Performance Tuning - How to Isolate and Resolve System Bottlenecks

Ying Fan
Support Engineer
Customer Service & Support
Asia Pacific & Greater China Region

Welcome!

Microsoft

CSS Academy

Last Workshop Re-cap

- Blocking Issue
  - Locking
  - PageIOlatch
  - Pagelatch
  - ASYNC NETWORK IO
  - Tempdb Contention
- Deadlock

Agenda

- **Introduce SQL Server performance tuning concepts and tools**
- Analyze CPU performance bottleneck
- Analyze Memory performance bottleneck
- Analyze Disk performance bottleneck

Introduce SQL Server performance tuning concepts and tools

- Perform tuning overall introduction
- Tools:
  - Performance monitor(PerfMon)
  - Profiler
  - ReadTrace
- Others: Blocking script, Pssdiag, SQL Nexus, DMVs, Performance Dashboard.

Performance tuning introduction

- What do we mean by a performance issue?
- Performance Definitions
  - Baseline
  - Cost
  - Optimization

What impacts SQL Server performance

- Application
- Query
- Transaction Management
- Database Design
- Data Distribution
- SQL Server
- Network
- Operating System
- Hardware

PerfMon

- A tool to monitor system performance
  - CPU
  - Memory
  - Disk IO

Important performance counters

- CPU:
  - Process
  - Processor
  - SQLServer: SQL Statistics
- Memory:
  - Memory
  - SQLServer: Buffer Manager
  - SQLServer: Memory Manager
- Disk
  - PhysicalDisk

Sample Interval of PerMon

- Run PerfMon long enough.
- Guideline:
  - < 2 Hrs: every 5 secs
  - 2 Hrs~1 D: every 30 secs
  - 1~5 Ds: every 3 mins
- Interval > = 5 Seconds to avoid impacting performance
  - Exception: Disk counters

Profiler

- A graphical tool used to Monitor SQL Server events (mainly for CPU and compilation)
  - Capture all SQL Server activities
  - As workload for Database Tuning Advisor
  - Audit

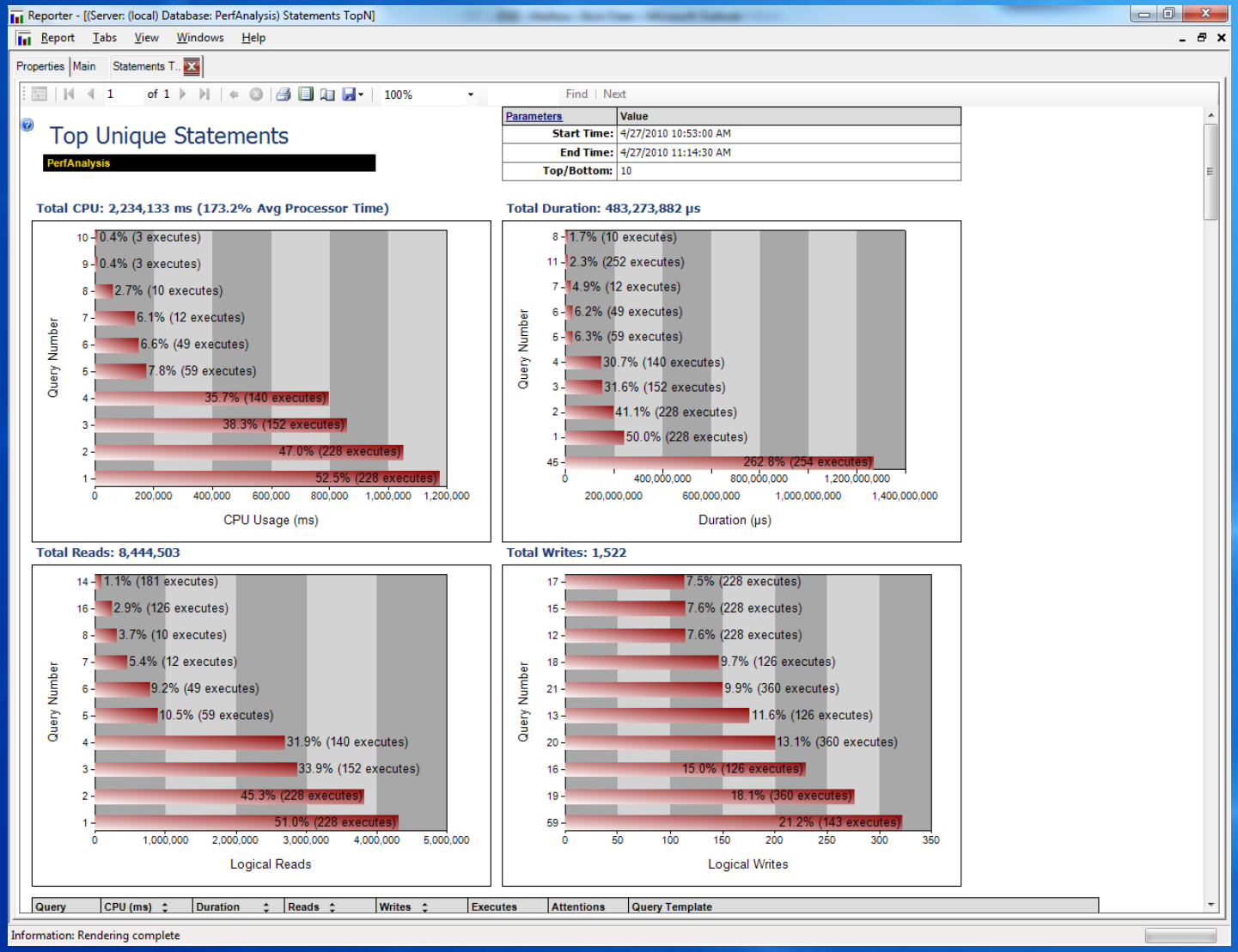
Important events

- Statements:
  - RPC: Completed, SP: StmtCompleted, SQL: BatchCompleted, SQL: StmtCompleted
- Errors and warnings
- Execution plan/(re)compilation:
  - Performace: Showplan Statistics Profile, SP: Recompile, SP: CacheMiss, SP: CacheRemove

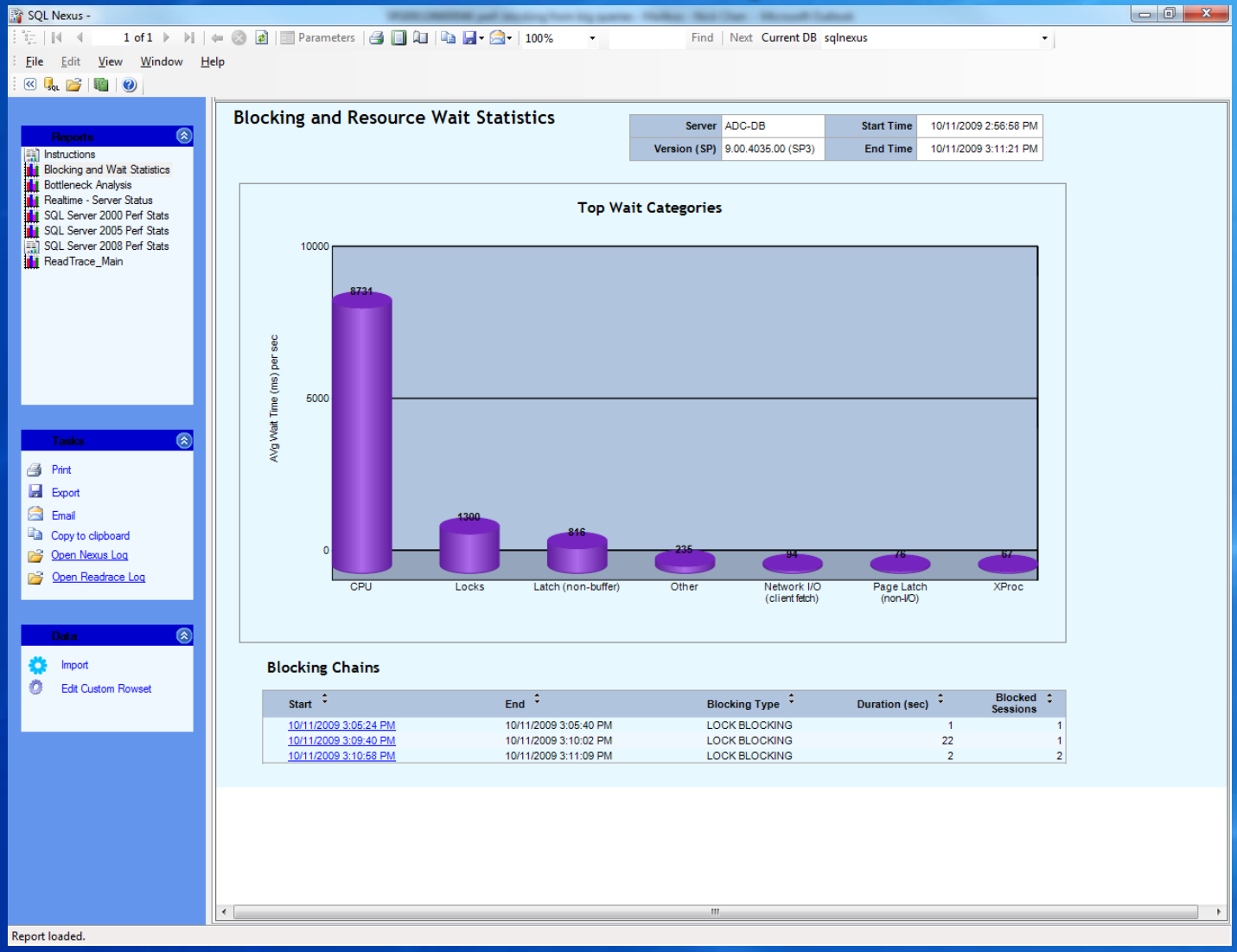
Read Trace

- Install RML utility: http://support.microsoft.com/?scid=kb%3Ben-us%3B944837&x=3&y=12
- ReadTrace –I”trace file” [-f] [-d”dbname”]
- What to check
  - Expensive queries

ReadTrace



Demo 1.1 - SQL Nexus



Blocking script

- KB 271509

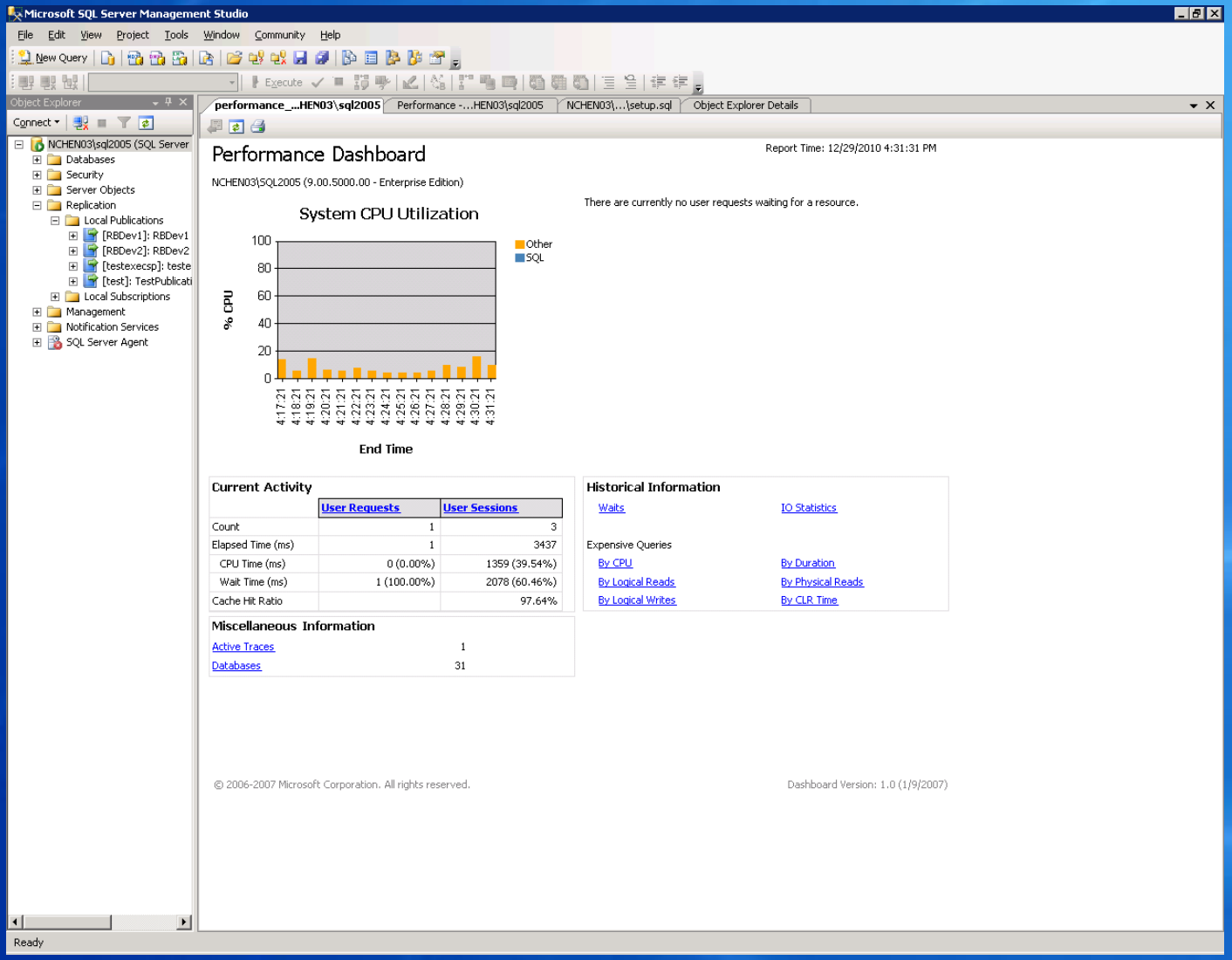
  http://support.microsoft.com/kb/271509/

- Querying the follow info periodically:
  - Sysprocesses
  - Syslockinfo
  - DBCC sqlperf(waitstats)
  - DBCC inputbuffer
  - DBCC opentran

DMVs

- Dynamic Management Views and functions.
- Examples
  - sys.dm_exec_query_stats
  - sys.dm_exec_cached_plans
  - sys.dm_exec_connections
  - sys.dm_exec_sessions
  - sys.dm_exec_requests
  - sys.dm_tran_locks

Demo 1. 2 - Performance Dashboard



Lab 1: Use PerfMon and Profiler

Agenda

- Introduce SQL Server performance tuning concepts and tools
- **Analyze CPU performance bottleneck**
- Analyze Memory performance bottleneck
- Analyze Disk performance bottleneck

Analyze CPU performance bottleneck

- CPU performance bottleneck symptoms
- Common causes and solutions

Symptoms

- Investigate when:
  - Processor:% Processor Time for each CPU consistently above 80%
- Check if:
  - Process:% Processor Time for sqlservr is high
- What's next?
  - Find the CPU consumers and tune them.
  - Add more CPUs.

Lab 2 - High CPU Issue

1. Click Start > Run > Perfmon
2. Click Performance Monitor, and then click View Log Data on the Performance toolbar. Click the Source tab, click the Log files option, and then click Add.
3. In Select Log file, locate and select lab2.1.csv, and then click Open.
4. Click Add on the Performance toolbar to select counters.
5. From the CPU related counters, do you think there are any CPU performance bottlenecks? Why?
6. Double click Lab2.1.trc to open it in profiler
7. Find out the most CPU intensive query.

Common causes

- Excessive compilations/recompilations
- Inefficient query execution plan
- Intra-query parallelism
- Poor cursor usage

Excessive compilations/recompilations

- Compilations and recompilations
  - Recompilations: before SQL Server executes a query, it checks for the validity and correctness of the existing query plan. If one of these checks fails, compilations/recompilations happen.
  - CPU intensive

Detection

- PerfMon
  - High SQL Recompilations/sec to Batch Requests/sec ratio
- Trace
  - SP: Recompile
  - RPC: Completed

Demo 2

Cause & Solutions

- SET statement
  - Put the SET at the connection level
- Statistics change
  - Specify KEEPFIXED PLAN query hint
- Ambiguous object names
  - Use qualified object names
- Hint usage
  - Remove WITH RECOMPILE

Inefficient query plan

- Background
  - SQL Server optimizer attempts to find a plan with fastest response time
  - The plan is not always fastest, due to several factors (missing indexes, out of date statistics).
  - Certain elements of such sub-optimal execution plan (table scan/index scan, etc) could drive CPU high
- Detection
  - Trace, DMVs, T-SQL Statements

Cause & Solutions

- Obsolete statistics
  - UPDATE STATISTICS
- Missing indexes
  - USE Database engine Tuning Advisor (DTA)
- Code quality
  - Modify queries.
- Wrong indexes
  - Query hints, plan guide

Intra query parallelism

- Background
  - Query optimizer picks the fastest plan.
  - If the query's cost > cost threshold for parallelism, query executes in parallel
  - In most scenarios, parallelism should enhance query performance.
  - However, the response time for a given query must be weighed against the overall throughput and responsiveness of the rest of the queries on the system.

Detection

- sys.sysprocesses (Blocking script)
  - Multi entries for a single SPID
  - Wait type: CXPACKET

Solutions

- MAXDOP
  - Server level and statement level
- Analyze missing indexes
  - Threshold not reached
- Modify the query

Poor cursor usage

- Background
  - Earlier versions allowed only 1 command per connection.
  - In scenarios where another command needs to be run based on the row just read, customers resorted to server side cursors.
  - sp_cursorfetch controls the number of rows to be returned by the server to the client. This allows ODBC or OLEDB driver to cache the rows.

Detection

- PerfMon
  - SQL Server: Cursor Manager by Type: Cursor Requests/Sec
- Trace
  - RPC: Completed, search for sp_cursorfetch

Solutions

- Avoid server side cursors
- Use MARS
- Use a bigger fetch buffer size

Common scenarios to avoid in OLTP

Rule	Description	Value	Source	Problem Description
1	**Signal Waits**	**>25 %**	**Sys.dm_os_wait_stats**	**Time in runnable queue is pure CPU wait.**
2	**Plan reuse**	**<90 %**	**Perfmon object SQL Server Statistics**	**OLTP identical transactions should ideally have >90% plan reuse.**
3	**Parallelism: Cxpacket waits**	**>5%**	**Sys.dm_os_wait_stats**	**Parallelism reduces OLTP throughput. CXPACKET indicates that multiple CPUs are working in parallel, dividing up the query in smaller pieces. Ordinarily a well tuned OLTP application would not parallelize unless an index is missing, there is an incomplete WHERE clause, or the query is *not* a true OLTP transaction.**

Common Scenarios to avoid in Data Warehousing

Rule	Description	Value	Source	Problem Description
1	**Signal Waits**	**> 25%**	**Sys.dm_os_wait_stats**	**Time in runnable queue is pure CPU wait.**
2	**Avoid plan reuse**	**> 25%**	**Perfmon object SQL Server Statistics**	**Data warehouse has fewer transactions than OLTP, each with significantly bigger IO. Therefore, having the correct plan is more important than reusing a plan. Unlike OLTP, data warehouse queries are *not* identical.**
3	**Parallelism: Cxpacket waits**	**<10%**	**Sys.dm_os_wait_stats**	**Parallelism is desirable in data warehouse or reporting workloads.**

Agenda

- Introduce SQL Server performance tuning concepts and tools
- Analyze CPU performance bottleneck
- **Analyze Memory performance bottleneck**
- Analyze Disk performance bottleneck

Analyze Memory performance bottleneck

- How SQL Server uses memory
- How to identify memory performance bottleneck

How SQL Server uses memory

- VAS and physical memory
- Dynamic Memory Management
- AWE and Locked Pages

Virtual vs. physical memory

- Windows provides virtual memory services
- *Committed* virtual memory is backed by some type of physical storage – usually the page file or physical memory
- Windows handles translations between virtual memory and physical storage
- 32-bit processes have a 4GB virtual memory address space divided between kernel- and user-accessible areas

Dynamic Memory Management

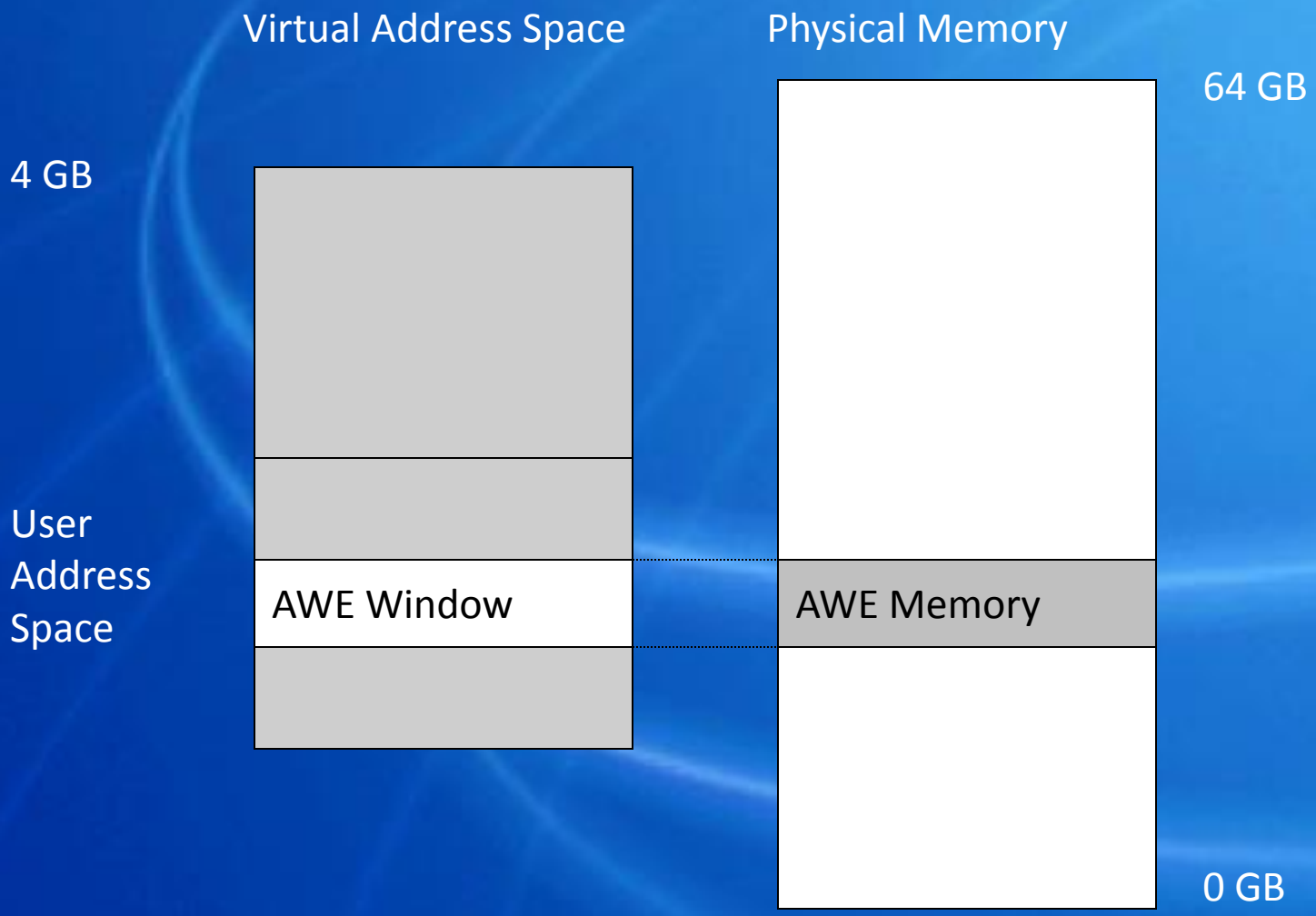
- When SQL Server is using memory dynamically, it queries the system periodically to determine the amount of free physical memory.

- Target/total server memory

Memory Configuration Options

- Min server memory (MB)
  - Provides floor
  - Does not automatically commit on startup
  - May not be able to reach floor
- Max server memory (MB)
  - Provides ceiling
  - SQL Server may not be able to back off due to current usage
- AWE enabled/ Lock Pages In Memory
  - To avoid Working Set Trimming (KBA 918483)
  - Supported by Enterprise Edition, or Standard Edition with specific patch, KBA 970070)

AWE Memory Layout



The Effect of Memory On Others

- Memory drives overall performance
- Effect on disk I/O
  - Excessive hard paging can interfere with data I/O operations on page file device
  - Decrease the effectiveness of system file cache
- Effect on CPU
  - Drives up kernel time
  - If memory is stressed (< 4 MB) paging can drive a large percentage of CPU time

SQL Server Memory Consumers

- Data Cache
- Consumer (memory clerks)
  - Connection
  - General: Resource structures, parse headers, lock manager objects
  - Utilities – Recovery, Log Manager
  - Optimizer – Query Optimization
  - Query Plan – Query Plan Storage
- DBCC MEMORYSTATUS (KB 907877)

Buffer Pool Size

- SQL Server calculates max buffer pool size and reserves it at startup
- The computation is based on physical memory
- Lowering **max server memory** does not change the amount of memory *reserved*
- To change the reservation size, increase MemToLeave (and shrink the maximum BPool size) via the -g parameter

Lazywriter

- Two purposes:
  - Tries to keep a minimum number of free buffers (freeing dirty buffers requires I/O)
  - Keeps enough physical memory free on the machine to avoid paging
- Many things that it can't remove

Memory pressure types

- External vs. Internal
- Physical vs. Virtual

Detection

- PerfMon
  - Key counters: Memory:Available Mbytes, SQL Server:Memory Manager, SQL Server:Buffer Mananger
- Symptoms
  - External
  - Internal

Common scenarios to avoid in OLTP

Rule	Description	Value	Source	Problem Description
1	Page life expectancy	<300 sec	Perfmon object SQL Server Buffer Manager	Page life expectancy is the average number of seconds a data page stays in cache. Low values could indicate a cache flush that is caused by a big read. Pure OLTP workloads do NOT issue big reads, thus possible missing index.
2	Page life expectancy	Drops by 50%	Perfmon object SQL Server Buffer Manager	Same as above
3	Memory Grants Pending	>1	Perfmon object SQL Server Memory Manager	Current number of processes waiting for a workspace memory grant.
4	SQL cache hit ratio	<90%	Perfmon object SQL Server Buffer Manager	SQL cache hit ratio falls under 90% for sustained periods of time greater than 60 sec. It is likely that large scans have to be performed, which in turn flushes out the buffer cache.

Common Scenarios to avoid in Data Warehousing

Rule	Description	Value	Source	Problem Description
1	Memory grants pending	>1	Perfmon object SQL Server Memory Manager	Memory grant not available for query to run. Check for Available memory and page life expectancy.
2	Page life expectancy	Drops by 50%	Perfmon object SQL Server Buffer Manager	Page life expectancy is the average number of seconds a data page stays in cache. Low values could indicate a cache flush that is caused by a big read. Look for possible missing index.

Demo 3.1 – Investigate Memory Bottlenecks

Troubleshooting steps

- PerfMon
- External pressure
- Internal pressure
- SQL ERRORLOG
- Sp_configure
- DBCC MEMORYSTATUS

Memory Myths

- **Myths**
  - System has large available memory = No memory bottleneck
  - SQL Server will always release memory under pressure
  - SQL Server can lock pages in physical memory
  - Increasing MemToLeave will help performance
  - “Min server memory” is committed on startup
  - Increasing memory will always help performance

Demo 3.2 – Resource Bottleneck Scenario

Problem:

- The end user complains application “slow”.

Info Gathered:

- Perfmon log
- Blocking script output

Questions:

- What do you think the problem is?
- Action plan?

Lab 3 – Investigate Memory Bottlenecks

Problem Symptom:
SQL 2005 was very slow between 6/3 23:48 – 6/4 2:50
Error message found in error log is as following:
“ **A significant part of sql server process memory has been paged out. This may result in a performance degradation.”**

Info Gathered:
Perfmon log

Questions:
What’s the normal SQL memory usage?
What’s the SQL memory usage at the problem time?
Action Plan?

Agenda

- Introduce SQL Server performance tuning concepts and tools
- Analyze CPU performance bottleneck
- Analyze Memory performance bottleneck
- **Analyze Disk performance bottleneck**

Analyze Disk performance bottleneck

- **Introduction**
  - SQL Server's performance depends upon the performance of your disk subsystem.
  - Not all the data pages of all your databases can fit into memory at all times, thereby causing I/O to disk.

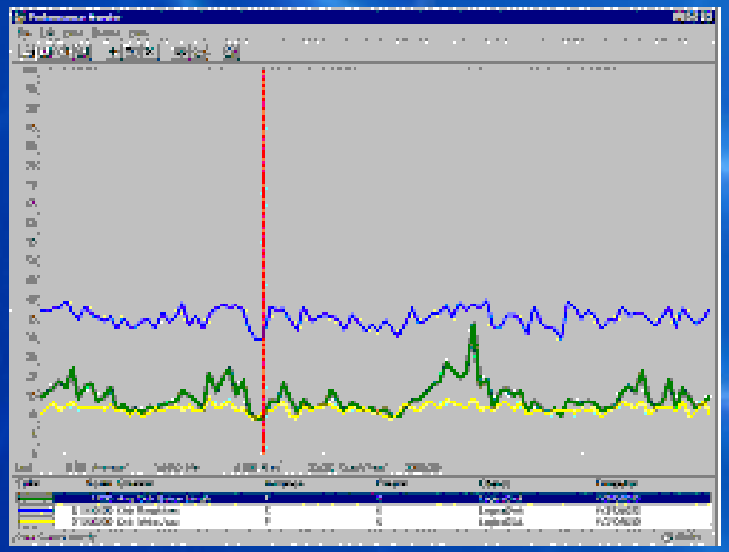
Physical vs. Logical Disks

- Physical Disks
  - Used to evaluate disk performance
  - Multiple disks may appear as one physical disk (RAID)
  - Disk topology may be required to isolate RAID performance issues
  - PhysicalDisk Instance list shows what logical disks are associated with each physical disk

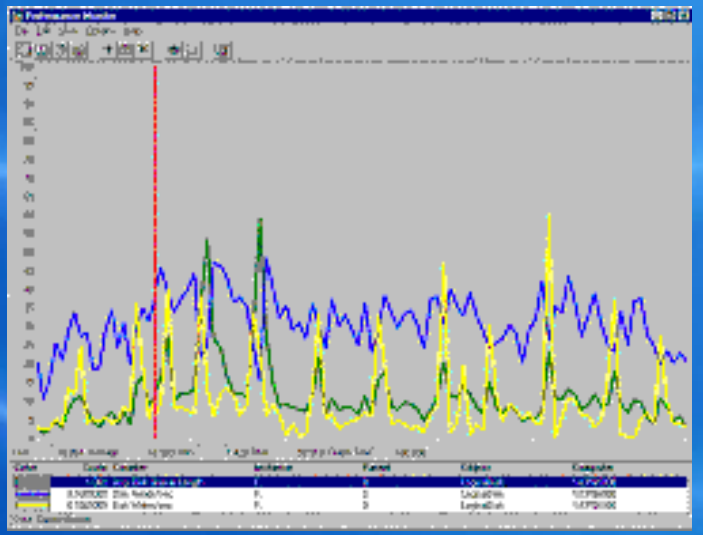
Monitoring I/O

- Use a separate log
- Use a small interval: 1-4 seconds
- Do not log to disks being evaluated

20 Seconds



2 Seconds

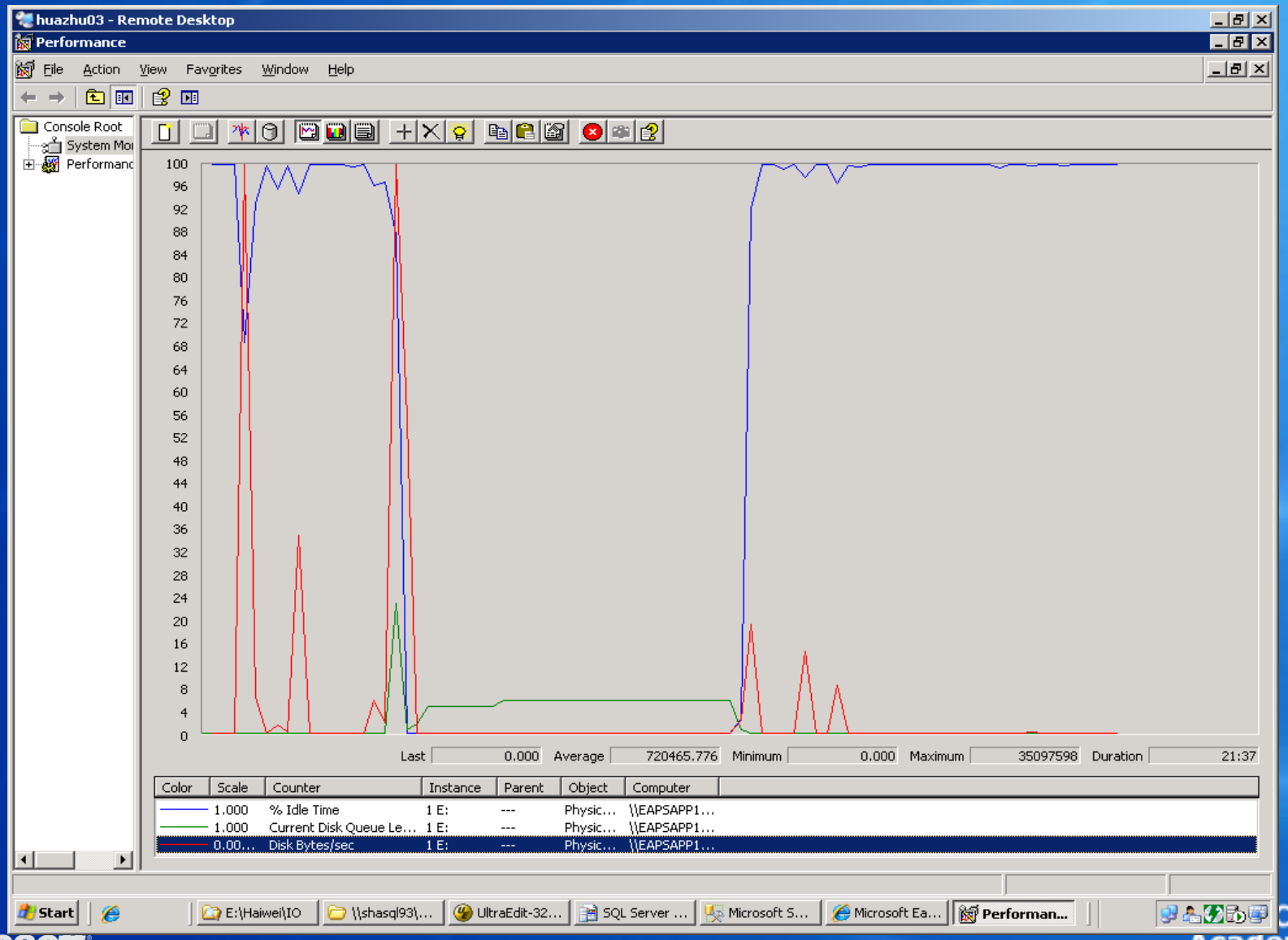


Detection

- PerfMon

Counter	Values
PhysicalDisk Object: Avg. Disk Queue Length	Consistently > 2
Physical Disk: %Disk Time	> 50%
Avg. Disk Sec/Read	Less than 10 ms - very good Between 10 - 20 ms - okay Between 20 - 50 ms - slow, needs attention Greater than 50 ms – Serious I/O bottleneck
Avg. Disk Sec/Write	
Avg. Disk Read Bytes/Sec	> 85% of disk capacity
Avg. Disk Write Bytes/Sec	

Case Study



Demo 4 – Check for IO bottlenecks

Items influencing Disk IO

- Poor execution plan (lack of efficient index)
- Too many OLAP style queries
- Hardware capacity not sufficient for IO Demands

Common scenarios to avoid in OLTP

Rule	Description	Value	Source	Problem Description
1	Average Disk sec/read	>20 ms	Perfmon object Physical Disk	Reads should take 4-8 ms without any IO pressure.
2	Average Disk sec/write	>20 ms	Perfmon object Physical Disk	Writes (sequential) can be as fast as 1 ms for transaction log.
3	Big IOs Table Scans Range Scans	>1	Perfmon object SQL Server Access Methods	A missing index flushes the cache.
4	If Top 2 values for wait stats are any of the following: ASYNCH_IO_COMPLETION IO_COMPLETION LOGMGR WRITELOG PAGEIOLATCH_x	Top 2	Sys.dm_os_wait_stats	If top 2 wait_stats values include IO, there is an IO bottleneck. Special care needs to be taken on log file

Common Scenarios to avoid in Data Warehousing

Rule	Description	Value	Source	Problem Description
1	**Average Disk sec/read**	**>20 ms**	**Perfmon object Physical Disk**	**Reads should take 4-8ms without any IO pressure.**
2	**Average Disk sec/write**	**>20 ms**	**Perfmon object Physical Disk**	**Writes (sequential) can be as fast as 1 ms for transaction log.**
3	**Big scans**	**>1**	**Perfmon object SQL Server Access Methods**	**A missing index flushes the cache.**
4	**If Top 2 values for wait stats are any of the following: ASYNCH_IO_COMPLETION IO_COMPLETION LOGMGR WRITELOG PAGEIOLATCH_x**	**Top 2**	**Sys.dm_os_wait_stats**	**If top 2 wait_stats values include IO, there is an IO bottleneck**

Lab 4 - Check for IO bottlenecks

Info gathered:

- Perfmon log (Lab4.csv)
- SQL ERRORLOG and sysdatabases output (Lab4_sqldiag.TXT)

Question:

- Does the SQL server have any IO bottlenecks?

Solutions

- Query Plans
- Data Compression
- Upgrading the IO subsystem

Practices

Info gathered

- Perfmon: *Practice.BLG*
- SQLdiag output: *Practice(sqldiag).OUT*
- Blocking script: *Practice2(blocking).OUT*

Tasks

- Find out the resource bottlenecks on the SQL server based on the available information
- Find out the next Action Plans

Reference

- Troubleshooting Performance Problems in SQL Server 2008

http://msdn.microsoft.com/en-us/library/dd672789(SQL.100).aspx

Q & A

Thank You!

Microsoft®

Your potential. Our passion.™