

SQLskills Immersion Event

IE2: Performance Tuning

Module 8: CPU Performance Tuning

Joe Sack

Joe@SQLskills.com

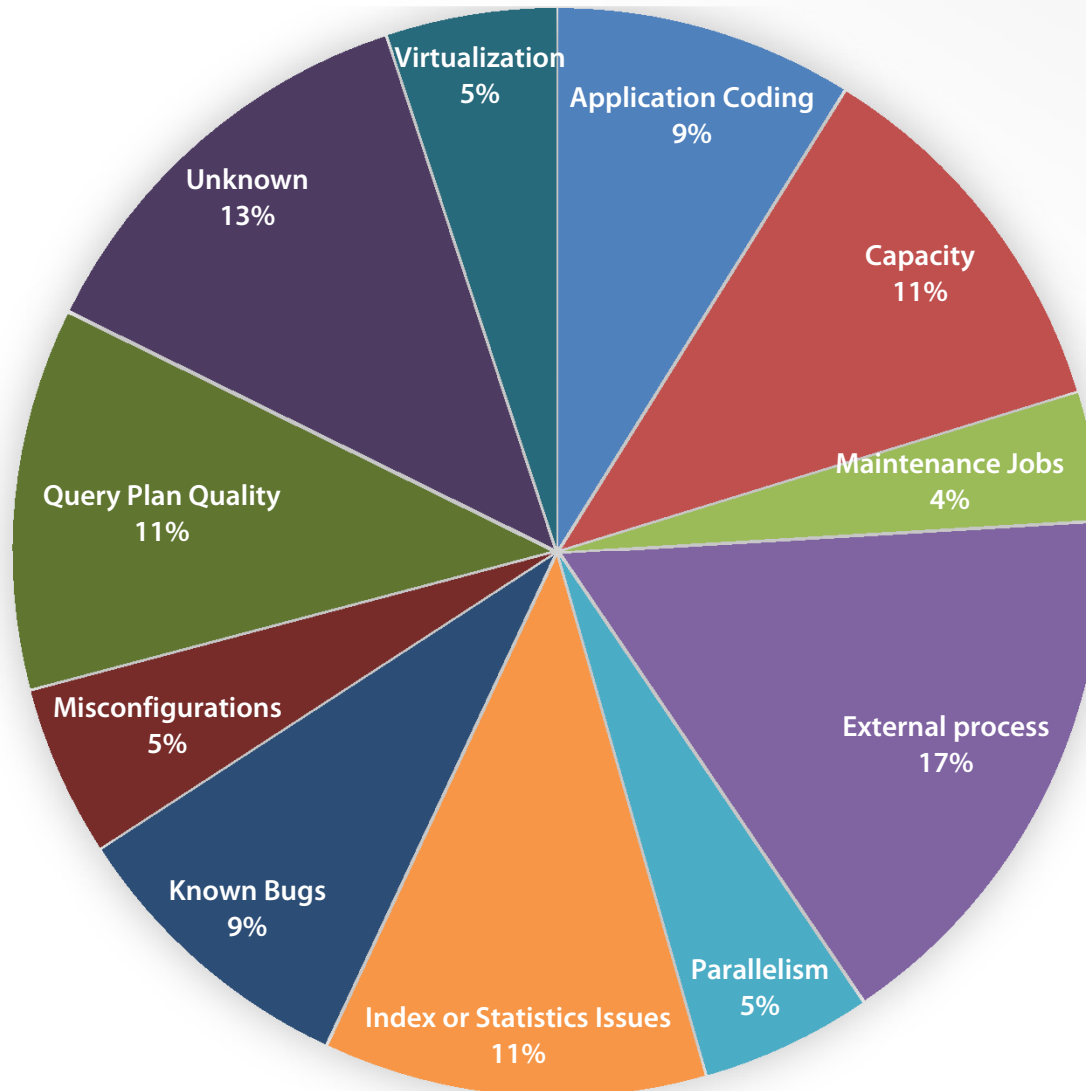


Overview

- Defining CPU performance issues
- SQL Server vs. external CPU issues
- Kernel vs. User time
- Where to find the SQL Server CPU consumers
- What to look for and how to remediate
- Controlling CPU with Resource Governor

CPU Issue “Themes”

Informal categorization of 79 “high CPU” issues from serverfault.com, stackoverflow.com, recent client, MSDN forums of issues as of January 21st, 2013...



Defining CPU Performance Issues

- **Define the problem space:**
 - Availability impacted
 - For example “pegged CPUs” bring all else to a halt
 - Low CPU and no requests completing
 - Degradation of overall performance
 - Workloads run, but slower than desired (SLAs)
 - Capacity and scalability limitations
 - Workloads run fine, but throughput has ceiling
 - Workload competition
 - Some queries run fast, some slow: winners/losers
 - One request takes all the resources: the rest suffer
 - Capacity constrained
 - Power options?
 - Over-provisioned virtualization host?

Baseline Information (1)

- **Physical server**
 - ❑ Sockets, cores, hyper-threading
 - ❑ Processor model, architecture (32-bit/64-bit)
 - ❑ NUMA, L2/L3 cache sizes
- **Virtual server**
 - ❑ Host info (same as physical SQL Server) and guest VM virtual cores
 - ❑ Hyper-V CPU Reserve, VMware CPU Reservation
 - ❑ Hyper-V CPU Relative Weight, VMware CPU Shares
 - ❑ vCPUs and co-scheduling to other guests
- **SQL Server instance configuration settings:**
 - ❑ Max degree of parallelism setting
 - ❑ Cost threshold for parallelism option
 - ❑ Processor affinity setting
 - ❑ Priority boost setting
 - ❑ Max worker threads setting
 - ❑ Lightweight pooling setting

Baseline Information (2)

- **CPU power-option settings:**
 - Windows
 - High Performance, Balanced, Power Saving
 - ESX
 - See VMWare KB article 1018206
- **Resource Governor configuration (more on this later)**
- **SQL Error Log**
 - We'll discuss CPU related DMVs, but it helps to take a quick look for unusual errors and warnings prior to going deeper
 - You may see downstream errors as well
- **Windows Event Logs**
 - Look for hardware related errors (drivers, failures, degradation)

Demo

CPU-Z, powercfg.cpl and sys.configurations

But is SQL Server the Culprit?

- **Don't jump to conclusions until you've confirmed that the issue is indeed SQL Server**
 - ❑ Process: % Privileged Time (kernel mode)
 - ❑ Process: % User Time (user mode)
 - ❑ Task Manager
 - ❑ Or if you're already connected to SQL, you can rule out SQL itself via `sys.dm_os_ring_buffers` (more on this later)

Kernel or User Time?

- **Validated through Process: high SQL % Privileged Time (sqlservr object)**
 - Multiple instances?
 - Map SELECT SERVERPROPERTY ('processid') to PID
- **sys.dm_os_ring_buffers**
 - $100 - (\text{SystemIdle} + \text{ProcessUtilization})$

SQLSERVER Privileged Time Causes

- What could cause high kernel (privileged) time for SQL?
 - Filter-drivers inject themselves into the Windows driver stack (Anti-virus, encryption services)
 - E.g. large I/O operations causing high CPU (privileged time)
 - Windows Server 2008 R2 Hfix KB 976700
 - Missing firmware updates or drivers
 - Defective or significantly insufficient I/O components

Demo

Measuring % User vs. Kernel time

What About Wait Statistics?

- **Wait statistics can point to CPU pressure**
 - High ratio of signal wait time to resource wait time
 - SOS_SCHEDULER_YIELD – when used in conjunction with other diagnostic data
 - CXPACKET – again, when used in conjunction with other diagnostic data
- **Otherwise, wait statistics point to contention for other resources (synchronization, I/O, network, memory, locking)**

What about Virtualization?

- Is the guest on an over-committed vCPUs on virtual machine host?
- Even if not over-committed – are there co-scheduling issues causing a Ready state?
- What is CPU Ready Time (VMware) telling you?
 - Covered further in IE3, but see “CPU Ready Time in VMware and How to Interpret its Real Meaning” (<http://bit.ly/UTe7Wr>) for further information

If it's SQL User Time, What Next?

- We next have to first identify which queries are driving the CPU consumption
- Common patterns:
 - One “bad” serial query executed by multiple requests
 - Easier to prioritize (no triage required)
 - One “bad” parallel query using all schedulers
 - Again, this is easier to identify
 - Numerous requests consuming schedulers (“death by 1,000 cuts”)
 - Look at cumulative worker time (we’ll discuss)

CPU's Correlation to I/O

- **You may see high CPU queries responsible for minor amounts of I/O**
 - Calculations, loops, conversions
- **You'll also see high CPU with high I/O**
- **Resolving an I/O issue often solves your CPU issue as well**
 - Adding more memory or improving the I/O path may not help you though, because logical I/O still drives CPU
 - A reason why Resource Governor is limited in this scenario

Demo

The CPU and I/O relationship

Resist the Urge to Do the Following...

- **Not sure of the root cause yet? Don't fall back on the following "solutions" unless you want to risk masking the root cause:**
 - Update all statistics
 - Rebuild all indexes
 - Restart SQL Server
- **Example: you see high CPU so you blindly update all statistics and that "solves the problem" (for now)**
 - Why is this a problem?
 - The statistics caused the problem procedure to recompile, removing the actual issue (parameter sniffing), and the recompiled plan is "good" but will revert to "bad" at a future time (problem not solved)

The Observer Effect

- When possible, demote SQL Profiler and Trace in favor of other methods in order to avoid adding your own overhead
- Factors which increase overhead:
 - Types of events tracked (e.g. Showplan XML Statistics Profile)
 - Volume of requests
 - Use of SQL Profiler versus server-side traces
 - Distance of SQL Profiler to the server
 - Number of concurrent traces

Live Troubleshooting Advice

- **Its okay to use pre-canned queries and procedures, but be sure you understand what exactly they're doing**
 - Procedures that create intermediate result sets or are calculation-intensive may not complete execution on a heavily constrained system
 - Understand the core DMVs and don't be afraid to run them in isolation and in smaller pairings (execute-collect-execute)
- **Don't forget about the Dedicated Admin Connection (DAC)**

Answering Questions with DMVs (1)

- **Which request is running right now?**
 - `sys.dm_exec_requests`
- **What is it executing?**
 - `sys.dm_exec_sql_text`
- **Where is it coming from?**
 - `sys.dm_exec_sessions`
 - `sys.dm_exec_connections`
- **What's its plan? (careful on this one)**
 - `sys.dm_exec_query_plan` or
 - `sys.dm_exec_text_query_plan` (for very large plans)
- **Who's waiting on the scheduler and why?**
 - `sys.dm_os_waiting_tasks`

Answering Questions with DMVs (2)

- Which queries have taken up the most CPU time since the last restart?
- **sys.dm_exec_query_stats**
 - Aggregate by total_worker_time
 - Define averages with execution_count
 - If ad hoc workloads, group by query_hash or query_plan_hash
 - Use the plan_handle with sys.dm_exec_query_plan to grab the plan

Answering Questions with DMVs (3)

- Is this query using parallelism?
- **sys.dm_os_tasks**
 - Ordered by session_id, request_id
- **sys.dm_exec_query_plan**
 - Look at plan operators
- **sys.dm_exec_query_stats**
 - Filter total_elapsed_time less than total_worker_time
 - Can be a false negative for blocking scenarios – where duration is inflated due to a wait on resource

Capturing Compilation Stats

- **Excessive compilation activity can also have a negative impact on performance**
 - Check SQL Compilations/sec, SQL Re-Compilations/sec
 - This is where baselines come in handy – helping you define what is “normal”
 - Compare to Batch Requests/sec over time
 - Check sys.dm_exec_cached_plans and see if there is a high number of single use plans
- **Validate sys.dm_exec_query_stats and query_hash to validate queries that are “the same” but creating new plans due to a lack of parameterization**
 - Form an argument for parameterization and preparing queries
- **Profiler SP:Recompile/SQL:StmtRecompile EventSubClass can be used to validate reasons behind recompiles (schema/stats/SET options/temp table changes/hints)**

Analyzing the Data (1)

- High I/O operations (logical I/O driving CPU)
 - Missing indexes, scans, lookups, sorts, spools
- Hash joins and aggregates (merge too)
- Warnings (stale statistics, Cartesian products)
- Cardinality estimate issues
- Parameter sniffing
- Unexpected or skewed parallelism
- Cursor fetches (including API cursors), doing 1,000s of singleton ops

Analyzing the Data (2)

- **Implicit data type conversion**
 - Example: Microsoft JDBC Driver (sendStringParametersAsUnicode=false)
- **Excessive compilation / recompilation**
- **Inappropriate hinting, plan guides**
- **Calculation intensive UDF calls**
- **Scalar UDFs called RBAR on large result sets**
- **Concurrent maintenance activities (e.g. UPDATE stats with FULLSCAN)**

Demo

High-CPU scenarios

Resource Governor

- A user-defined classifier function runs at login and evaluates properties (name, workstation, database) and assigns incoming connections to a pre-defined workload group
- A workload group is bound to a resource pool
- A resource pool can be associated with ≥ 1 workload groups
- Monitor resource usage by resource pool and workload group
- Dynamically alter any of the above
- You can use it to:
 - Implement quotas, with a chargeback system
 - SQL Server hosting partners
 - Learn about aggregated resource usage on a system
 - Limit concurrent users, parallelism
 - Avoid runaway queries
 - Guarantee resources for critical workloads

Concepts: Classification

- Incoming connections are classified into workload groups
- Classification function is a T-SQL user-defined function
 - Takes no parameters and returns group name as a sysname type
- Various functions to use for classification:
 - HOST_NAME, APP_NAME, SUSER_NAME, IS_MEMBER, new CONNECTIONPROPERTY
- Easy to think of scenarios where connections classified by time of day too
 - e.g. batch processing or maintenance
- Registered with the resource governor using:
 - ALTER RESOURCE GOVERNOR WITH (
 - CLASSIFIER_FUNCTION = <name>)
- NOTE: Once a connection is assigned to a group it cannot be changed (i.e. you can't further throttle an active query)
- NOTE: Dedicated Admin Connection bypasses all of this
- Beware of classifier functions (or logon triggers) that take so long to execute that the query timeout limit is reached

Concepts: Workload Groups (1)

- **A workload group allows grouping connections into a 'bucket'**
 - E.g. batch processes, reports, executives, helpdesk, maintenance
- **Syntax: CREATE / ALTER / DROP WORKLOAD GROUP**
- **Assigned set of resource limits by mapping to a resource pool**
- **Internal tasks (e.g. checkpoint, ghost cleanup) always are part of the internal group**
 - No limits on CPU or memory and will pressurize all other groups, regardless of their limits
 - Cannot be changed
 - Login processing and trace event generation are done in the internal group temporarily
- **Unassigned connections go into the default group**
 - No limits on CPU or memory but will be pressurized by all other groups and settings can be changed – for example, memory grants for columnstore indexes

Concepts: Workload Groups (2)

▪ Options (defaults in bold):

- ❑ IMPORTANCE = {LOW | **MEDIUM** | HIGH}
 - ❑ Not the same as thread priority, for instance
 - ❑ Applies to position in RUNNABLE queue on a single scheduler for workers from groups using the same pool
 - ❑ LOW:MEDIUM:HIG = 1:3:9 ratio
- ❑ REQUEST_MAX_MEMORY_GRANT_PERCENT = value (**25%**)
 - ❑ Max memory grant from the resource pool for workspace memory (sorts, hash join operations)
- ❑ REQUEST_MEMORY_GRANT_TIMEOUT_SEC = value (**0**)
 - ❑ How long to wait for a query execution memory grant
 - ❑ Timeout != failed query – may get execute at reduced performance
- ❑ REQUEST_MAX_CPU_TIME_SEC = value (**0**)
 - ❑ Max amount of CPU time a request can use before event is fired
- ❑ MAX_DOP = value (**0**)
 - ❑ Max degree of parallelism. Overrides the sp_configure option
- ❑ GROUP_MAX_REQUESTS = value (**0**)
 - ❑ Maximum concurrent requests per group

Concepts: Resource Pools (1)

- A resource pool is a way to limit resource consumption for one of more workload groups
 - (2008) 18 pools, (2012) 62 user-definable resource pools
 - Syntax: ALTER / CREATE / DROP RESOURCE POOL
- Options (with defaults in bold):
 - MIN_CPU_PERCENT = value (**0**)
 - MAX_CPU_PERCENT = value (**100**)
 - Only enforced when multiple workers use a single scheduler
 - This can be confusing at first... an opportunistic maximum...
 - (2012) CAP_CPU_PERCENT = value (**100**)
 - Hard cap limit on CPU resource usage
 - Beware: Scripting out configurations using SSMS in SQL Server 2012 RTM will omit new RG features
 - Addressed in SQL Server 2012 SP1

Concepts: Resource Pools (2)

■ Options (continued):

- (2012) AFFINITY = AUTO
 - Attach pool to specific schedulers or NUMA nodes
 - For example... WITH (AFFINITY SCHEDULER = (0))
 - If using this setting, be sure to reconfigure after CPU changes – for example, vCPUs allocated on a virtual machine
 - Troubleshooting via sys.dm_resource_governor_resource_pool_affinity
- MIN_MEMORY_PERCENT = value (**0**)
 - Non-buffer pool allocations (plan cache for example)
- MAX_MEMORY_PERCENT = value (**100**)
 - Non-buffer pool allocations (plan cache for example)

MAXDOP Settings

- **Server-wide setting applies as a limit...**
- **Except that MAXDOP query hint will override it**
 - And anyone can specify a MAXDOP hint that overrides server-wide MAXDOP
- **Except if the query is running under resource governor in a workload group that specifies a non-zero MAX_DOP**
 - Workload group MAX_DOP cannot be exceeded

Demo

Resource Governor and parallelism

Limitations

- **Works with the Database Engine only**
- **Single instance only**
 - Each instance controlled individually
- **Controls for CPU usage and memory allocation ONLY**
 - CPU usage includes CLR, but not pre-emptive operations
 - CPU governing occurs per scheduler
 - Certain workloads may not be entirely suited, e.g. short-lived OLTP queries
- **No way to tell if a specific query was throttled in any way**
- **No I/O governing (not in 2012 either)**
- **(2008) Multiple page allocations not governed**
 - (2012) Multiple page allocations are now governed along with single page allocations
 - Memory governance doesn't include the buffer pool

Review

- Defining CPU performance issues
- SQL Server vs. external CPU issues
- Kernel vs. User time
- Where to find the CPU consumers
- What to look for and how to remediate
- Controlling CPU with Resource Governor (overview and application)

Resources

- **Whitepaper: Troubleshooting Performance Problems in SQL Server 2008**
 - <http://download.microsoft.com/download/D/B/D/DBDE7972-1EB9-470A-BA18-58849DB3EB3B/TShootPerfProbs2008.docx>
- **Whitepaper: Using the Resource Governor**
 - <http://msdn.microsoft.com/en-us/library/ee151608.aspx>
- **Whitepaper: Resource Governor in SQL Server 2012**
 - <http://msdn.microsoft.com/en-us/library/jj573256>

Questions?

