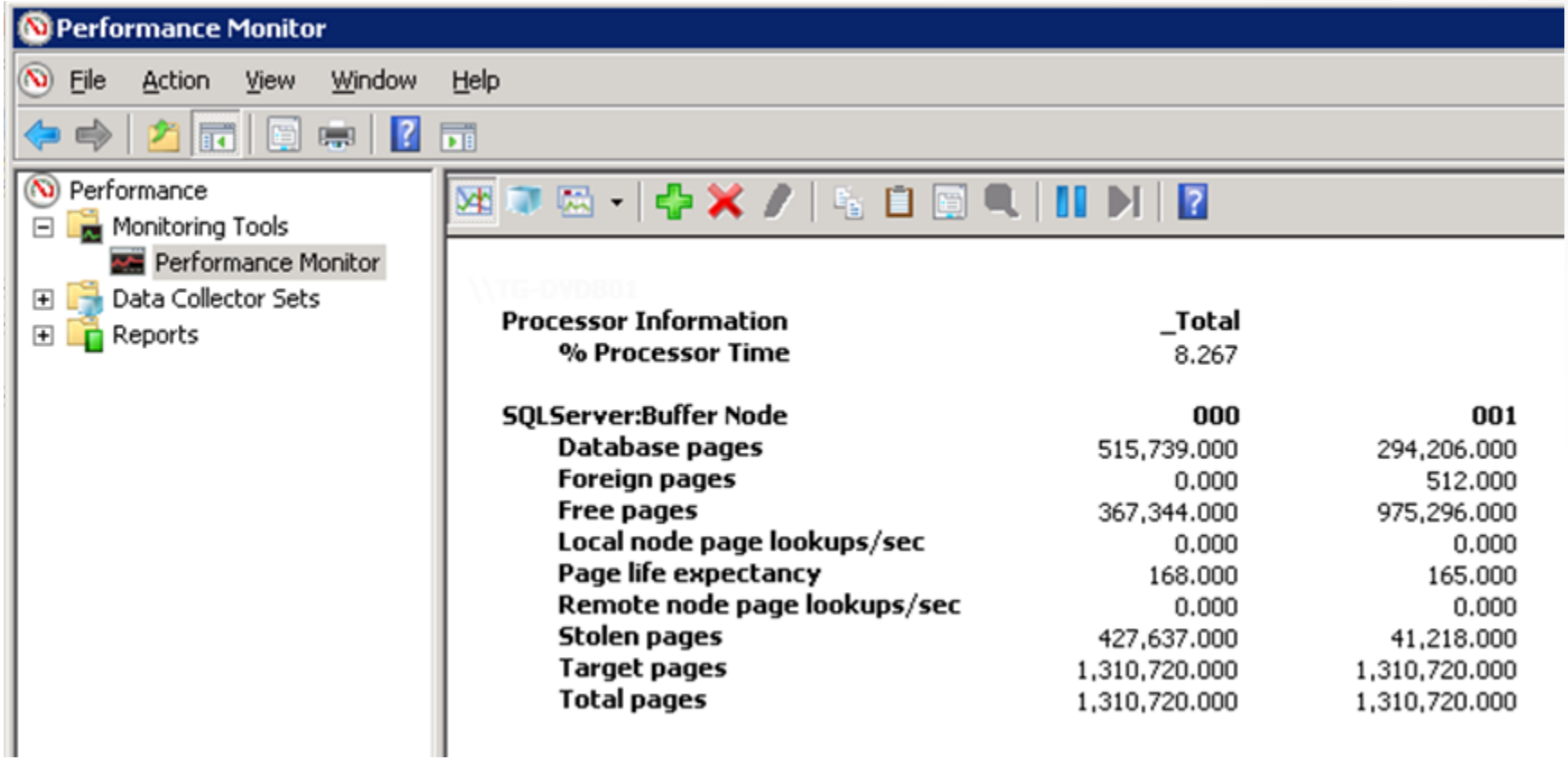
Server Memory Configuration
(Max Server Memory)

- **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 = 512KB; 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	_Total	
% Processor Time	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

How to Identify Memory Issues

- **Performance counters**
  - Memory\Available Mbytes
  - SQLServer:BufferManager\Page Life Expectancy
  - SQLServer:BufferManager\Lazy Writes/sec
  - SQLServer:BufferManager\Page Reads/sec
  - SQLServer:MemoryManager\Memory Grants Pending
  - SQLServer:MemoryManager\Memory Grants Outstanding
  - Physical Disk\Disk sec/Reads
- **Memory clerks**
  - sys.dm_os_memory_clerks
  - sys.dm_exec_cached_plans
  - sys.dm_os_ring_buffers
  - sys.dm_os_memory_cache_clock_hands

How to Identify Memory Issues

- **Wait types**
  - CMEMTHREAD
    - Waiting on a thread-safe memory object, generally for inserting or removing execution plans from cache
  - RESOURCE_SEMAPHORE
    - Waiting for a workspace memory grant for execution of a sort or hash operation
    - Error 8645 – a time out occurred while waiting for memory resources to execute the query. Rerun the query.
  - RESOURCE_SEMAPHORE_QUERY_COMPILE
    - Waiting for a memory grant during query compilation typically due to excessive plan compilations at a given point in time
  - RESOURCE_SEMAPHORE_SMALL_QUERY
    - Waiting for a memory grant from the small gateway during execution

Internal Memory Pressure
Buffer Pool

- **Symptoms**
  - SQLServer:Buffer Manager\Page Life Expectancy low
    - Or on NUMA, SQLServer: Buffer Node X\Page Life Expectancy low
  - SQLServer:Buffer Manager\Lazy Writes/sec > 0 outside of CHECKPOINT operations
  - SQLServer:Buffer Manager\Free List Stalls > 0
  - SQLServer:Buffer Manager\Page Reads/sec excessive
  - SQLServer:Memory Manager\Target Server Memory (KB) > Total Server Memory(KB)
  - Physical Disk\Disk sec/Reads > 30

Internal Memory Pressure
Buffer Pool

- **Causes**
  - High data cache churn inside of the SQL Server instance due to workload
  - Excessive table or index scans due to inefficient or missing indexes
  - Excess index fragmentation or indexes with low page density due to incorrect fillfactor specifications
  - Implicit conversion/probe residual issues in the workload causing excessive Scan operations
  - Plan cache bloat due to ad hoc non-parameterized workloads
- **Resolution**
  - Tune index usage in the environment
  - Perform analysis-based index maintenance and adjust fillfactor based on analysis of index fragmentation ratios
  - Remove implicit conversions/probe residual from column side of filtering predicates
  - Add memory

Internal Memory Pressure
Plan Cache

- **Symptoms**
  - SQLServer:Plan Cache\Cache Hit Ratio < 80
  - SQLServer:SQL Statistics\Compilations/sec to SQLServer:SQL Statistics\Batch Requests/sec exceeds 1 compile per 100 batch requests
  - High numbers of single use plans in cache
- **Causes**
  - Ad hoc, non-parameterized workload generated by the application
  - Not enough memory to support the workload being generated
  - Auto-Close database option set to true
  - 32-bit SQL Server memory limitations (the plan cache cannot use AWE-mapped memory)

Internal Memory Pressure
Plan Cache

- **Resolution**
  - Set Auto-Close database option to false to prevent cache store flushing from occurring
  - Use 'optimize for ad hoc workloads' sp_configure option
  - Convert application to use parameterized calls with fixed SqlDbType parameters
  - Convert application to use stored procedures
  - Add more memory to the server, or increase 'max server memory' in 64-bit environments
- **Reference**
  - https://www.SQLskills.com/blogs/kimberly/plan-cache-and-optimizing-for-adhoc-workloads/ (http://bit.ly/9MhQTt)
  - https://www.SQLskills.com/blogs/kimberly/plan-cache-adhoc-workloads-and-clearing-the-single-use-plan-cache-bloat/ (http://bit.ly/ae4RRI)

Internal Memory Pressure
Execution Memory

- **Symptoms**
  - SQLServer:Memory Manager\Memory Grants Outstanding low
  - SQLServer:Memory Manager\Memory Grants Pending high
  - RESOURCE_SEMAPHORE wait types on executing tasks
- **Causes**
  - Missing indexes resulting in hash and sort operations that consume large amounts of memory
  - Insufficient execution memory for current workload in the environment
  - Incorrect configuration of Resource Governor
- **Resolution**
  - If using Resource Governor, configure the resource pool memory limits higher for the workload
  - Tune queries to reduce the execution memory grant requirements

Internal Memory Pressure
Execution Memory (Semaphores)

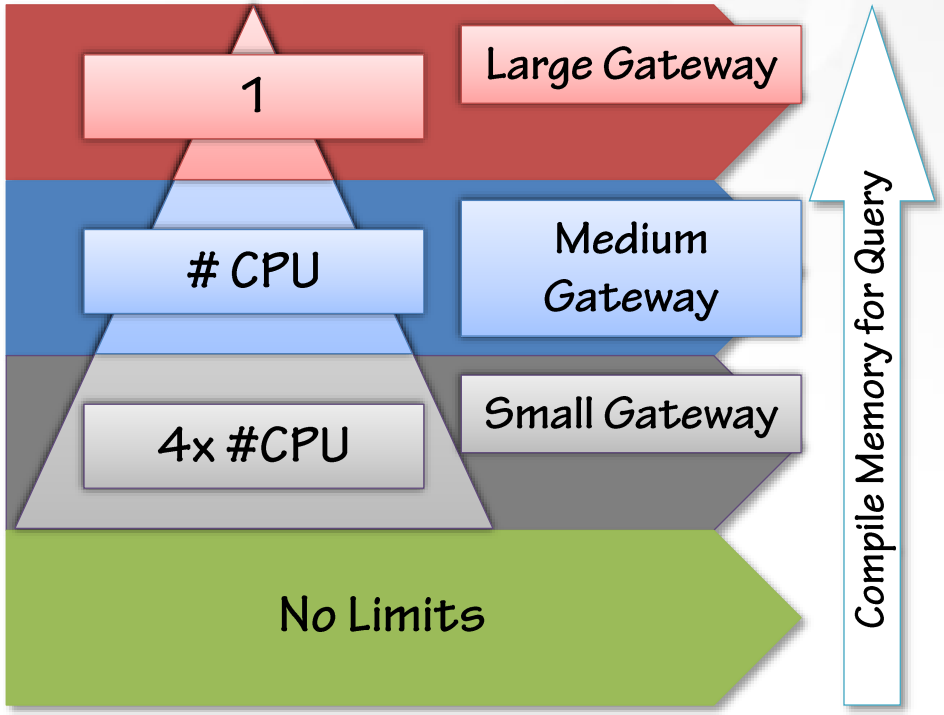
- **Track resource usage and availability**
- **Prevent race and deadlock conditions**
- **Library analogy**
  - 10 study rooms available
  - Librarian tracks the number of available rooms
  - Students request rooms for use and return them when finished
  - If no room is available students wait in a queue for the next available room

Internal Memory Pressure
Execution Memory (Semaphores)

- **Allows a query to reserve memory, only if enough is available**
- **For a new request, it first checks if any query is waiting**
  - If a query is already waiting, the new query is placed into a queue
    - The wait queue is implemented as FIFO queue
    - Small queries go into the Small Resource queue favoring < 5MB memory requirements
  - If it does not find any waiting query, it then checks available free memory
    - If it does not find enough free memory, then it puts the current query into the waiting queue
    - If it finds enough free memory, then the requested memory is granted and the query can start running
- **Grant attempts are made when there is no waiting query, or when an existing query returns reserved memory**
- **Resource Semaphore wakes up queries in the waiting queue when enough free memory becomes available**

Internal Memory Pressure
Compile Memory (Gateways)

- Queries begin compiling with no memory limitations
- Once a query passes the compile memory threshold for each gateway, it must obtain a semaphore from the gateway to continue executing
  - If the gateway is full the query will wait with RESOURCE_SEMAPHORE_QUERY_COMPILE until the memory is available at the gateway
  - If the query requires a larger gateway, it maintains its semaphore grant from the lower gateway(s) as well
- Small gateway has fixed threshold
  - x86 – 250,000 bytes
  - x64 – 380,000 bytes
- Medium and large gateways are dynamic based on load



- sys.dm_exec_query_optimizer_memory_gateways

Internal Memory Pressure
Compile Memory

- **Symptoms**
  - RESOURCE_SEMAPHORE_QUERY_COMPILE wait type on executing tasks
- **Causes**
  - Incorrect configuration of Resource Governor
  - Ad hoc workloads causing high compilations
- **Resolution**
  - Review Resource Governor configuration and impact to memory gateway thresholds
    - Each Resource Governor resource pool allows separate large compilations to occur but reduces the gateway threshold
    - Could cause medium gateway queries to become large and compile serially
  - Rewrite queries to reduce compilation memory requirements
  - Improve plan cache reuse through parameterization, stored procedure usage
  - Reduce compilations by using plan guides for common ad hoc queries

External Memory Pressure
Windows OS Pressure

- **Symptoms**
  - SQL Server memory usage constantly changing
  - Memory\Available Mbytes < 150-300MB
  - RING_BUFFER_RESOURCE_MONITOR entries in sys.dm_os_ring_buffers of type RESOURCE_MEMPHYSICAL_LOW
- **Causes**
  - Concurrent application loads on the server with incorrectly configured 'max server memory'
  - Large system cache issues or VM ballooning (future slides)
- **Resolution**
  - Configure 'max server memory' to reserve memory for the OS and other applications

External Memory Pressure
Windows System Cache

- Windows caches buffered I/O operations in the System Cache (e.g. file-copy operations)
- Working set trim problems on 64-bit systems
  - 2GB VAS limit constrains system cache usage on 32-bit
- Dynamic cache service can be used to limit the size of the System Cache
- http://support.microsoft.com/kb/976618

External Memory Pressure
Working Set Trims

- **Symptoms**
  - Slow response times, high CPU, and high disk activity on the page file drive
  - SQL Server 2005 SP2+ error log entry "A significant part of SQL Server process memory has been paged out"
- **Causes**
  - Incorrect settings for the max server memory sp_configure option, when Lock Pages in Memory is not being used
  - Large system cache in Windows caused by caching of non-buffered I/O operations such as file copy operations
  - Hardware-driver issues that result in memory leaks or excessive memory allocations by the driver

External Memory Pressure
Working Set Trims

- **Resolution**
  - Configure 'max server memory' appropriately to leave memory available for the OS as well as any other applications running on the server
  - Track down the faulty drivers
  - Install the Dynamic Caching Service to minimize the system cache size
  - Remove file shares from the SQL Server and create a separate file server if appropriate
  - Change file copy operations to use a tool that doesn't buffer the I/O
    - eseutil /y <srcfile> /d <destfile> (http://tinyurl.com/eseutil)
    - XCOPY /J (Windows Server 2008 R2)
  - Use Lock Pages in Memory in for SQL Server instance
- **Reference:**
  - http://support.microsoft.com/kb/918483

External Memory Pressure
VMware Ballooning

- **Symptoms**
  - Slow response times with high disk I/O on the data drives
  - Windows OS Available Memory < 150MB
  - SQL Server memory usage less than server configuration
  - VMware Memory\Memory Ballooned > 0
- **Causes**
  - Incorrect configuration of the SQL VM in ESX hypervisor for memory overcommit usage in VMware
  - Hypervisor memory pressure resulting in VM memory ballooning to maintain hypervisor stability
- **Resolution**
  - Set min server memory values for SQL Server process inside the VM
  - Set static memory reservations for the SQL Server VM in the hypervisor
- **Reference: http://bit.ly/VMwareSQLBalloon**

Virtual Address Space Issues

- **Symptoms (especially on 32-bit environment)**
  - Error 701 FAILED_VIRTUAL_RESERVE
  - CLR App Domain Unload due to memory pressure
  - Failed backups
- **Causes**
  - Limited Virtual Address Space inside of a 32-bit process
  - Fragmentation of memory regions due to allocation and de-allocation over time (all VAS allocations must be contiguous)
  - Excessive memory requirements (ex. backups with 4MB MAXTRANSFERSIZE and 16 BUFFERCOUNT would require a 64MB contiguous region of VAS)
- **Resolution**
  - Migrate to 64-bit installation
  - Remove/reduce VAS consumption
    - Lower BUFFERCOUNT and MAXTRANSFERSIZE for backups
    - Tune CLR code to efficiently use memory
  - Increase startup reservation with -g<size> startup parameter

DBCC Commands

- **FREEPROCCACHE ({ plan_handle | sql_handle | pool_name })**
  - Clears the procedure cache of all plans or a single plan if specified
  - Often misused as a brute force fix for parameter sniffing issues (this is not the correct way to deal with bad plans)
- **FREESYSTEMCACHE ({'ALL' | cache_name } [, pool_name])**
  - Clears all entries for the specified cache from memory, or optionally for a single pool
  - Use WITH MARK_IN_USE_FOR_REMOVAL to asynchronously free the currently used cache entries as soon as they are free
- **FREESESSIONCACHE**
  - Flushes the distributed query connection cache used by distributed queries against an instance of Microsoft SQL Server.
- **DROPCLEANBUFFERS**
  - Clears the buffer pool of non-dirty data pages forcing physical reads from disk (useful for testing performance but not for production)

Key Takeaways

- **Use Lock Pages in Memory to prevent paging of the buffer pool**
- **Configure 'max server memory' to reserve memory for the OS**
  - 1GB for the OS + 1GB for each 4GB between 4-16GB + 1GB for each 8GB above 16GB in the server
- **Monitor memory usage with performance counters, DMVs and DBCC MEMORYSTATUS output**
- **Large Page support should not be enabled for large amounts of RAM, but will slow down instance startup time and makes the buffer pool non-dynamic**
- **Run DBCC CHECKDB in a separate resource governor workload group to limit execution memory grants flushing the buffer pool (fixed in 2014)**

Review

- SQLOS Memory Manager
  - SQL Server 2008 R2 and earlier
  - SQL Server 2012 and higher
- SQLOS memory components
- Configuration options for SQL Server
- How to identify memory issues
- Internal memory pressure
- External memory pressure
- Virtual address space issues
- Resource semaphores
- DBCC commands

Questions?

Appendix: 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

Appendix: 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.

Appendix: 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

Appendix: SQLOS Memory Manager

SQL Server 2005 – 2008 R2

- **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

Appendix: Memory Manager

SQL Server 2005 – 2008 R2