Windows System Performance Measurement and Analysis

Jeffry A. Schwartz
Integrated Services, Inc.
jeffrys@isi85.com
Personal Introduction

- Specialized in Performance, Modeling, & Capacity Planning for over 30 years
- Concentrated **solely** on Windows for 12 years
  - Worked on many mainframe class Windows machines for customers like Microsoft IT
  - Added SQL Server specialization in mid-2001
    - Now primary concentration
- Have conducted approximately 775 Windows performance studies
- Developed and have taught this course since 2006
- Developed 310,000 line Windows performance analysis & reporting suite
- Provide series of **free**, recorded, one-hour Windows and SQL Server performance webcasts
  - @ [http://www.sqlrx.com/technicalpapers.aspx](http://www.sqlrx.com/technicalpapers.aspx)
  - @ MSSQLTips.com
  - @ [http://www.msdev.com](http://www.msdev.com), the Microsoft Developers Training Site
Class Overview

• Windows Architecture
  – Approximately 35% of the class
  – **Absolute** minimum required to understand metrics

• Performance Data
  – Approximately 65% of the class
  – Monitoring
    • Tools and collection mechanisms
  – Available metrics and their interpretation
  – Supplementing standard performance data with derived metrics
What’s New from Previous Courses

• **Windows Architecture**
  – Streamlined even further from previous classes to make room for more practical examples

• More performance data examples

• Considerations for monitoring in virtual and shared disk environments

• Additional practical Event Tracing for Windows examples & using Windows Performance Analyzer
  – Determining what files are causing a particular disk LUN to be too busy

• Obtain more information regarding what is consuming processor resources
Class Timeline – 3 Sessions – 2 Parts

• Windows Architecture
  – CPU
  – Memory
  – Disk cache and I/O
  – Network (very minimal treatment)

• Performance Data
  – Basics and Counters
  – Monitoring
    • Tools and collection mechanisms
    • Analysis tools
  – Available metrics, their interpretation and usage
  – Supplementing standard performance data
Class Emphasis

• Fundamental Performance-Related Windows OS Concepts
  – Processor interrupts & scheduling
  – Memory management
  – I/O handling

• Collection, interpretation, and usage of informative performance counters and other performance-related information
  – Add calculated measurements
  – Expand upon PerfMon explanations
  – Understand when additional metrics and tools required
Class Emphasis

• Useful graphical depictions
• Insights acquired from many customer analyses
• Possible courses of action
• Tools to answer in-depth questions illuminated by standard performance counters
• Have as interactive a course as possible
  – Audience encouraged to ask questions throughout the course
Topics Not Covered

• Windows architecture that does not apply directly to performance analysis
  – Resources for those desiring in-depth Windows architecture knowledge
    • Windows Internals 5
    • Windows 2000 Performance Guide
    • Windows 2003 Performance Guide (only available in Windows 2003 Resource Kit)

• Windows
  – Administration
  – Development topics

• General Performance methodologies
• Capacity planning methodologies
Audience Questionnaire

• How many
  – Still using Windows 2000?
  – Using Windows 2008?
  – Have access to Windows 7?
  – Using x86 Windows and supporting machines with > 4 GB memory?
  – Using x64 Windows?
Why Study Processor Internals?

- Must understand significance of processor counters

- Understand how various counters indicate problems and what problems they point to
  - High kernel processor usage
  - What DPCs and interrupts are
  - High interrupt processor usage
  - High DPC processor usage

- Understand why processor queue values are erratic

- Understand why context switching is necessary
  - What happens when it is really high
Processor

• Processor accounting
• Processor scheduling
• Processor ramifications
  – Hyperthreading
  – Multi-core (2, 4, 6, and 8)
• Interrupt handling
Processor Busy

• Each processor goes to its own Idler process while waiting for work

• Processor busy actually time not spent in idler processes
Kernel Mode Versus User Mode

• Kernel mode – Processor execution mode that grants access to all system memory and all CPU instructions
  – Operating system code runs in kernel mode
    • System services
    • Device drivers
  – Examples include memory allocation and deallocation

• User mode – Execution mode that prevents one user from infringing upon either system or another user’s memory structures unless explicitly shared

• Critical that analyst understands difference because kernel activities supersede others
Process Processor Time

• Windows’ processor times inexact measurements
• “Sampled”

• Sum of process’ processor times can exceed 100% on multiple processor systems
  – Multiple threads can execute simultaneously
    • SQL Server and Oracle common examples

• Comparing total processor consumption with total process consumption
  – May be close or quite different
  – More later
Processor Scheduling

- Windows schedules processors at **thread** level
- Windows has a single “queue” of threads waiting to use processor(s)
  - Ready queue
- Only one ready queue for all processors on multiprocessor systems
- Windows Processor Scheduling appears to be classic M/M/c queuing system
  - Queuing by priority, not FIFO
- Scheduling “tweaks” improve throughput and prevent “perpetual overtaking”
Windows Processor Scheduling

• Uniprocessor systems use preemptive-resume priority scheduling

• Multiprocessor systems “almost” use preemptive-resume priority scheduling

• Ready queue actually 32 separate queues, one for each of Windows kernel's 32 priority levels
  – FIFO within priority level
Multiprocessor Scheduling

• Scheduling on multiprocessor systems more complex

• Multiprocessor Scheduling does not insure highest priority threads are running
Thread States

- Ready – Waiting to execute
- Standby – Selected to run on a specific processor
- Running – Active
- Waiting – Wait for I/O, sleep, waiting for an event
- Transition – Ready, but kernel stack is out of memory
- Terminated – Finished and able to be torn down
- Initialized – Internally used while thread being created
Quantum

• Thread not allowed to run on a processor forever

• Quantum
  – Length of time thread can use a processor
    • Time-slice quanta = 3 * time between regularly scheduled timer
      interrupts (processor-dependent)

• Threads start with default quantum values of
  – Default quantum value for Windows Server = 36

• Thread’s quantum value reduced each time processor
  clock interrupt occurs

• Thread’s quantum expires when quantum value
  becomes 0 → thread forced back into ready queue
Scheduling “Tweaks”

• Reasons
  – Avoids certain types of problems
  – Increases system throughput
  – Insures better responsiveness

• Two types
  – Priority boosts
  – Quantum stretching

• Priority boosts occur
  – After I/O completion
  – GUI threads entering wait state
  – CPU starvation (perpetual overtaking)
Hyperthreading

- **Not** completely independent processors
  - Two or more pipelines into **one** core
- **Some** applications, mainly OLTP, experience at most a 30% improvement
  - Not recommended for SQL Server
  - Be **very careful** when using this
- Two, four, or even eight processors reported for each processor chip
- Windows 2003 **internally** distinguishes physical and logical processors
  - Earlier versions make **no** distinction
- Vista/2008/Windows 7 expose physical & logical procs
Dual-Core, Hex-Core, Quad-Core, 8-Core

- Effectively separate processors on same chip
- Must distinguish between **physical** processor **chips** and **cores**
  - Chips sometimes referred to as sockets
- Much more common on newer machines than hyperthreading (HT)
  - But, HT is making a comeback
- **Beware of this**: “8-core, 16-threaded”
  - This **really** means 8 cores, and HT provides 16 threads, 2 per core
Processor Interrupts

• Tell processor to stop what it is doing either now or later (immediate or deferred)

• Can be generated by hardware or software

• Two basic types
  – Immediate – Have interrupt service routine handle it NOW!
  – Deferred – Can wait a while before handling
    • Deferred Procedure Call (DPC)

• Queued by type

• PerfMon interrupt category depends on Windows version
  – Specifics discussed shortly
Why Should You Care About Immediate or Deferred Interrupts?

• 8, 16, and 24-processor systems have been throttled by DPC handling that regularly approached or reached 100%
  – Once this occurs, nothing can increase system throughput

• High utilization indicates
  – Bad driver
  – Improperly configured NICs, e.g., checksum calculations not offloaded to NIC
  – Teamed NICs can cause serious DPC problems because they use the TCP/IP stack for inter-card communication
  – Improperly configured processor interrupt assignments
  – DBMS connection affinity not used
Interrupt Handling

Source: Windows Internals Version 5, Solomon & Russinovich, pg. 573
Interrupt Processor Assignments

• Assigned at boot time prior to Server 2008
  – Frequently assigned to the same processor, but may move especially after hardware or software changes

• Significant because interrupts often single-threaded through one processor
  – Applies to both immediate interrupts and deferred procedure calls (DPC)
  – Usually a good thing because processor cache reuse is excellent
  – Problematic only when sufficient number of interrupts saturate a single processor
  – Unfortunately, some hardware vendors regularly violate this methodology (more later)
Interrupt Processor Assignments

• Usually one of the highest numbered processors
  – Often highest numbered processor
  – May be the highest numbered processor in a module

• Exceptions
  – Multiple NICs
    • Often assigned to different processors, especially in modular machines such as the Unisys ES7000
  – Mask or HAL can be used to spread interrupts
    • HP machines (Proliant) often spread interrupt handling across multiple processors using the HAL (Hardware Abstraction Layer)

• Assigned per connection on Server 2008 & Vista SP1
  – Each connection remains assigned to a processor
Why Study Memory Internals?

• Must understand page faults

• Essential to distinguish page faults that are
  – Ones that only involve memory initialization or movement (cheap)
  – Involve disk (costly)

• Differentiate between
  – Data being read in
  – Data being moved around in memory
  – Memory space being initialized

• Otherwise memory counters impossible to understand
  – Essential to understand when Windows short on memory
Memory

• Capacities

• 32-bit and 64-bit layouts

• Memory management architecture
  – Mechanisms
  – Page faulting
  – Cache

• Small and large page sizes vary depending upon processor type (x86, x64, and IA64)
Types of Memory

• Nonpaged
  – Must be resident in physical memory

• Paged
  – Can be removed from memory
  – Includes both code and data
  – Data must be written to backing store before removal from memory
Memory Management – 32-bit

• Supports 4 GB flat address space (32-bit addresses)

• Divided into system (2 GB) and user (2 GB) areas (Enterprise Edition allows 1 GB system and 3 GB user)

• Uses two-level page tables to map from virtual memory to real memory

• Memory limits for all OS’ and bit types (32, 64) in presentation Appendix (slide 200 – 207)
Memory Management – 64-bit

• Supports 7,152 GB flat address space

• Divided into 5 regions
  – Process Address Space
  – System Page Table
  – System Cache
  – Paged Pool
  – Nonpaged Pool

• Uses two-level page tables to map from virtual memory to real memory
Memory Address Mapping

- Page directory index
- Page table index
- Byte index

CR3 Register → PDE → Page tables → PTE → Page

Page directory
Page tables
Physical memory
Memory Management Component Definitions & Abbreviations

- Page Directory – Structure whose entries point to page table entries
- Page Table – Entries point to physical memory pages
- PDE – Page Directory Entry
- PTE – Page Table Entry
- CR3 Register – Special register on x86 systems inside CPU
  - Loaded by operating system
  - Contains physical address of page directory
### PTE Field Descriptors

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page frame number</td>
<td>12-31</td>
<td></td>
</tr>
<tr>
<td>Global</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Dirty</strong></td>
<td>6</td>
<td>Modified</td>
</tr>
<tr>
<td>Accessed</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Cache disabled</strong></td>
<td>4</td>
<td>Used by databases</td>
</tr>
<tr>
<td><strong>Write through</strong></td>
<td>3</td>
<td>Goes directly to disk, i.e., not handled by Lazy Writer</td>
</tr>
<tr>
<td>Owner</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Valid</strong></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Physical Address Extension (PAE) Mode

- Allows 32-bit Windows to use more than 4 GB
- Memory address has 4 parts
- PDEs and PTEs are 64 bits
- Bits 30-31 of address are page directory index
- Three-level page table, but conceptually similar to two-level 32-bit mechanism shown earlier
Memory Allocation

• Two-part process
  – Reserve address space
  – Commit pages in address space

• Reserving space allows thread to reserve range of virtual addresses

• Committed pages, when accessed, will translate to valid pages in physical memory
Shared Memory

- Windows allows processes to share memory
- Virtual addresses in two or more separate processes mapped to same physical pages
- Used to allow sharing of code files
- Implemented as section or file mapping objects
Re-entrant Code

• When code re-entrant, can be shared by all processes running the code file

• Program code re-entrant **only** if code never modified

• Example:
  – Five Excel processes could run sharing one copy of the Excel.exe file
  – All processes can share a single copy of a DLL’s code
Page Fetch Policy

• Windows implements fetch on-demand

• Page fault generated when page table entry accessed with Valid bit off
  – Causes Windows to fetch page into memory or allocate memory
  – Reads in multiple pages when handling a fault
Lifetime of Process Memory

- Initial process address space constructed
  - Page directory
  - Intermediate working area for various memory transitions
  - Working set list

- Following entities mapped into address space
  - Page tables for non-paged system memory and system cache

- Main process thread started

- As program executes, page faults generated for those parts of its address space not in memory
  - Demand Zero page fault occurs first time data accessed
  - First time code accessed, it may or may not cause a page fault (could already be in memory)

- Process working set increases as page faults occur
Page Frame Database

• Describes each page in real memory

• Each page in real memory can be in one of 9 states
  – 8 on Windows 2000
Most Important Page Frame States

- **Active** – Part of working set, pointed to by valid PTE
- **Transition** – I/O in progress
- **Standby** – Pointed to by valid PTE, but marked invalid; unmodified page removed from working set
- **Modified** – Pointed to by valid PTE, but marked invalid; modified page removed from working set
- Modified no write – Same as modified, but marked so will not be written to disk (used by NTFS)
- **Free** – Free page, but not zeroed
- **Zeroed** – Free page initialized with zeroes
Page Lists

- Standby
- Modified
- Modified no write
- Free
- Zeroed
- Bad
- Read-only memory
Page List Dynamics

Source: Windows Internals Version 5, Solomon & Russinovich, pg. 808
Process Working Sets

- Processes start with default minimum and maximum working set sizes
  - 50 MB min, 345 MB max

- If sufficient free memory exists, memory manager will allow process to grow its working set up to its limit when page fault occurs

- After process reaches its working set limit, page faults remove pages from process’ working set to satisfy page faults (local policy)

- **Exception**: working set allowed to grow above its maximum – **provided** enough free pages exist
System Working Set

- **Single** working set for system
- Five different kinds of pages in system working set
  - System cache pages
  - Paged pool
  - Pageable code in NTOSKRNL.EXE
  - Pageable code in device drivers
  - File system mapped views
Working Set Trimming

- Performed by memory manager when available physical memory runs low
- Trim cycle runs every 6 seconds
- Examines each process in memory
- Removes pages for processes above their working set minimum
- If available memory still too low, continues to remove pages until enough free memory exists
Which Pages Removed?

• **2003/2000 Single-processor/XP**
  – Steps through working set list (list of all pages in process working set)
  – Examines accessed bit in hardware PTE
    • If bit is 0
      – Page aged (count incremented indicating page not referenced since last scan)
      – Age used to locate candidate pages for removal
    • If bit is 1, clears bit and goes to next entry
  – Each scan starts where last scan ended

• **Multiprocessor Windows 2000**
  – Access bit not used because clearing would require invalidating TLB (translation lookaside buffer) entries on other processors
  – Scans working set list and removes each page in turn without regard to access bit
Available Page Mechanisms

- In-Use pages can be made available
- Available pages kept on one of four lists
  - Standby page list
  - Modified page list
  - Free page list
  - Zero page list
Page List Dynamics

- When page removed from process working set
  - Page moved to *standby* list, if unmodified
  - Page moved to *modified* list, if “dirty”

- Background processes move pages among lists
Modified Page Writer

- Writes pages in modified page list to *page file* using two threads
- Moves page to standby list when write completes
- Runs when
  - Number of pages on modified page list above system threshold (800 pages)
  - Number of system free pages falls below minimum value
- Waits 300 seconds (5 minutes) for separate event that indicates *mapped pages* (not modified pages) should also be written to *disk* using separate thread
  - Reduces data loss in case of crash
  - Used by programs like those of *Microsoft Office*
Zero Page Thread

- Runs when free list has 8 or more pages
- Runs at lower priority than all other processes on system
- Write zeroes to page and moves it from free list to zero list
Address Windowing Extensions (AWE)

  - AWE almost never used on 64-bit Windows (unnecessary)

- Setting SQL Server to use almost all memory has often caused problems when AWE used
  - Effectively causes most of memory to become non-paged

- Can cause system to
  - Exhaust Windows memory
  - Page heavily
  - Cease to function entirely

- **Only** recourse on 32-bit OS with memory > 4 GB
AWE Mapping

Figure 7-6  Using AWE to map physical memory

Source: Windows Internals Version 5, Solomon & Russinovich, pg. 720
Address Windowing Extensions

- AWE-related tables hard, if not impossible, to identify
  - Experts have stated that AWE-related tables consume more Windows memory as more AWE memory locations accessed
  - Experience has shown this to be true
When Windows Runs Low on Memory

• Believe it or not, this still happens in 2011!

• Page faults increase
  – Soft faults first, then hard faults

• Increases reads from and writes to page file

• Reads from image files can increase

• Utilization of some disks or the network (NAS) will increase, usually nonlinearly
  – Consistent with Denning’s Working Set Theory

• System will eventually stop responding
Memory Page Fault PerfMon Counters

- Memory: Demand Zero Faults/Sec
  - Soft fault

- Memory: Transition Faults/Sec
  - Soft fault
  - Recovered pages used by another process
  - Retrieved from modified or standby lists
  - Being written to disk at time of request

- Memory: Cache Faults/Sec
  - May result in either soft or hard fault

- Memory: Page Faults/Sec
  - Encompasses both soft and hard faults
Windows’ Disk Cache

• Provide data caching for all Windows file systems (both local and network) unless explicitly bypassed

• Uses memory manager to control portions of file retained in physical memory

• Caches on “virtual blocks,” not hardware addresses, allowing intelligent read-ahead

• Used primarily by non-database applications that do not perform their own memory management, i.e., programs that use flat files
  – Most programs that use files fall into this category
Virtual Cache Memory Size

- Calculated based on physical memory size
- Default is 64 MB
- Adjusted dynamically as necessary
- **Virtual** Cache size set to 128 MB plus 64 MB for each additional 1,024 pages (4 MB)
- 64 MB physical memory example calculation
  - $128 \text{ MB} + (64 \text{ MB} - 16 \text{ MB})/4 \text{ MB} \times 64 \text{ MB} = 896 \text{ MB}$
Cache Physical Size

- Handled by normal memory management
- Cache is part of system working set
- Physical size changes through working set expansion and trimming

- PerfMon Counters
  - Memory: System Cache Resident Bytes
  - Memory: Cache Bytes
  - Memory: Cache Bytes Peak
  - Memory: Cache Faults/Sec
Disk Cache Mechanism

- Cache maps regions of file into system virtual address space
- As cache virtual addresses accessed, memory manager reads in blocks not already in memory
- Memory manager pages data out of cache
- Pages for cached files may not be in physical memory because of normal working set trimming
Virtual Block Caching

- Most systems cache based on physical disk address (NetWare, OpenVMS, Unix, Linux)
- Windows caches based on file-relative offset
- Maps files into slots of system virtual space called views
- Allows for
  - Intelligent read-ahead
  - Fast I/O bypassing generation of IRP (I/O request packet)
Cache Operation

• Cache handles “opened” files

• Can open files without caching (discussed earlier on *PTE Field Descriptors* slide)
  – Used by databases (SQL Server, Oracle)

• Write
  – Copy data to view in virtual memory
  – Written out using “lazy write”
  – Calls memory manager to flush page to disk
Dirty Page Threshold

• Wake up thread to flush dirty pages when dirty pages > “dirty page threshold” (800 pages)

• Threshold computed at initialization based on size of physical memory and registry key:
  \System\Control\SessionManager\Memory Management\LargeSystemCache

• Default key value is 0

• Modified by properties of network server
Lazy Writer Thread

• When dirty page threshold exceeded, lazy write thread activated
  – Threshold = 37.5% * # physical pages (3/8 * # physical pages)

• Once **EVERY** second, thread enqueues 12.5% (1/8th) of dirty pages to be flushed

• I/O actually done by **different** worker thread
  – Makes workload I/O characterization **virtually impossible**
  – Easily observed when system idle after many files updated – flurry of disk activity for no apparent reason

• **PerfMon Counters:**
  – Cache: Lazy Write Flushes/sec
  – Cache: Lazy Write Pages/sec
Write Through

- Write through
  - Can open file with FILE_FLAG_WRITE_THROUGH
  - Page will be flushed **immediately** when updated
  - Often used by database software
  - Disk controller may do something behind the scenes that Windows **knows nothing about** depending upon its cache settings

- PerfMon Counters
  - Cache: Data Flushes/sec
  - Cache: Data Flush Pages/sec
Read-ahead

- Two types:
  - Virtual address read-ahead
  - Asynchronous read-ahead with history

- Virtual address read-ahead improves performance for sequential I/O

- Asynchronous read-ahead with history attempts to determine pattern in random I/O
Why Study I/O Subsystem Internals?

• Most prevalent problem in today’s server environments
• Must understand where I/O measurements come from
• Understand meanings of captured metrics
  – Incredible amount of confusion regarding which measurements to watch and what they mean
I/O Subsystem

• I/O architecture

• Disk measurements and collection
  – Collector locations
  – Available measurements
Disk Performance Data Collection

- On Windows 2000 default is Physical Disk, but not Logical Disk
  - Must install driver to get data
  - Must run diskperf utility to activate disk performance data (diskperf -y, diskperf -ye)
- Must be member of Administrators group to run diskperf
  - Installs driver DISKPERF.SYS
  - Must perform this procedure if disk PerfMon counters missing
- Beginning with Windows 2003, both on by default
I/O Architecture

Source: Windows Internals Version 5, Solomon & Russinovich, pg. 539
Physical I/O Measurements

• I/O time measured directly by disk driver, which provides transfer times and other disk information to Windows

• I/O time = service time + queue time due to driver’s location in I/O path
  – Disk response time

• Important to know whether queuing causing large I/O times
  – May not be possible to improve large service times due to physical or financial constraints

• Hardware RAID obscures implementation details
Windows Storage Stack

Source: Windows Internals Version 5, Solomon & Russinovich, pg. 646
Windows’ Software RAID

- Supports RAID 0, 1, and 5
- Can see activity on each physical disk if diskperf installed properly
- Hardware solutions generally perform better
- Avoid RAID 5 (uses significant CPU)
- Some customers with Write Once-Read Many Times data use both hardware and software RAID
  – Perfect for research data
Windows Networking

- LAN segments shared resource
- Transmissions can be queued because NIC is busy
- Transmissions can be delayed because of collisions trying to use segment
- Networking delays not reliably measured
Windows Performance Analysis

- How Windows collects data
- How system monitors obtain performance data
- Sources of counter inaccuracy
- Performance counter problems
- Performance data collection issues
- PerfMon considerations
- Useful PerfMon counters
- Graphical examples
- Calculation of missing values
Sources of Windows Performance Data

- Registry API
  - Provides counters to PerfMon

- Windows Management Instrumentation (WMI)
  - Can provide counters to PerfMon

- Event logs

- Event Tracing for Windows Trace logs

- Special programs, e.g., kernrate
How System Monitors Obtain Data

- Calls RegConnectRegistry with computer name to obtain key to registry
  - Parameter is HKEY_PERFORMANCE_DATA
- Calls RegQueryValueEx using key to capture performance data
- Translates object and counter numbers into meaningful names for UI
- Gets counter values from returned data
- Users almost never see data in its raw form
Data Collection Registry Keys

HKLM
  \SYSTEM
  \CurrentControlSet
  \Services
  \ApplicationName
  \Performance

Library = <name>
Open = <name>
Collect = <name>
Close = <name>
First counter = <value>
First help = <value>
Last counter = <value>
Last help = <value>

Application’s performance key
Data Collection Registry Keys

Names and descriptions of objects and counters

- HKLM
  \SOFTWARE
  \Microsoft
  \Windows NT
  \CurrentVersion
  \PerfLib
    Last Counter = <value>
    Last Help = <value>
    \009
    Counters = <strings>
    Help = <strings>
Security

• Must ensure access to registry has been allowed
  – Remote Registry Service **must** be running on target machine when collecting from remote machine

• Simplest to run under administrator user id
  – Need not be domain user id
  – If using PerfMon, change Performance Logs and Alerts service to run under a specific id and **NOT** `LocalSystem`
  – Can also control this using the user ID in a log definition

• Or change access rights to:
  – `HKLM\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg`
Windows Performance Counters

• Three-level hierarchy

• Objects at top level
  – Multiple objects often required, e.g., SQL Server

• Counters
  – Comprise **bottom** level
  – Always pertain to particular object (explanations available)
  – Memory object → Page writes/sec counter

• Instance level added between object and counter levels when necessary
  – Processor object → Processor 0 instance → % Processor Time counter
Windows Performance Counters

• Three most common types of counters
  – **Difference or Accumulator**
    • Hardware interrupts, # of paging operations, % Processor Time
  – **Instantaneous**
    • Processor Queue Length
  – **Compound**
    • % Free Space, % Cache Hits

• Most often gathered using PerfMon

• Also collected by **many** other tools, e.g.,
  – NTSMF
  – SQL Sentry
  – Spotlight
  – BMC
  – Sightline
Windows Performance Counters

• Generally low overhead
  – Fears of excessive overhead unfounded unless Process and Thread information collected without in-memory summarization

• Obtainable from local or remote machine
  – Local collection can increase system overhead
  – Remote collection can increase network overhead
    • Agents often reduce this overhead

• Collection tool architecture determines
  – Whether collected locally via agent or remotely

• Intel benchmark centers use remote method over separate network segment
PerfMon Counter Extensibility

- Windows provides core set of counters
- Applications can add own counters
- Microsoft services (SQL Server, IIS, .Net, …) add their own counters
- Third party software can also add their own counters (Oracle, IBM)
Performance Counter Difficulties

• Many System Performance Monitor explanations useless

• Example explanations
  – SQL Compilations/sec → “Number of SQL compilations”
  – Table Lock Escalations/sec → “The number of times locks on a table were escalated”
  – Bulk Copy Rows/sec → “Number of rows bulk copied per second”

• Literally hundreds of unique counters
  – Potentially thousands of instance-counter combinations

• Prior to Server 2008, PerfLib counters periodically corrupted
Sources of Counter Inaccuracy – Very Important!!

• Counters always incremented internally

• **NOT** guarded by locks
  – Minimizes impact of performance data gathering on overall Windows performance

• Updated **while** sample being taken

• Time between samples **not same** for all counters, but values **calculated** as if they were

• Therefore
  – Values not always synchronized with each other
  – Reconciling multiple counters with each other may be difficult, if not impossible
  – **Plan on discrepancies**
Reload Performance Counters Example

- Identify currently loaded PerfMon DLLs
  - Use CTRLIST.EXE from Windows Resource Kit
    
    \texttt{CTRLIST > c:\temp\counters.txt}

- Unload SQL Server counters
  
  \texttt{Unlodctr mssqlserver}

- Reload counters
  
  \texttt{Lodctr <SQL Binn path>\sqlctr.ini}

- Restart of SQL Server service may be required

- If this fails, \textbf{all} counters must be rebuilt
  - Very involved and requires Windows reboot
System Performance Monitor, a.k.a. PerfMon

• **Reliability & Performance Monitor** on Server 2008, Vista, Windows 7

• Includes two main components
  – System (or **Performance**) monitor
  – Performance Logs and Alerts
    • Counter logs
    • Trace logs
    • Alerts
  – Components rearranged on Server 2008, Vista, Windows 7

• Uses Performance Logs and Alerts service when logging
  – Service is stopped when not collecting (starts when needed)
  – Appropriate codefiles MUST be used on x64 machines – mismatches result in **no data collection**
    • Also true when collecting remotely from an x86 machine
Performance Data Analysis

• PerfMon
  – Initially not intended as primary performance data analysis tool, but evolved into just that
  – Analysis typically VERY slow and laborious, especially with large files
  – No really effective way to export substantial amounts of raw data via GUI
  – Improved greatly on Windows 7 compared to prior releases
    • Still easier to move the data into something like Excel

• Windows Performance Tools (WPT) Kit (more later)
  – Useful, very detailed, takes time to learn, run sparingly
  – Requires Vista/Server 2008/Windows 7 for installation
  – However, data collection can occur on any Windows platform
Performance Data Analysis

- Data collection frequency
  - 30 seconds usually sufficient for performance problems
  - 1 to 5 minutes usually sufficient for trending/capacity planning
  - **DO NOT** use GUI default of one second for any monitoring
    - Places undue and unnecessary pressure on system
    - Displays misleading values due to volatility
    - Minimize GUI usage unless small # of counters and low collection frequency
    - Smallest frequency ever should be 5 seconds for very short periods

- GUI default settings can be changed and saved so proper behavior automatic
  - Update frequency
  - Selected objects, instances, and counters
PerfMon Analysis Considerations

• CSV format
  – Easier to use outside of PerfMon, if # of counters < 256 on XL 2003 or using XL 2007/2010
  – Records sometimes can be impossibly long for spreadsheet programs, although 2007/2010 seem to handle them well
  – Imposes 10x more overhead on collection machine (NOT target machine) than if binary used

• Binary format
  – Usually significantly larger than CSV format
  – Cumbersome because primarily usable only with PerfMon
  – Required if process data gathered
  – Convertible to other formats using Relog program shipped with 2003 or Windows 7
    • Converted files MUCH smaller, but NO data lost in conversion
PerfMon Analysis Considerations

• New instances will **NOT** be collected on Windows 2000 **OR** if **non-binary** method used
  – Only instances present when collection begins will be captured
  – Most problematic with **Processes** and **Threads**
    • Subsequent processes will be ignored
    • Threads **very seldom necessary**
  – Corrected beginning with Server 2003 **ONLY** when **binary** format used
    • Textual formats **prevent** new instances, i.e., processes, from being added, regardless of Windows version
      – Header written when file opened so the layout cannot be changed
Relog

- Converts NT 4 and Windows 2000/2003/2008 logs for easier manipulation outside PerfMon
  - Any format → CSV, tab-delimited, SQL Server tables
- Runs on Windows 2000, if XP/2003 pdh.dll located in same directory as executable
  - XP pdh.dll ≠ 2003 pdh.dll
- **Windows 7** version **UNBELIEVABLY** improved, especially when converting very large binary files (literally days to minutes)
  - Unfortunately, Windows 7 version cannot be ported to another machine
- If **Windows 7 machine unavailable**, use 2003 version instead of XP or Vista
  - 2003 version has fewest problems
- Can create counter list for binary logs from unfamiliar systems
  - Counter list can be specified to minimize/tailor resulting output file
  - Usually manual procedure
PerfMon Log Collection Topics

- Templates can be created to capture consistent metrics
  - On Windows XP/2000/2003, an **HTM** template can be created and imported into PerfMon
    - You need not use PerfMon to create an entire template
    - Can extend an existing (or exported template) by adding objects
      - Must be careful to maintain object numbering system
      - Update objects total at the end of the template
  - On Windows 2008/Vista/7, an **XML** template can be created and imported into PerfMon
    - XML Notepad 2007 (free tool) quite useful for editing an XML template
    - Use a tool that really understands XML tree structures to edit
    - Main issue is that the objects occur in two places — Counter and CounterDisplayName
Performance Counter Categories to Collect

- Processor
- Memory
- Physical Disk
- Logical Disk
- Network I/O
- Redirector (NAS or mapped drives)
- Server (file servers)
- Terminal Services (if used)
- Web Service and Active Server Pages (IIS servers)
- Application-specific objects
Performance Counter Locations

• Extremely important for analyst to become familiar with crucial counter locations
  – Processor Queue Depth and Context Switches/sec in the **System** object, **NOT** the Processor object
  – Single processor queue and context switches apply to all processes and processors

• **Must** monitor “system-related” and application, e.g., DBMS metrics

• Remember applications like SQL Server, Oracle, or Exchange run within Windows operating environment
  – If Windows not performing well, application won’t either, even if only application on server
Useful Processor Performance Counters

- Processor
  - % Processor Time
  - % User Time
  - % Interrupt Time
    - Mostly disk interrupt handling on Windows 2000
    - On 2003 and 2008, usually VERY small unless a driver problem
  - % DPC Time (Deferred Procedure Calls)
    - Mostly network interrupt handling on Windows 2000
    - Much larger on 2003 and 2008 because of the architecture changes
      - More interrupts deferred on 2003 and 2008
Useful Processor Performance Counters

• Processor
  – Each processor, logical or physical, is an instance
  – DPC & Interrupt often isolated to specific processors (single-threaded through interrupt processor) and ignored during analysis
  – Remainder is REAL program kernel mode time
    • Must compute actual Kernel time because % Privileged Time contains %Interrupt and %DPC
    • Kernel time = % Privileged Time - %Interrupt Time - %DPC Time

• System
  – Processor Queue Length
  – Context Switches/sec
“Published” Rules of Thumb

• Processor queue length > 2 * # of processors
• Context Switches < 10,000
  – 20,000 (8 or more processors)
Interpreting Performance Counters

• Processor queue length cannot be used reliably unless long spikes occur
  – Application design can affect these metrics greatly, e.g., Oracle 8
  – Also issues with timing of information collection
  – Most useful as a relative measurement

• Context switches counter valuable to a point
  – 8, 16, 24, 32, and 64 processor systems often generate at least 20,000 per second with normal activity
  – Sometimes extremely high on 64-bit systems with many processors without severely affecting performance
  – Sometimes unavoidable because of extremely heavy network packet traffic
Useful Processor Graphs

• Processor Composition
  – %DPC, %Interrupt, adjusted %Priv, %User on primary axis
  – DPCs Queued/sec, Interrupts/sec on secondary axis

• Processor Queue Depth & Context Switches
  – Processor Queue Length (primary axis) versus Context Switches/sec (secondary axis)

• Queued Deferred Procedure Calls (DPC) & Context Switches
  – DPCs Queued/sec (primary axis) versus Context Switches/sec (secondary axis)
Processor Queue Depth & Context Switches Overview

The graph illustrates the number of waiting threads and context switches per second over a period. The graph shows peaks and troughs indicating times of high and low activity, with annotations and labels for specific times. The data points are marked with distinct lines for different metrics, allowing for a clear visualization of performance metrics over time.
Queued Deferred Procedure Calls (DPC) & Context Switches Overview
Memory Considerations

• Can reduce I/O subsystem load and improve performance
  – Disk speed still in $10^{-3}$ seconds range
    • Memory speed in $10^{-9}$ seconds range
  – Also depends upon Read/Write ratio

• However, must balance with Windows memory requirements
  – If Windows does not have enough memory to operate efficiently, application cannot possibly operate efficiently either

• Windows has no “hit ratio” metric
  – Page Reads/sec can be an indicator (keep ~ zero) depending upon workload
Useful Memory Performance Counters

• Memory Object
  – Page Reads/sec (not just reads from paging file!)
    • SQL Server and Oracle I/Os not counted here
  – % Committed Bytes in Use, Committed Bytes
    • Helps find memory leaks for non-SQL processes
  – Available Bytes (Kbytes or Mbytes)
    • Should be at least 500 MB to allow for dynamic activities
    • Some books suggest 4 MB ok – it is NOT
    • 4 MB (10 MB on 2003 and 2008) available memory limit is “hard” limit to be avoided at ALL costs
    • System stops responding LONG before this point
    • Experience has shown that memory management activities begin to increase below 100 MB and continue to increase as available memory decreases
    • Remember that non-essential Windows activities stop when memory management occurs
Simple File Copy Page Reads Example
Useful Memory Performance Counters

- Page Faults/sec (both hard and soft faults)
- Transition Faults/sec (soft fault)
- Demand Zero Faults/sec (soft fault)
- Cache Faults/sec (either hard or soft faults)
- Cache Bytes
- System Code Resident Bytes
Useful Memory Graphs

• Memory Usage
  • System Driver Resident Bytes
  • System Code Resident Bytes
  • Pool Nonpaged Bytes
  • Pool Paged Resident Bytes
  • System Cache Resident Bytes

• PerfMon does not have a total process bytes metric or a total machine memory metric

Total Process Bytes = Total Physical Memory
(obtained from WMI or some other source) – Sum of the 5 entities above
Memory Usage Overview
Page Faults Composition

• Difficult to determine actual page fault composition
  – No definitive formula exists

• Reasonable approximation
  – Page Faults/sec = \textbf{Cache Faults/sec} + Demand Zero Faults/sec + Page Reads/sec + Page Writes/sec + Transition Faults/sec

• Remember that Page Reads/sec includes any read performed to satisfy a page fault \textit{regardless} of whether data from page file or any other flat file
Hard Faults Estimation

• No **definitive** formula exists

• Reasonable **approximation**

  – if Page Faults - Demand Zero Faults - Transition Faults - Write Copies - Page Reads - Cache Faults

    \[ \geq \text{Page Reads} + \text{Page Writes} + \text{Write Copies} \]

    \[ \text{then} \]

    \[ \text{Hard Faults} = \text{Page Faults} - \text{Demand Zero Faults} - \text{Transition Faults} - \text{Write Copies} - \text{Page Reads} - \text{Cache Faults} \]

  Else

  \[ \text{Hard Faults} = \text{Page Reads} + \text{Page Writes} + \text{Write Copies} \]
Useful Memory Graphs

• Memory Management
  • Write Copies/sec
  • Transition Faults/sec
  • Demand Zero Faults/sec
  • Cache Faults/sec
  • Page Reads/sec + Page Writes/sec
Memory Management Overview

Operations per second

- Memory Management I/Os
- Write Copies
- Transition Faults
- Demand Zero Faults
- Cache Faults

Slide 116
AWE User Experience #1

• 30 GB allocated to SQL Server on 32 GB system
  – System paged heavily after user activity increased
  – Little or no available memory for Windows
  – Could not attribute Windows memory usage to any process

• Reduced allocation to 28 GB
  – Paging ceased
  – Both system and SQL Server ran fine
  – Buffer cache hit ratio hardly affected
AWE User Experiences #2

• All but 65 MB allocated to SQL Server on 8 GB Windows 2000 system
  – System ran fine for several weeks, but no one noticed slow decreases in available memory

• Full text index creation executed
  – System cache size increased
  – Full Index creation programs require dedicated non-SQL memory

• System began to page heavily
  – Little or no available memory for Windows
  – System stopped functioning when available memory dropped to 4 MB
Available & Cache Memory

Bytes

- Available Memory
- Cache Memory
- Windows Memory Threshold
AWE User Experiences #2

- SQL memory allocation reduced to insure 678 MB available – problems ceased permanently
Disk Objects

• Most frequently (and badly) misunderstood Windows metrics

• Data reported for
  – Each physical disk or LUN (PhysicalDisk)
    • If using hardware RAID, many actual disk drives can appear as one “physical” disk or LUN
    • Use disk transfer times to detect contention within RAID itself (see Disk Performance Counters Incomplete slide)
  – Each logical disk partition (LogicalDisk)

• LogicalDisk also reports
  – Free Megabytes
  – % Free Space
Physical I/O Measurements

• SQL Server and Oracle can generate very large I/Os, e.g., 65,535 bytes or larger
  – 131 KB and larger I/Os have been observed
  – Sometimes between 200 and 800 KB

• HBAs can saturate fairly quickly under these conditions

• Service times can cause transfer times to be high, even when queuing does not occur
Useful Disk Performance Counters

- Physical Disk
  - **Avg. Disk sec/Transfer**
    - Should be 0.020 seconds (20 ms) at most unless I/O size huge
  - **% Idle Time**
    - Once this reaches zero, no more I/Os can be processed
    - Performance usually degrades as it approaches zero
  - **Disk Transfers/sec, Disk Bytes/sec**
    - Beware of disk specs because they usually cite very large I/Os
  - Read and Write-specific counters also valuable, especially when a read/write performance disparity exists or using RAID 5

- Logical Disk
  - Same counters available plus space-related ones
  - Useful when multiple logical drives reside on one physical LUN
“Published” Rules of Thumb

- Disk queue length > 2 for single drive or LUN
- Disk transfers per second > 80 for single drive
- Disk time per transfer > 0.090 seconds (90 milliseconds)
Interpreting Performance Counters

• Disk Queue lengths
  – By far, most commonly quoted and used disk performance measurement
    • Actually least useful, except when outrageously high
    • Use Transfer times and % Idle instead
  – Can no longer be interpreted as most Windows performance books suggest, i.e., disk in trouble when queue length > 2
    • Sometimes 16-32 physical disks comprise a single LUN
    • 14 or more queued I/Os common, even on well-performing I/O subsystems
  – Remember that Disk Queue Length includes I/Os in progress
Misunderstood PerfMon Counters

- Many PerfMon counters misunderstood, e.g., % Disk Time
  - Many people continue in 2011 to believe that this metric is a utilization metric! It is most definitely NOT!
- Sometimes perpetuated by PerfMon explanation
  - “% Disk Time is the percentage of elapsed time that the selected disk drive was busy servicing read or write requests.”
- % Disk Time actually = 100 * Avg. Disk Queue Length
  - Artificially constrained to 100% by PerfMon
  - Actually useless, but frequently referenced and interpreted as disk “busy” times
- Actual busy = 100 - % Idle Time
  - Can indicate a capacity constraint even when performance is excellent
Utilization versus Queue Depth Graph

- **No Match**

- **Match**

**Note:** Utilization peaks not matched by queue depth peaks
Unusual Utilization versus I/O Time Graph

Note: Utilization peaks not matched by response times
Useful Disk Graphs

- Disk Driver Activity
  - 100 - % Idle Time

- Disk I/O Times
  - Avg. Disk sec/Transfer

- Disk Queue Depths
  - Avg. Disk Queue Length

- Disk I/O Traffic
  - Avg. Disk Transfers/sec

- Disk I/O Byte Traffic
  - Avg. Disk Bytes/sec

- Include multiple disk LUNs per graph
Disk Driver Activity Overview

% Disk busy

- Disk 03 E
- Disk 01
- Disk 04 E
- Disk 03 F
- Disk 05 D
- Disk 02 F
- Disk 00 C
- Disk 06 D
Disk Queue Depths

Disk queue depth
Disk I/O Traffic Overview

Disk IOs/second

- Disk 04 E
- Disk 01
- Disk 03 F
- Disk 05 D
- Disk 03 E
- Disk 02 F
- Disk 00 C
- Disk 06 D
Disk I/O Byte Traffic Overview

Disk Bytes/second

Mon Jul 17 5:00 PM
Tue Jul 18 6:00 AM
10:00 AM 12:00 PM 2:00 PM 4:00 PM
6:00 PM
Wed Jul 19 7:00 AM
9:00 AM 11:00 AM 1:00 PM 3:00 PM 5:00 PM
Thu Jul 20 6:00 AM
10:00 AM 12:00 PM 2:00 PM 4:00 PM 6:00 PM

Disk 13
Disk 11
Disk 10
Disk 12
Disk 08
Disk 09
Disk 07

Disk Bytes/second
I/O Performance Counters Incomplete

• Some important metrics not measured directly
  – Avg. Disk Service Time per Transfer
  – Avg. Disk Queuing Time per Transfer

• Missing values can be computed using Utilization Law
Utilization Law

- $U = X \times S$
  - $U$ => utilization of a resource
  - $X$ => program completion rate
  - $S$ => average service time required of a resource

- System assumed to be in steady state
- Not perfect, but an excellent tool
Using Utilization Law to Compute Missing I/O-Related Times

• Restate the Utilization Law
  \[ S = \frac{U}{X} \]

• All calculations use PhysicalDisk counters
  – LogicalDisk counters can be used, if necessary

• *Italicized* entities are PerfMon counters

• Disk Utilization = 100 - % Idle Time

• Disk service time = Disk Utilization / Disk Transfers/sec

• Disk queue time = Avg. Disk sec/Transfer - Disk service time
# RAID Example Calculations #1 and #2

<table>
<thead>
<tr>
<th>LUN #1</th>
<th>LUN #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Utilization</td>
<td>Disk Utilization</td>
</tr>
<tr>
<td>36.57%</td>
<td>77.67%</td>
</tr>
<tr>
<td>Disk Transfers/sec</td>
<td>Disk Transfers/sec</td>
</tr>
<tr>
<td>0.65</td>
<td>30.89</td>
</tr>
<tr>
<td>Avg. Disk sec/Transfer</td>
<td>Avg. Disk sec/Transfer</td>
</tr>
<tr>
<td>2.0095 seconds!</td>
<td>2.4424 seconds!</td>
</tr>
<tr>
<td>Disk service time</td>
<td>Disk service time</td>
</tr>
<tr>
<td>0.3657 / 0.65 = 0.563 seconds or 563 milliseconds</td>
<td>0.7767 / 30.89 = 0.025 seconds or 25 milliseconds</td>
</tr>
<tr>
<td>Disk queue time</td>
<td>Disk queue time</td>
</tr>
<tr>
<td>2.0095 – 0.563 = 1.447 seconds or 1,447 milliseconds</td>
<td>2.4424 – 0.025 = 2.4174 seconds or 2,417 milliseconds</td>
</tr>
<tr>
<td>Bytes/Transfer</td>
<td>Bytes/Transfer</td>
</tr>
<tr>
<td>1,307</td>
<td>22,437</td>
</tr>
</tbody>
</table>
RAID Example #1 vs. #2

- I/O times (2.0095 vs. 2.4424) not that far apart despite being outrageously high
- Queuing occurred on both disks
- Low I/O rate of Disk #1 appears to contribute to high service times
  - 1,307 bytes should not require 563 milliseconds
- How could this happen?
RAID Example #1 vs. #2

- Disk #2 doing much more work
  - Utilization double that of Disk #1
  - I/O size 17 times larger, but not huge
  - Service time much more reasonable @ 25 milliseconds

- Problems began when faster processor complex attached
  - Customer blamed new processor for poor performance
  - Wanted vendor to take it back because architecture was supposedly defective and slower than original
  - In reality, it was MUCH faster and it was swamping the disk subsystem!

- Solution was to reconfigure EMC drives
  - Customer refused to state exactly what they changed
  - Probably multiple LUNs shared same physical drives
RAID Example Calculations #3

- Disk Utilization
  - 99.59%

- Disk Transfers/sec
  - 58.2

- Avg. Disk sec/Transfer
  - 0.7678 seconds or 767.8 milliseconds

- Disk service time
  - 0.9959 / 58.2 = 0.0171 seconds or 17.1 milliseconds

- Disk queue time
  - 0.7678 – 0.0171 = 0.7507 seconds or 750.7 milliseconds

- Bytes/Transfer
  - 168,536
RAID Example #3 Discussion

• 100% utilization is suspicious
  – RAID can provide good I/O response times at this utilization level
  – Obviously not in this case

• 17 millisecond service times good considering very large average I/O size

• Queuing was the problem

• Drives comprising LUN clearly saturated

• Needed another disk to understand issues completely
RAID Example #3 Discussion Continued

- Another Disk, #4, processed
  - 143.7 I/Os per second @ 7.8 milliseconds per I/O

- Queue time
  - 2.1 milliseconds

- Service time
  - 5.7 milliseconds

- Bytes/Transfer
  - 28,786

- Combining all disk metrics proved that HBA was approaching saturation

- Original suggestion
  - Add many drives to the I/O subsystem and spread load across the drives

- Solution was to add drives to #3 and HBAs
  - Neither alone would have corrected the problem
RAID Example Summary

• Both case studies involved SQL Server systems
• Neither was SQL Server’s, processor’s, or memory’s “fault”
• In case #2, query optimizations devised after crisis over
  – Optimizations were improvements, not solutions
Interpreting Performance Counters

• Disk transfers per second dependent upon I/O sizes, especially on database systems
  – **20 millisecond maximum** transfer times should be the objective
  – Typical transfer times should be less than 15-20 milliseconds unless I/O sizes are very large, i.e., > 65 KB
I/O Subsystem Capacity Planning

PerfMon Counters

- System: File Control byte counters summarize **entire** I/O subsystem usage, **including** DBMS usage

- **PerfMon Counters**
  - System: File Control Bytes/sec
  - System: File Control Read Bytes/sec
  - System: File Control Write Bytes/sec
Network Objects

• Network Interface object
  – Instance 0 is for loopback
  – One additional instance per NIC

• Network Segment PerfMon object
  – Must install Network Monitor (Professional) or Network Monitor Tools (Server)
  – Puts NIC in **promiscuous** mode, which can degrade overall network performance
    • Only use when other measuring tools unavailable
Useful Performance Counters

• Network Interface
  – Each NIC is an instance
  – Bytes Total/sec (for each NIC)
  – Packets/sec (for each NIC)
    • Packets usually saturate NICs long before byte traffic does
    • Especially true if outboard optimizations disabled
      – Many NICs were set this way by default
  – Sent and received metrics available for both
  – Can use these metrics to calculate utilization, which is sometimes useful
Useful NIC Graphs

• Network Card Byte Traffic
  – Bytes Total/sec

• Network Card Packet Traffic
  – Packets/sec

• Network Card Packet Traffic & Context Switches
  – Packets/sec versus Context Switches/sec
Network Card Byte Traffic Overview

Bytes/Second

TEAM ORC35public
Network Card Packet Traffic Overview

Packets/second

TEAM ORC35public
Network Card Packet Traffic & Context Switches Overview

- Packets/second
- Context Switches/second

Graph showing network card packet traffic and context switches over time, with different traffic and context switches thresholds marked.

Legend:
- TEAM ORC35public
- Context Switches/Sec
- Context Switch Threshold
Other Useful Network-Related Performance Objects

- Server
- Server Work Queues
- Redirector
  - Network Attached Storage (NAS)
  - Mapped disk drives
- TCP
- UDP
- Browser
Terminal Server Objects and Counters

• Terminal Services Session object has all Process/Thread object counters

• Has counters specific to a session
  – Thread totals
  – Input Bytes and Input WdBytes
  – Input Frames and Input WdFrames
  – Output Bytes and Output WdBytes
  – Output Frames and Output WdFrames
Useful Performance Counters

- Active Server Pages
  - Requests/sec
  - Requests Executing
  - Requests Queued
  - Used to derive average response times
    - Closest to response time service levels on IIS machines
Little’s Law

• $N = X \times R$
  – $N$ => average # customers at a service center
  – $X$ => program completion rate
  – $R$ => average elapsed time

• System assumed to be in steady state

• Not perfect, but an excellent tool
Active Server Pages

• Requests Executing
• Requests Queued
• Requests per Second
• Request Execution Time only records LAST transaction’s response time
• Use Little’s Law to compute average response time
• Avg. Response = (Requests Executing + Requests Queued) / Requests per Second
Useful ASP Graphs

• ASP Response
  – Calculated ASP response time versus Requests/sec

• ASP Requests in System
  – Requests Executing and Requests Queued
ASP Response vs. Volume Graph
Important Process Counters

- Private Bytes
- Virtual Bytes
- Working Set
- Page File Bytes
- Page Faults/sec
Capture Ratio Considerations

- Portion of overall processor consumption that can be attributed to specific processes and threads
- Closer to 100% best
- Capture ratios typically not very good using PerfMon data
  - To get close, either
    - System processes must be very consistent or
    - Sample rates would have to be very high
  - Threads must be associated with specific processes
  - Values of 70% or even lower fairly common
- Use Event Tracing for Windows to capture complete workload processor information
  - I/O information will still be imperfect because of lazy writes
Invaluable PerfMon Hardware Metrics

- % User Time (Processor)
- % Privileged Time (Processor)
- % Interrupt Time (Processor)
- % DPC Time (Processor)
- % Idle Time (each Disk)
- Avg. Disk sec/Transfer (each Disk)
- Avg. Disk sec/Write (each Disk)
- Available Bytes (Memory)
- Page Life Expectancy (SQLServer:Buffer Manager)
- Page Reads/sec (Memory & SQLServer:Buffer Manager)
# Invaluable PerfMon Hardware Metrics

- Potential problems exist if consistently...

## Table: Invaluable PerfMon Hardware Metrics

<table>
<thead>
<tr>
<th>Counter</th>
<th>Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Processor Time</td>
<td>&gt; 70%</td>
</tr>
<tr>
<td>% Privileged Time</td>
<td>&gt; 30% (Processor)</td>
</tr>
<tr>
<td>% Interrupt Time</td>
<td>&gt; 20% (Processor)</td>
</tr>
<tr>
<td>% DPC Time</td>
<td>&gt; 25% (Processor)</td>
</tr>
<tr>
<td>% Idle Time</td>
<td>&lt; 40% for any Disk LUN and especially SQL LUNs</td>
</tr>
<tr>
<td>Avg. Disk sec/Transfer</td>
<td>&gt; 0.040 seconds (40 ms)</td>
</tr>
<tr>
<td>Avg. Disk sec/Write</td>
<td>&gt; 0.040 seconds (40 ms)</td>
</tr>
<tr>
<td>Available Bytes</td>
<td>&lt; 500 MB (Memory)</td>
</tr>
<tr>
<td>Page Life Expectancy</td>
<td>&lt; 300 seconds (SQLServer:Buffer Manager)</td>
</tr>
</tbody>
</table>
How does all this apply to a Virtual Machine Environment?

• Regardless of the host OS, Windows still runs on the guest, so any performance analysis must include both the host and the guest at a minimum
  – Key to know what the host “sees” versus what the guest “sees”
  – These may be entirely different

• Sometimes multiple guests must be examined as well because one guest may be causing stress at the host and the guest in question may simply be a victim

• Be careful of rates because clock speed variations may skew rates
  – However, if one rate is okay, the others probably are too
How does all this apply to a Shared Disk Environment?

• This methodology also applies to shared disk subsystems

• One server can significantly and adversely affect another if any physical disks inside a SAN are shared

• Shared controllers may also cause poor performance

• Insure numbers make sense
  – Is the work presented by this server consistent with the performance?
  – Low volumes should not result in high utilizations or poor response times
  – Do the same levels of work result in similar utilizations and response times?
Other Sources of Windows Performance Data

- Windows Management Instrumentation (WMI)
- SysInternals Process Explorer
- Event Tracing for Windows (ETW)
- Kernrate and Krview
- COM+
- IIS logs
WMI

• Windows Management Instrumentation
• Obtains data via client script or application
  – Must have privileges to execute scripts on Server 2008
  – Use Elevation PowerToys for Windows Vista (or Windows 7)
• SQL-like query interface
  – Select * from Win32_OperatingSystem
• DOS Command line
  – cscript scriptfile.vbs
  – cscript scriptfile.vbs > output.txt
• Scriptomatic version 2.0 expedites learning
Important Primary WMI Objects

• Computer System Hardware
• Operating System
• Installed Applications
• Performance Counters
WMI Hardware Example

On error resume next

strComputer = "." ' local machine

Set objWMIService = GetObject("winmgmts:" & 
   "\{impersonationLevel=impersonate}\!\" & strComputer & "\root\cimv2")

Set colSettings = objWMIService.ExecQuery  
   ("Select * from Win32_OperatingSystem")

For Each objOperatingSystem in colSettings
   Wscript.Echo "Available Physical Memory: " & 

Next

⇒ Available Physical Memory: 2,269,996; Physical Memory Usable by Windows: 3,405,868 (commas added)
WMI OS Example

On error resume next

strComputer = "."

Set objWMIService = GetObject("winmgmts:\" & 
    "\{impersonationLevel=impersonate}\" & strComputer & "\root\cimv2\")

Set colOperatingSystems = objWMIService.ExecQuery
    ("Select * from Win32_OperatingSystem")

For Each objOperatingSystem in colOperatingSystems

Next

→ Microsoft Windows 7 Ultimate 6.1.7600 0.0
WMI Installed Application Example

On error resume next

strComputer = "."

Set objWMIService = GetObject("winmgmts:" & 
"{impersonationLevel=impersonate}!\" & strComputer & 
"\root\cimv2")

Set colSoftware = objWMIService.ExecQuery( 
("Select * from Win32_Product")

For Each objSoftware in colSoftware

    WScript.Echo objSoftware.Name & "," & objSoftware.Version

Next

→ Microsoft SQL Server 2008 Database Engine Services; 
  10.1.2531.0 (among others)
WMI Analysis Issues

• Determining whether hyperthreading active quite difficult
  – Neither WMI Processor nor WMI BIOS information tells us

• Logical processor properties documented in SDK only available in Vista and Server 2008

• Cannot use WMI to determine hardware RAID type
WMI Analysis Issues

• Cannot use WMI to determine whether a controller is using write-through or write-back
  – Supposedly another tool will identify this, but it is easier to view the disk drive properties in Device Manager
Sysinternals Process Explorer

- Part of the Sysinternals Suite developed by Mark Russinovich and Bryce Cogswell
  - Russinovich is one of the authors of *Windows Internals 5*, probably the definitive work on Windows architecture and functionality

- “Process Explorer shows you information about which handles and DLLs processes have opened or loaded.”

- Process Explorer **freely** available at

- Entire suite accessible from this location
  - Process Monitor can provide more detailed usage information
### SysInternals Process Explorer Example

<table>
<thead>
<tr>
<th>Process</th>
<th>PID</th>
<th>CPU</th>
<th>Description</th>
<th>Company Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Idle Process</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupts</td>
<td>n/a</td>
<td>100</td>
<td>Hardware Interrupts</td>
<td></td>
</tr>
<tr>
<td>DPCs</td>
<td>n/a</td>
<td>0</td>
<td>Deferred Procedure Calls</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>smss.exe</td>
<td>244</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ccmw.exe</td>
<td>376</td>
<td>0.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wininit.exe</td>
<td>482</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>svchost.exe</td>
<td>684</td>
<td>38.53</td>
<td>Hardware Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>2032</td>
<td></td>
<td>Windows Explorer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>processexp64.exe</td>
<td>576</td>
<td></td>
<td>Synternals Process Explorer</td>
<td>Synternals - <a href="http://www.sysinter">www.sysinter</a>...</td>
</tr>
<tr>
<td>FlashUtil10_ActiveX</td>
<td>596</td>
<td></td>
<td>Synternals Process Explorer</td>
<td>Synternals - <a href="http://www.sysinter">www.sysinter</a>...</td>
</tr>
<tr>
<td>flash.exe</td>
<td>2800</td>
<td></td>
<td>Adobe® Flash® Player Installer/Uninstaller 10.1 r82</td>
<td>Adobe Systems, Inc.</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>764</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>9228</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>3582</td>
<td></td>
<td></td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>audiohost.exe</td>
<td>924</td>
<td>0.1</td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>dwm.exe</td>
<td>3816</td>
<td>2.7</td>
<td>Desktop Window Manager</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>980</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>3358</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>as.exe</td>
<td>1240</td>
<td></td>
<td>Spooler SubSystem App</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>1272</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>atashare.exe</td>
<td>1364</td>
<td></td>
<td>WebEx Host for Support Center</td>
<td>Webex Communications, I...</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>1424</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>IntUpdateService.exe</td>
<td>1502</td>
<td></td>
<td>IntUpdateService</td>
<td>Intu Inc.</td>
</tr>
<tr>
<td>ndm.exe</td>
<td>1700</td>
<td></td>
<td>Machine Debug Manager</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>salserv.exe</td>
<td>1784</td>
<td></td>
<td>SQL Server Windows NT - 64 Bit</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>cssvchst.exe</td>
<td>1823</td>
<td>12.84</td>
<td>Symantec Service Framework</td>
<td>Symantec Corporation</td>
</tr>
<tr>
<td>cssvchst.exe</td>
<td>3784</td>
<td>8.31</td>
<td></td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>rundll32.exe</td>
<td>4048</td>
<td>20.40</td>
<td>Windows host process (Rundll32)</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>rundll32.exe</td>
<td>3080</td>
<td>5.29</td>
<td>Windows host process (Rundll32)</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>VProSvc.exe</td>
<td>1916</td>
<td></td>
<td>Service Module</td>
<td>Symantec Corporation</td>
</tr>
<tr>
<td>PDAgent.exe</td>
<td>1996</td>
<td>0.76</td>
<td>PDAgent Module</td>
<td>Raxco Software, Inc.</td>
</tr>
<tr>
<td>perfhost.exe</td>
<td>1392</td>
<td></td>
<td>x86 Performance Counter Host</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>544</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>sqlbrowser.exe</td>
<td>1580</td>
<td></td>
<td>SQL Browser Service EXE</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>sqlwrt.exe</td>
<td>1548</td>
<td></td>
<td>SQL Server VSS Writer - 64 Bit</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>1136</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>3006</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>5000</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>SymSnapServicex64.exe</td>
<td>1556</td>
<td></td>
<td>Symantec Snapshot Service</td>
<td>Symantec</td>
</tr>
<tr>
<td>TASKhost.exe</td>
<td>3824</td>
<td></td>
<td>Host Process for Windows Tasks</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>SearchIndexer.exe</td>
<td>983</td>
<td></td>
<td>Microsoft Windows Search Indexer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>wininet.exe</td>
<td>3569</td>
<td></td>
<td>Windows Media Player Network Sharing Service</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>4648</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>4904</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>4332</td>
<td></td>
<td>Microsoft Distributed Transaction Coordinator Service</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>dillhost.exe</td>
<td>2932</td>
<td></td>
<td>COM Surrogates</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>media.exe</td>
<td>4836</td>
<td></td>
<td>PDAEngine Module</td>
<td>Raxco Software, Inc.</td>
</tr>
<tr>
<td>pdagents.exe</td>
<td>3736</td>
<td></td>
<td>PDAgent1 Module</td>
<td>Raxco Software, Inc.</td>
</tr>
<tr>
<td>pdagents.exe</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdagents.exe</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# SysInternals Process Explorer

<table>
<thead>
<tr>
<th>Process</th>
<th>PID</th>
<th>CPU</th>
<th>Description</th>
<th>Company Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Idle Process</td>
<td>0</td>
<td>64.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>n/a</td>
<td>0.78</td>
<td>Hardware Interrupts</td>
<td></td>
</tr>
<tr>
<td>DPCs</td>
<td>n/a</td>
<td></td>
<td>Deferred Procedure Calls</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>smss.exe</td>
<td>244</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csrss.exe</td>
<td>376</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wininit.exe</td>
<td>452</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>services.exe</td>
<td>504</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>svchost.exe</td>
<td>684</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>2092</td>
<td></td>
<td>Windows Explorer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>procepx.exe</td>
<td>576</td>
<td></td>
<td>Sysinternals Process Explorer</td>
<td>Sysinternals - <a href="http://www.sysinter">www.sysinter</a>...</td>
</tr>
<tr>
<td>procepx64.exe</td>
<td>3424</td>
<td>0.78</td>
<td>Sysinternals Process Explorer</td>
<td>Sysinternals - <a href="http://www.sysinter">www.sysinter</a>...</td>
</tr>
<tr>
<td>FlashUtil10i_ActiveX....</td>
<td>2900</td>
<td></td>
<td>Adobe® Flash® Player Installer/Uninstaller 10.1 r82</td>
<td>Adobe Systems, Inc.</td>
</tr>
<tr>
<td>WmiPrvSE.exe</td>
<td>3124</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WmiPrvSE.exe</td>
<td>3080</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>svchost.exe</td>
<td>764</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>828</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>audiogd.exe</td>
<td>3584</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>svchost.exe</td>
<td>924</td>
<td></td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>dwm.exe</td>
<td>3816</td>
<td></td>
<td>Desktop Window Manager</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>wispits.exe</td>
<td>1436</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## SysInternals Process Explorer

<table>
<thead>
<tr>
<th>Process Name</th>
<th>PID</th>
<th>Description</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>tasklist.exe</td>
<td>822</td>
<td>Task Process for Windows Tasks</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>SearchIndexer.exe</td>
<td>588</td>
<td>Microsoft Windows Search Indexer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>SearchProtocolHost.exe</td>
<td>3320</td>
<td>Microsoft Windows Search Protocol Host</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>SearchFilterHost.exe</td>
<td>3296</td>
<td>Microsoft Windows Search Filter Host</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>wmpnetwk.exe</td>
<td>3692</td>
<td>Windows Media Player Network Sharing Service</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>4648</td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svc.exe</td>
<td>4904</td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>msdtc.exe</td>
<td>4332</td>
<td>Microsoft Distributed Transaction Coordinator Service</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>dllhost.exe</td>
<td>2932</td>
<td>COM Surrogate</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>PDEngine.exe</td>
<td>4836</td>
<td>PDEngine Module</td>
<td>Raxco Software, Inc.</td>
</tr>
<tr>
<td>PDAgentS1.exe</td>
<td>3736</td>
<td>PDAgentS1 Module</td>
<td>Raxco Software, Inc.</td>
</tr>
<tr>
<td>VSSVC.exe</td>
<td>1724</td>
<td>Microsoft® Volume Shadow Copy Service</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>dllhost.exe</td>
<td>4632</td>
<td>COM Surrogate</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>1528</td>
<td>Host Process for Windows Services</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Isass.exe</td>
<td>512</td>
<td>Local Security Authority Process</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>csm.exe</td>
<td>520</td>
<td></td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>winlogon.exe</td>
<td>608</td>
<td></td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>3848</td>
<td>Windows Explorer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>ISUSPM.exe</td>
<td>3324</td>
<td>Macrovision Software Manager</td>
<td>Macrovision Corporation</td>
</tr>
<tr>
<td>Weather.exe</td>
<td>880</td>
<td></td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>lexplore.exe</td>
<td>3428</td>
<td>Internet Explorer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>4656</td>
<td>Internet Explorer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>explorer.exe</td>
<td>2440</td>
<td>17.05 Internet Explorer</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>PerfectDisk.exe</td>
<td>4780</td>
<td>0.78 PerfectDisk 11</td>
<td>Raxco Software, Inc.</td>
</tr>
<tr>
<td>taskmgr.exe</td>
<td>3184</td>
<td>Windows Task Manager</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>SnippingTool.exe</td>
<td>4636</td>
<td>2.33 Snipping Tool</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>VProTray.exe</td>
<td>3876</td>
<td>Tray Application</td>
<td>Symantec Corporation</td>
</tr>
<tr>
<td>juser.exe</td>
<td>3548</td>
<td>Java(TM) Update Scheduler</td>
<td>Sun Microsystems, Inc.</td>
</tr>
<tr>
<td>ielowutil.exe</td>
<td>3720</td>
<td>Internet Low-Mic Utility Tool</td>
<td>Microsoft Corporation</td>
</tr>
</tbody>
</table>
Event Tracing for Windows

- Records tracing data already available from within OS itself
  - Instrumentation improves dramatically with each Windows release

- Useful for understanding
  - Why kernel time too high
    - Which drivers or Windows modules used processor most
  - Files being hit hardest and which processes are using them
  - Process causing excessive page faults or context switches
  - On Vista, Windows 7, and Server 2008 can collect stack traces
    - OS symbols required
Event Tracing for Windows

• Use Windows Performance Toolkit (WPT)
  – Released February 2008
  – Can collect on XP/2003/Vista/2008/Windows 7
  – Analysis ONLY on Vista/2008/Windows 7
  – No longer need
    • Tracelog, TraceRpt, or SPA
  – Free download
  – Documentation
ETW Trace Classes

- Kernel
- Disk I/O
- File I/O
- Hardware Configuration
- Image Load/Unload
- Page Faults
- Process Create/Delete
- Thread Create/Delete
- Registry Activity
- TCP/UDP Activity
Simplified WPT ETW Collection

• To collect on XP/2003
  – Copy `xperf.exe` and `perfctrl.dll` from a Vista/2008/Windows 7 installation to a separate directory on the target system

• Start and Stop are valid commands, but others preferred
  – On and D options perform similar, but expanded functions

• To enable and initiate collection of Base and FileIO groups in addition to the REGISTRY flag (*no spaces between ‘+’ signs*)
  
  `xperf –on LOADER + PROC_THREAD + DISK_IO + HARD_FAULTS + DPC + INTERRUPT + CSWITCH + PERF_COUNTER + FILE_IO_INIT + REGISTRY -maxbuffers 1024`

• Stops and merges sessions into trace.etl file
  
  `xperf –d etwtrace.etl`
  – Can be used to merge multiple trace files into a single file
ETW Analysis

• Use XperfView to analyze data
  – Must install package on Vista/2008/Windows 7
  – **Must** use Vista/2008/Windows 7 for analysis **regardless** of data origin
XperfView

[Image of XperfView software interface with various graphs and charts, including CPU Sampling by CPU, CPU Sampling by Process, Disk I/O, and CPU Sampling by Thread.]
Collecting File Usage Information

• To enable and initiate collection of Base and FileIO groups in addition to the REGISTRY flag (no spaces between ‘+’ signs)

  xperf –on LOADER + PROC_THREAD + DISK_IO + 
  HARD_FAULTS + PERF_COUNTER + FILE_IO_INIT - 
  maxbuffers 1024

• Stops and merges sessions into trace.etl file

  xperf –d fileiotrace.etl

• Use XperfView to create a summary table
  – Can specify all disks or just specific ones
  – Can highlight and copy sections of the summary table using the clipboard to another program
### XperfView File Summary Table

<table>
<thead>
<tr>
<th>Path Name</th>
<th>Service Time (us)</th>
<th>Size</th>
<th>Read Counts</th>
<th>Read Service Time (us)</th>
<th>Read Size</th>
<th>Write Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\Device\HarddiskVolume8\SQLData\WinPerfDataSQL_Data.mdf</code></td>
<td>487274397.921</td>
<td>12680364032</td>
<td>121787</td>
<td>349738405.665</td>
<td>7874215936</td>
<td>76948</td>
</tr>
<tr>
<td><code>\Device\HarddiskVolume8\SQLData\UKTraceP2.mdf</code></td>
<td>28610.348</td>
<td>16384</td>
<td>2</td>
<td>28610.348</td>
<td>16384</td>
<td>0</td>
</tr>
<tr>
<td><code>\Device\HarddiskVolume8\SQLData\PlanText.mdf</code></td>
<td>19356.121</td>
<td>16384</td>
<td>2</td>
<td>19356.121</td>
<td>16384</td>
<td>0</td>
</tr>
<tr>
<td><code>\Device\HarddiskVolume8\SQLData\BeforeAndAfterComparison...</code></td>
<td>23203.729</td>
<td>16384</td>
<td>2</td>
<td>23203.729</td>
<td>16384</td>
<td>0</td>
</tr>
<tr>
<td><code>\Device\HarddiskVolume8\SQLData</code></td>
<td>605.429</td>
<td>4096</td>
<td>0</td>
<td>0.000</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### XperfView File Summary Table

<table>
<thead>
<tr>
<th>Line</th>
<th>Path Name</th>
<th>Service Time (ms)</th>
<th>Size</th>
<th>Read Counts</th>
<th>Read Service Time (us)</th>
<th>Read Size</th>
<th>Write Counts</th>
<th>Write Service Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\Device\HarddiskVolume8\SQLData\WinPerfData\SQL_Data.mdf</td>
<td>487274397.921</td>
<td>12600364032</td>
<td>121787</td>
<td>349738405.656</td>
<td>787421936</td>
<td>76948</td>
<td>13753992.256</td>
</tr>
<tr>
<td>2</td>
<td>\Device\HarddiskVolume1\SQLData\WinPerfData\SQL2_Data.mdf</td>
<td>400779338.449</td>
<td>1031417856</td>
<td>97913</td>
<td>29458271.154</td>
<td>6400704512</td>
<td>64763</td>
<td>10612065.295</td>
</tr>
<tr>
<td>3</td>
<td>\Device\HarddiskVolume10\SQLData\tempdev2_2.mdf</td>
<td>284400805.466</td>
<td>6740900880</td>
<td>301440</td>
<td>238520212.393</td>
<td>3385917440</td>
<td>36870</td>
<td>46155069.073</td>
</tr>
<tr>
<td>4</td>
<td>\Device\HarddiskVolume6\pagefile.sys</td>
<td>211423431.205</td>
<td>2515261696</td>
<td>37644</td>
<td>168984090.890</td>
<td>533872640</td>
<td>2245</td>
<td>41433942.225</td>
</tr>
<tr>
<td>5</td>
<td>\Device\HarddiskVolume2\SQLData\SQL2008\tempdb.mdf</td>
<td>189294662.712</td>
<td>6314696704</td>
<td>278485</td>
<td>151528535.867</td>
<td>37767905.845</td>
<td>35533</td>
<td>37767905.845</td>
</tr>
<tr>
<td>6</td>
<td>\Device\HarddiskVolume2\SQLData\WinPerfData\SQL_Log.ldf</td>
<td>63987673.553</td>
<td>5031657472</td>
<td>48</td>
<td>6445290</td>
<td>196608</td>
<td>57757</td>
<td>63981428.263</td>
</tr>
<tr>
<td>7</td>
<td>\Device\HarddiskVolume2\SQLData\SQL2_Log.ldf</td>
<td>49283455.456</td>
<td>3919330034</td>
<td>42</td>
<td>38015910</td>
<td>180224</td>
<td>40595</td>
<td>49245493.546</td>
</tr>
<tr>
<td>8</td>
<td>\Device\HarddiskVolume2\Studens\Win\Data[EZ]\XIA\IndexanalysisCALas\CompanyID\AndLinkCTrace..</td>
<td>13314258.562</td>
<td>1073872896</td>
<td>16536</td>
<td>13314258.562</td>
<td>1073872896</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>\Device\HarddiskVolume2\Studens\Win\Data[EZ]\XIA\IndexanalysisCALas\CompanyID\AndLinkCTrace..</td>
<td>12601176.395</td>
<td>1073872896</td>
<td>16536</td>
<td>12601176.395</td>
<td>1073872896</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>\Device\HarddiskVolume2\Studens\Win\Data[EZ]\XIA\IndexanalysisCALas\CompanyID\AndLinkCTrace..</td>
<td>11512305.947</td>
<td>1073872896</td>
<td>16536</td>
<td>11512305.947</td>
<td>1073872896</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>11</td>
<td>\Device\HarddiskVolume2\Studens\Win\Data[EZ]\XIA\IndexanalysisCALas\CompanyID\AndLinkCTrace..</td>
<td>11429286.632</td>
<td>1073872896</td>
<td>16536</td>
<td>11429286.632</td>
<td>1073872896</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>12</td>
<td>\Device\HarddiskVolume5\LogFile</td>
<td>5652592.089</td>
<td>7811072</td>
<td>0</td>
<td>0.000</td>
<td>0</td>
<td>1222</td>
<td>5652592.089</td>
</tr>
<tr>
<td>13</td>
<td>\Device\HarddiskVolume5\SMft</td>
<td>4654396.318</td>
<td>158282592</td>
<td>571</td>
<td>2160365.401</td>
<td>13651968</td>
<td>553</td>
<td>2494030.917</td>
</tr>
<tr>
<td>14</td>
<td>\Device\HarddiskVolume5\Program Files\Microsoft SQL Server\MSSQL\MSSQLSERVER\MSSQL\Bin...</td>
<td>4165630.302</td>
<td>38533296</td>
<td>1339</td>
<td>4165630.302</td>
<td>38533296</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>15</td>
<td>\Device\HarddiskVolume5\Studens\Win\Data[EZ]\XIA\IndexanalysisCALas\CompanyID\AndLinkCTrace..</td>
<td>4137714.785</td>
<td>322830336</td>
<td>4926</td>
<td>4137714.785</td>
<td>322830336</td>
<td>0</td>
<td>0.000</td>
</tr>
</tbody>
</table>
XperfView

- Provides excellent default graphs of most processor and I/O categories
  - CPU Sampling by CPU
  - CPU Sampling by Process
  - CPU Sampling by Thread
  - CPU Usage by CPU
  - CPU Usage by Process
  - CPU Usage by Thread
  - CPU Scheduling
  - Disk I/O
  - Disk Utilization
  - Disk Utilization by Process
  - Process Lifetimes
  - DPC CPU Usage
  - Interrupt CPU Usage
  - Hard Faults
XperfView

• Graphs have dropdown selections that can reduce displayed detail

• Right click on graph to choose additional display options depending upon data
  – Simple Summary Table
  – Summary Table
  – Detail Graph

• Using the Windows Sample Profiler with Xperf blog article describes how to use stack walking

• May obviate use of kernrate on Server 2008 and above
Kernrate

• Sample profiling tool helps identify where CPU time being spent

• Both Kernel and user mode processes can be profiled
  – Separately or simultaneously

• Especially helpful when kernel usage inexplicably high
  – Multiple collections required
  – Initial one determines which drivers and processes consuming most processor
  – Use in conjunction with symbols to determine Windows kernel modules where time spent
    • Modules can identify types of activity which can lead back to app
Krview

- Aids in viewing and comparing output data
  - Available from MS
  - Contains both Krview and Kernrate for all 3 platforms (x86, x64, and IA64)
  - Provides excellent Excel summary of results
  - Also contains excellent documentation for both tools
Kernrate & Krview

• Minimum recommended uniprocessor kernrate options for use with krview
  Kernrate –v 2 –b 4 –f –ts –yr filename.kv

• Minimum recommended multiprocessor kernrate options for use with krview
  Kernrate –v 2 –b 4 –m –f –ts –yr filename.kv
Kernrate & Krview Graphical Output

Top 5 Privileged Modules by CPU Utilization

- Intelppm
- NTKernel
- HAL
- Win32k
- Rdpdd

Percent Privileged Time, Excluding Idle

All CPUs
## Kernrate & Krview Tabular Output

<table>
<thead>
<tr>
<th>Element</th>
<th>StartValue</th>
<th>StopValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Physical Memory</td>
<td>1,155,829,760</td>
<td>1,137,754,112</td>
</tr>
<tr>
<td>Available Page File(s)</td>
<td>3,263,303,680</td>
<td>3,253,211,136</td>
</tr>
<tr>
<td>Available Virtual Memory</td>
<td>2,132,881,408</td>
<td>2,131,832,832</td>
</tr>
<tr>
<td>Available Extended Virtual Memory</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Context Switches</td>
<td>17,900,639</td>
<td>16,824,505</td>
</tr>
<tr>
<td>System Calls</td>
<td>141,668,230</td>
<td>127,172,234</td>
</tr>
<tr>
<td>Page Faults</td>
<td>7,981,427</td>
<td>7,463,742</td>
</tr>
<tr>
<td>I/O Read Operations</td>
<td>12,755,982</td>
<td>7,090,682</td>
</tr>
<tr>
<td>I/O Write Operations</td>
<td>362,949</td>
<td>350,910</td>
</tr>
<tr>
<td>I/O Other Operations</td>
<td>5,972,575</td>
<td>5,731,577</td>
</tr>
<tr>
<td>I/O Read Bytes</td>
<td>2,990,805,598</td>
<td>2,913,653,987</td>
</tr>
<tr>
<td>I/O Write Bytes</td>
<td>127,490,504</td>
<td>122,763,241</td>
</tr>
<tr>
<td>I/O Other Bytes</td>
<td>711,256,458</td>
<td>691,258,274</td>
</tr>
</tbody>
</table>
Oracle

• Most Oracle Windows implementations appear to be on 64-bit Windows

• Performance counters minimal and usually only for one instance
  – Many sites use multiple instances
  – Should use other tools to gather Oracle-specific data
SQL Server

- Provides large number of objects and counters
  - Significantly improved on SQL Server 2005/2008
  - Still need other tools that come with SQL Server for in-depth analysis of performance problems
    - SQL Trace
    - Dynamic Management Views and Functions (DMVs and DMFs)
    - SQL Profiler
COM+

- .Net counters may provide some insights into this code
- Most analyses require tools specifically designed to analyze COM+ objects, their queues, etc.
- AppMetrics by Xtremesoft is an example of such a tool
  - Author has no connection whatsoever to Xtremesoft
IIS Logs

- IIS options can be activated which will write response time-related information to its logs
- Logparser can be used to summarize important information from these logs
  - [http://support.microsoft.com/kb/910447](http://support.microsoft.com/kb/910447)
  - [http://forums.iis.net/t/1155655.aspx](http://forums.iis.net/t/1155655.aspx)
Summary

• Windows operating system very complex, as are the applications that run on it

• Plethora of performance counters can make Windows performance analysis appear overwhelming

• Once understood, PerfMon metrics are essential for understanding nature of problem very quickly

• Despite large number of performance counters, some values still missing or need to be calculated

• Other tools such as Windows Performance Toolkit, Process Explorer, or Logparser may need to be utilized
References

- Mark Friedman and Odysseas Pentakalos, Windows 2000 Performance Guide
- David Solomon and Mark Russinovich, Microsoft Windows Internals, Fifth Edition
## Kernel Priorities from Win32 Priorities

<table>
<thead>
<tr>
<th>Thread Priority</th>
<th>Real Time</th>
<th>High</th>
<th>Above Normal</th>
<th>Normal</th>
<th>Below Normal</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Critical</td>
<td>31</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Highest</td>
<td>26</td>
<td>15</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Above Normal</td>
<td>25</td>
<td>14</td>
<td>11</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Normal</td>
<td>24</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Below Normal</td>
<td>23</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Lowest</td>
<td>22</td>
<td>11</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Idle</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Memory Limits – 32-bit

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Limit in 32-bit Windows</th>
</tr>
</thead>
</table>
| User-mode virtual address space for each 32-bit process | 2 GB  
Up to 3 GB with IMAGE_FILE_LARGE_ADDRESS_AWARE and 4GT (4-Gigabyte Tuning) |
| Kernel-mode virtual address space | 2 GB  
From 1 GB to a maximum of 2 GB with 4GT |
| Paged pool | Limited only by kernel mode virtual address space, starting with Windows Vista.  
**Windows Server 2003:** 530 MB  
**Windows XP:** 490 MB  
**Windows 2000:** 350 MB |
| Non-paged pool | Limited only by kernel mode virtual address space and physical memory, starting with Windows Vista.  
**Windows Server 2003, Windows XP, and Windows 2000:** 256 MB, or 128 MB with 4GT. |
| System cache virtual address space (physical size limited only by physical memory) | Limited only by kernel mode virtual address space, starting with Windows Vista.  
**Windows Server 2003, Windows XP, and Windows 2000:** 860 MB with LargeSystemCache registry key set and without 4GT; up to 448 MB with 4GT. |
## Memory Limits – 64-bit

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Limit in 64-bit Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-mode virtual address space for each 32-bit process</td>
<td>2 GB&lt;br&gt;4 GB with IMAGE_FILE_LARGE_ADDRESS_AWARE</td>
</tr>
<tr>
<td>User-mode virtual address space for each 64-bit process</td>
<td>2 GB&lt;br&gt;x64: 8 TB with&lt;br&gt;IMAGE_FILE_LARGE_ADDRESS_AWARE&lt;br&gt;Intel IPF: 7 TB with&lt;br&gt;IMAGE_FILE_LARGE_ADDRESS_AWARE</td>
</tr>
<tr>
<td>Kernel-mode virtual address space</td>
<td>8 TB</td>
</tr>
<tr>
<td>Paged pool</td>
<td>128 GB starting with Windows Vista.&lt;br&gt;<strong>Windows Server 2003 and Windows XP:</strong> Up to 128 GB depending on configuration and RAM.&lt;br&gt;<strong>Windows 2000:</strong> Not applicable</td>
</tr>
<tr>
<td>Non-paged pool</td>
<td><strong>Windows Server 2008:</strong> 75% of RAM up to a maximum of 128 GB.&lt;br&gt;<strong>Windows Vista:</strong> 40% of RAM up to a maximum of 128 GB.&lt;br&gt;<strong>Windows Server 2003 and Windows XP:</strong> Up to 128 GB depending on configuration and RAM.&lt;br&gt;<strong>Windows 2000:</strong> Not applicable</td>
</tr>
<tr>
<td>System cache virtual address space (physical size limited only by physical memory)</td>
<td>Always 1 TB regardless of physical RAM, starting with Windows Vista.&lt;br&gt;<strong>Windows Server 2003 and Windows XP:</strong> Up to 1 TB depending on configuration and RAM.&lt;br&gt;<strong>Windows 2000:</strong> Not applicable</td>
</tr>
</tbody>
</table>
## Physical Memory Limits: Windows Server 2003

<table>
<thead>
<tr>
<th>Version</th>
<th>Limit in 32-bit Windows</th>
<th>Limit in 64-bit Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server 2003, Datacenter Edition SP2</td>
<td>128 GB</td>
<td>2 TB</td>
</tr>
<tr>
<td></td>
<td>64 GB with 4GT</td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003, Enterprise Edition SP2</td>
<td>64 GB</td>
<td>2 TB</td>
</tr>
<tr>
<td>Windows Storage Server 2003, Enterprise Edition</td>
<td>8 GB</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Windows Storage Server 2003</td>
<td>4 GB</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Windows Server 2003, Datacenter Edition R2</td>
<td>128 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td>Windows Server 2003, Datacenter Edition SP1</td>
<td>16 GB with 4GT</td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003, Enterprise Edition R2</td>
<td>64 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td>Windows Server 2003, Enterprise Edition SP1</td>
<td>16 GB with 4GT</td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003, Standard Edition R2</td>
<td>4 GB</td>
<td>32 GB</td>
</tr>
<tr>
<td>Windows Server 2003, Standard Edition SP1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003, Datacenter Edition</td>
<td>128 GB</td>
<td>512 GB</td>
</tr>
<tr>
<td></td>
<td>16 GB with 4GT</td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003, Enterprise Edition</td>
<td>32 GB</td>
<td>64 GB</td>
</tr>
<tr>
<td></td>
<td>16 GB with 4GT</td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003, Standard Edition</td>
<td>4 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>Windows Server 2003, Web Edition</td>
<td>2 GB</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Windows Small Business Server 2003</td>
<td>4 GB</td>
<td>128 GB</td>
</tr>
<tr>
<td>Windows Compute Cluster Server 2003</td>
<td>Not applicable</td>
<td>128 GB</td>
</tr>
</tbody>
</table>
## Windows 2000 Virtual and Physical Memory Support

<table>
<thead>
<tr>
<th>Edition</th>
<th>Virtual memory</th>
<th>Maximum physical memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional</td>
<td>4 GB</td>
<td>4 GB</td>
</tr>
<tr>
<td>Server</td>
<td>4 GB</td>
<td>4 GB</td>
</tr>
<tr>
<td>Advanced Server</td>
<td>4 GB</td>
<td>8 GB</td>
</tr>
<tr>
<td>Datacenter Server</td>
<td>4 GB</td>
<td>32 GB, if hardware supports PAE</td>
</tr>
</tbody>
</table>
# Physical Memory Limits: Windows Server 2008

<table>
<thead>
<tr>
<th>Version</th>
<th>Limit in 32-bit Windows</th>
<th>Limit in 64-bit Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server 2008 Datacenter (full installation)</td>
<td>64 GB</td>
<td>2 TB</td>
</tr>
<tr>
<td>Windows Server 2008 Datacenter (Server Core installation)</td>
<td>64 GB</td>
<td>2 TB</td>
</tr>
<tr>
<td>Windows Server 2008 Enterprise</td>
<td>64 GB</td>
<td>2 TB</td>
</tr>
<tr>
<td>Windows Server 2008 Standard</td>
<td>4 GB</td>
<td>32 GB</td>
</tr>
<tr>
<td>Windows Server 2008 for Itanium-Based Systems</td>
<td>Not applicable</td>
<td>2 TB</td>
</tr>
<tr>
<td>Windows Web Server 2008</td>
<td>4 GB</td>
<td>32 GB</td>
</tr>
</tbody>
</table>
## Physical Memory Limits: Windows Vista

<table>
<thead>
<tr>
<th>Version</th>
<th>Limit in 32-bit Windows</th>
<th>Limit in 64-bit Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Vista Ultimate</td>
<td>4 GB</td>
<td>128 GB</td>
</tr>
<tr>
<td>Windows Vista Enterprise</td>
<td>4 GB</td>
<td>128 GB</td>
</tr>
<tr>
<td>Windows Vista Business</td>
<td>4 GB</td>
<td>128 GB</td>
</tr>
<tr>
<td>Windows Vista Home Premium</td>
<td>4 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>Windows Vista Home Basic</td>
<td>4 GB</td>
<td>8 GB</td>
</tr>
<tr>
<td>Windows Vista Starter</td>
<td>1 GB</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
Physical Memory Limits: Windows 7

<table>
<thead>
<tr>
<th>Version</th>
<th>Limit in 32-bit Windows</th>
<th>Limit in 64-bit Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7 Ultimate</td>
<td>4 GB</td>
<td>192 GB</td>
</tr>
<tr>
<td>Windows 7 Enterprise</td>
<td>4 GB</td>
<td>192 GB</td>
</tr>
<tr>
<td>Windows 7 Professional</td>
<td>4 GB</td>
<td>192 GB</td>
</tr>
<tr>
<td>Windows 7 Home Premium</td>
<td>4 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>Windows 7 Home Basic</td>
<td>4 GB</td>
<td>8 GB</td>
</tr>
<tr>
<td>Windows 7 Starter</td>
<td>2 GB</td>
<td>2 GB</td>
</tr>
</tbody>
</table>