Metropolia

Olli Eklund

Analyzing Database Server Performance Using Windows Performance Monitor

Helsinki Metropolia University of Applied Sciences
Bachelor of Engineering
Information Technology
Bachelor's Thesis
14 February 2013

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Olli Eklund Tietokantapalvelimen suorituskyvyn analysointi Windows Performance Monitorin avulla 35 sivua 14.2.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmointi
Ohjaaja	lehtori Olli Hämäläinen

Insinöörityön tavoitteena oli etsiä rajoitettu määrä suorituskykymittareita Windows Performance Monitorille, jotta voitaisiin havaita resurssipullonkauloja Oracle-tietokantapalvelimella vaativien tietokantaoperaatioiden aikana. Projektin painotus oli esianalyysissa, jonka ensisijainen tarkoitus on todeta pullonkaulan olemassaolo ja vähentää mahdollisten tulevien, yksityiskohtaisempien tutkimusten laajuutta.

Työ esittelee 13 suorituskykymittaria ja antaa suosituksia niistä tehtäville tulkinnoille. minkä lisäksi mittareille ehdotetaan raja-arvoja, joiden ylittyminen on merkki resurssipullonkaulasta. Opinnäytetyössä esitellään myös ohjelma nimeltä Performance Analysis of Logs, jonka tarkoitus on helpottaa mittareiden lukemista. Neljä testiajoa suoritettiin virtuaalisella tietokantapalvelimella sen suorittaessa raskaita tietokantaoperaatioita, Windows Performance Monitorin kerätessä suorituskykytietoa käyttäen työssä esiteltyjä mittareita.

Monet valituista suorituskykymittareista osoittautuivat ongelmallisiksi virtuaalipalvelimilla, johtuen näiden piilotetusta laitteistokokoonpanosta. Tästä huolimatta testattavana olevalta palvelimelta onnistuttiin löytämään resurssipullonkaula, mikä osoittaa Windows Performance Monitorin olevan ainakin jossakin määrin käyttökelpoinen myös virtuaaliympäristöissä.

Avainsanat	Windows, suorituskykymittarit, tietokannat

Metropolia

Abstract

Author(s) Title Number of Pages Date	Olli Eklund Analyzing database server performance using Windows performance monitor 35 pages 14 February 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Programming
Instructor(s)	Olli Hämäläinen, Senior Lecturer

The objective of this thesis was to determine a limited set of performance counters for Windows Performance Monitor for detecting hardware resource bottlenecks on an Oracle database server during resource intensive database operations. The focus of the project was on preliminary analysis, with the primary purpose of establishing the existence of a bottleneck and limiting the scope of any following, more detailed investigation.

The thesis introduces thirteen performance counters and gives recommendations on interpreting them, in addition to suggesting threshold values that should be considered indicative of a bottleneck. A program called Performance Analysis of Logs is introduced facilitate the reading of counter data. Four test runs were performed on a virtual test server during intensive writing operations.

Several of the selected performance counters were found to be problematic on virtual servers due to a hidden hardware complement behind the virtualized hardware elements. Despite this, a resource bottleneck was found on the test server, showing at least moderate use for Windows Performance Monitor even in virtualized environments.

Keywords	Windows, Database Performance

Table of Contents

1 Introduction

Monitoring the performance of a database server can be challenging, requiring many hours of work to detect possible bottlenecks and establish the underlying root causes behind them. Ineffective utilization of database resources, faulty or insufficient hardware, bad architectural solutions and a misconfigured database management system are just some of the possible reasons behind degrading performance.

The purpose of this project was to determine suitable performance monitors for detecting hardware resource bottlenecks during resource intensive database operations in an Oracle database server functioning in a Windows Server environment, by using a program called Windows Performance Monitor. The program offers hundreds of different performance counters for monitoring server performance, of varying usefulness and applicability. The main focus of this work is to limit this set to approximately ten useful counters and give instructions on interpreting them, along with recommendations for alarming counter value ranges. By monitoring these performance counters during a test run, the server manager can quickly establish which hardware resources are forming a possible bottleneck. With this information, a more focused investigation on the root causes behind the performance problem can be performed.

This project was done for Process Vision Oy. Process Vision Oy is a Finnish IT company that provides information systems, applications and simulators for energy business companies. Currently the company has about 150 employees and has offices in Helsinki, Kuopio, Jyväskylä, Stockholm and Bussum. Process Vision Oy was founded in 1993.

2 Computer Performance

As a general definition a computer’s performance is the measure of its ability to perform the tasks allocated to it. Performance is often thought only as a function of speed, but also the resources used to achieve this speed should be considered. For example, a powerful computer system performing a relatively simple task quickly does not necessarily speak of good performance.

2.1 Effects of Performance

From the point of view of a software developer, performance is really only noticed when it is found lacking. In a business environment better than expected performance is often not given that much notice – what is more important is meeting the performance requirements set for the system by the whole process. If one part of the process fails to meet its performance requirements, this can lead to considerably larger delays, or in the worst case to cascading failures, further in the process chain. Performance exceeding these requirements, on the other hand, can rarely be taken full advantage of due to other parts of the process not being prepared for it.

2.2 Performance in Process Vision Oy

In Process Vision Oy a normal server setup is to have separate application and database servers. The application server contains the GENERIS application that is the main product of the company. The sole purpose of the database server is to run an Oracle database that contains the data utilized by the application server, for example time series data, billing information and general master data.

Most of the performance problems encountered at the company are related to database operations. Sometimes the database is utilized in a suboptimal way by the application, for example performing many small database operations when fewer larger ones should be called for. Or the database tables being utilized might be incompletely defined, lacking necessary indexing for heavily used key columns. However, these problems can be identified relatively easily, as they are not dependent on the hardware running the server.

Some performance problems only arise on specific servers. A process might run easily on a company test server, but major performance problems are encountered at the customer environment. As the database structure and the application are usually identical to company tests for the same product version, the problem can usually be found either in the Oracle database configuration or server hardware.

2.3 Performance Bottlenecks

A performance bottleneck occurs when a component in the system is under a load exceeding its capacity. This prevents other resources from working at their full capacity, or in some cases might cause an ineffective load shift to another component being less efficient at performing the required task.

Generally a bottleneck can be found in one of the four major hardware resource areas of the system: processor, memory, hard disk or network. Network configurations and bottlenecks are not in the scope of this project, so the first three resource areas will be concentrated on.

2.4 Monitoring Oracle Server Performance

Oracle has published several extensive documents for measuring and improving Oracle database server performance, for example the Oracle Database Performance Tuning Guide [1]. These documents offer an experienced database administrator several ways to measure Oracle performance and detect bottlenecks, giving methods and introducing tools even for a very detailed analysis. However, most of these tools require an additional license, bringing additional costs to the operation of every server. Many of them are also dependent on the Oracle version.

Another way of monitoring Oracle server performance is to monitor the performance of the hardware on the server. This does not give accurate information on the functions of the database software, but can be used to spot the hardware component that acts as a bottleneck. Measuring different hardware performance counters can also give some preliminary analysis on the root causes behind the bottleneck.

3 Tools Used for Monitoring Performance

In this chapter the tools used in this project are introduced. Windows Performance Monitor was used for collecting performance data using different, user-selectable performance counters, and creating log files of this data. The Performance Analysis of Logs tool was used to create reports from these logs, as the logs themselves are not in an optimal format for data interpretation.

3.1 Windows Performance Monitor

Although there is a multitude of tools available for performance monitoring, with different specializations and required degrees of user competence, finding a general tool that would both be widely available for all the required operating system configurations and would not require additional investment from the company did not leave many viable alternatives. Windows Performance Monitor has been the general tool in Process Vision Oy for first time performance analysis. It is a program included in Windows Server 2003 and 2008 installations. The application displays real time information about the use of hardware and software resources on the server. This information can be written to a log file for later inspection. These log files can then be read for example with Microsoft Office Excel, or the PAL tool that will be introduced in chapter 3.2. [2.]

Performance data can also be monitored in real time, but outside of a quick sanity check for specific state counters, such as the amount of free physical memory, this option is of limited use.

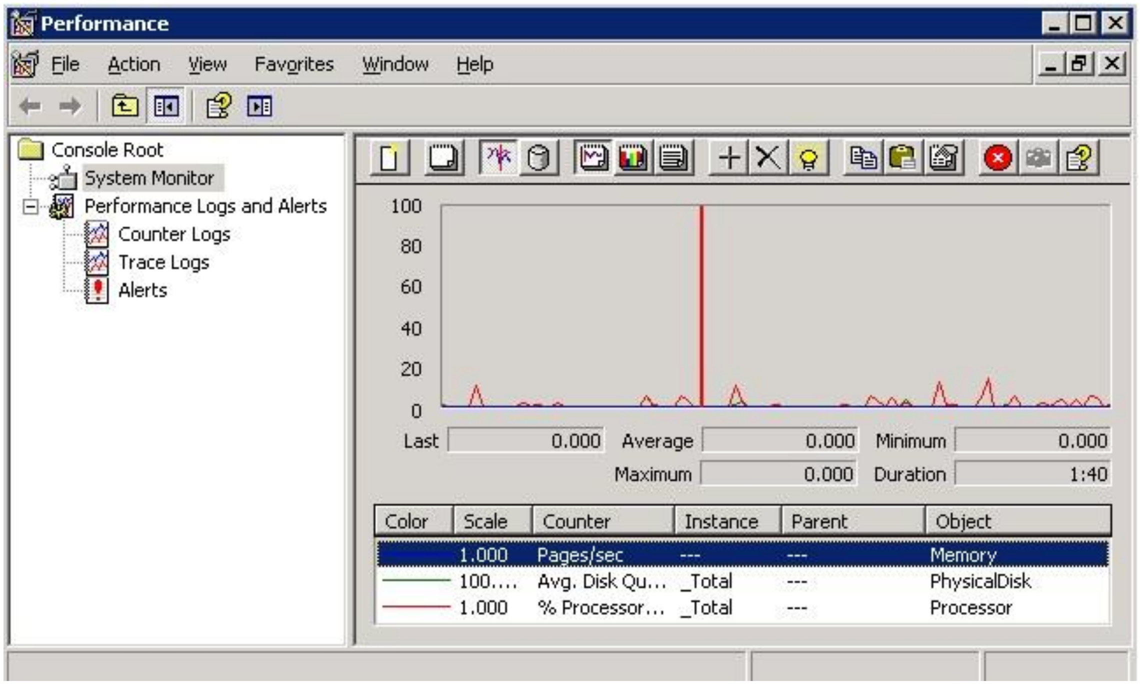
Windows Performance Monitor was chosen for this project for the following reasons:

Free. The Performance Monitor is automatically included in all Windows Server 2003 and 2008 installations, thus requiring no additional investment on part of the company.

Common. The program is an established tool and widely used by many professionals in the Windows environment. As competence in the use of the program can be expected from customers and partners, no additional training is required. Information about suitable performance counters and their threshold values in different situations can be exchanged quite easily, along with log data gathered from test runs. Also, when everybody is using the same program, possible problems arising from different ways of calculating values for these counters can be avoided.

Ease of use. While the program has a multitude of different performance counters available, predefined counter sets can be defined and saved in the form of an .htm file. Sharing this file enables users with little to no experience with the program, or performance monitoring in general, to start a performance monitoring session with a counter set suitable for the task.

Windows Performance Monitor allows the user to select different counters for monitoring server performance. A performance counter measures specific system activity or state, for example the amount of available physical memory or rate of hardware interrupts per second. The version included in Windows Server 2008 also allows the creation of data collector sets, which facilitates the exchange of performance counters sets between stakeholders. A screenshot of the Windows Performance Monitor is shown in figure 1.



Last 0.000 Average 0.000 Minimum 0.000
Maximum 0.000 Duration 1:40

Color	Scale	Counter	Instance	Parent	Object
	1.000	Pages/sec	---	---	Memory
	100....	Avg. Disk Qu...	_Total	---	PhysicalDisk
	1.000	% Processor...	_Total	---	Processor

Figure 1: Screenshot of the Windows Performance Monitor [2]

When the counter set is created, its *sampling interval* has to be defined. This interval defines the frequency of samples taken from the server. For example with a 15 second sampling interval, a sample will be taken from each counter target every 15 seconds. Counters defined as average values will show an average value during this interval. Other counters will show the value at the specific instant that the sample is taken and with a long interval they might give unreliable results.

The measurement interval should be set according to the length of the test run. For a short test run, a very long interval will not give accurate results. On the other hand, a needlessly short interval will give a lot of data but also lead to a huge log size, which in turn will be harder to analyze and might also consume a significant amount of space on the hard drive. Also, if the program is run on the same system than the actual test run, a short sampling interval might have an impact on system performance and thus the monitoring results.

In the Windows Performance Monitor documentation the values shown in table 1 are suggested for the measurement interval, depending on the length of the test run [2].

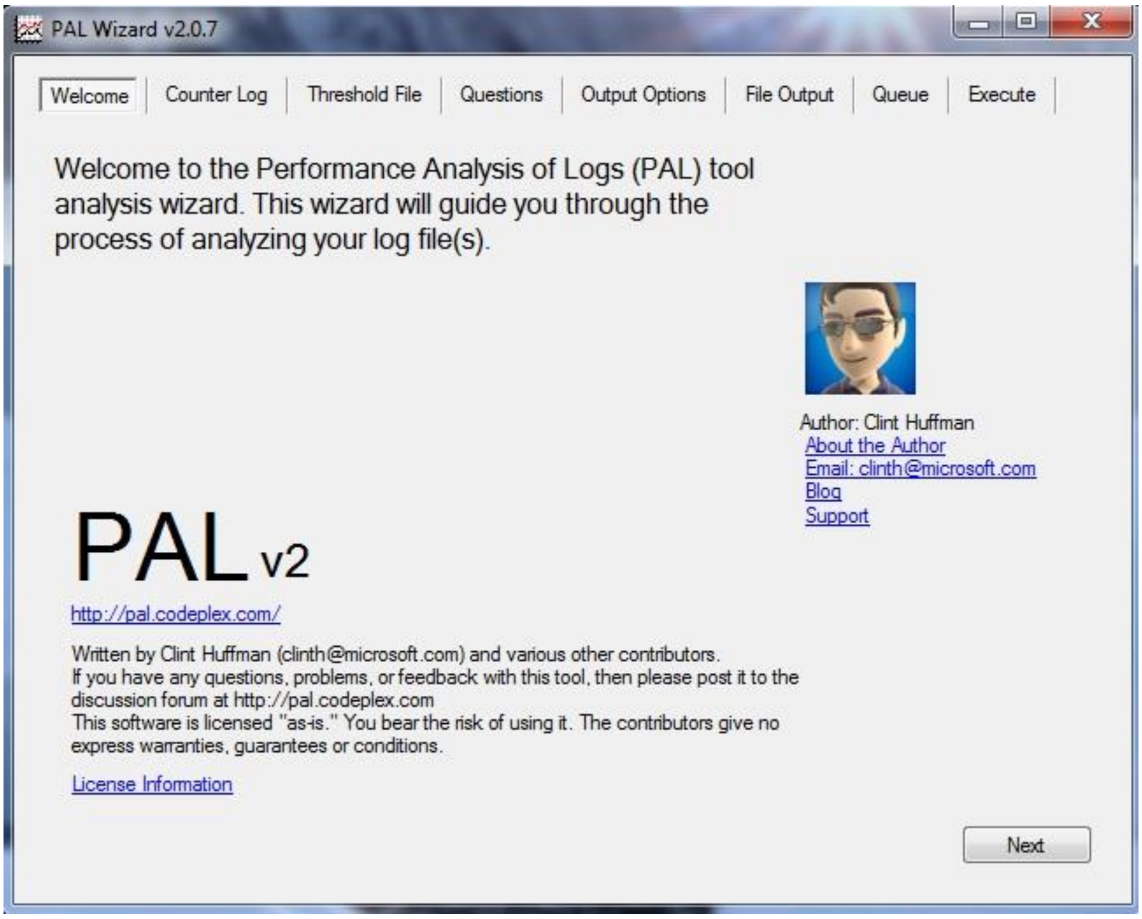
Table 1: Measurement intervals

Length of the test run	Measurement interval
< 1 hour	5 seconds
1-8 hours	15 seconds
8-24 hours	30 seconds
> 25 hours	60 seconds

A typical test run for the purposes of this project will lasted 2-8 hours, so a setting of 15 seconds for the sampling interval is recommended.

3.2 Performance Analysis of Logs Tool

The log files written by Windows Performance Monitor will, after a several hours' test run, contain a considerable amount of data. Reading and calculating this data manually is not very effective. For the purpose of facilitating the reading of Performance Monitor logs, Clint Huffman has written a program called Performance Analysis of Logs, often referred to as the PAL tool [3;4]. The main page of this program is shown in figure 2.



Welcome to the Performance Analysis of Logs (PAL) tool analysis wizard. This wizard will guide you through the process of analyzing your log file(s).

Author: Clint Huffman
About the Author
Email: clinth@microsoft.com
Blog
Support

http://pal.codeplex.com/

Written by Clint Huffman (clinth@microsoft.com) and various other contributors.
If you have any questions, problems, or feedback with this tool, then please post it to the discussion forum at http://pal.codeplex.com
This software is licensed "as-is." You bear the risk of using it. The contributors give no express warranties, guarantees or conditions.

License Information

Figure 2: Screenshot of the main page of the PAL tool [4].

This program allows the user to define a set of performance counters to be used with Performance Monitor, along with threshold values for each counter. These threshold values are set in the PAL tool and define a value threshold for each counter that should be considered indicative of a possible problem by the log analysis tool. The tool also has several built-in measurement templates for different server types, containing preset performance counters and threshold values considered suitable for the system in question. Unfortunately, these templates have been created for day-to-day monitoring and are not suitable for the purposes of this project. However, a custom template can be created, which will make setting up performance monitoring for different servers a lot easier, especially for systems running on Windows Server 2003, which lacks the Data Collector Set option included in later versions of the operating system.

The PAL tool will read the Performance Monitor log and create a report in HTML form, applying the established counter value thresholds to determine if bottlenecks exist. The tool shows minimum and maximum values encountered during the whole test run along with average values, also showing averages with 10%, 20% and 30% of the outliers removed.

4 Main Hardware Resource Areas

A computer system has three main hardware resource areas: processors, memory and hard disk drives. Any of these resource areas can form a bottleneck either because it is insufficient for the task at hand, or it is used in a suboptimal way.

In this chapter a cursory explanation will be given of the most important concepts related to performance monitoring for each major resource area.

4.1 Memory

Memory refers to the physical RAM and virtual memory of the computer system. These are used as a working space for applications.

4.1.1 Physical Memory

Physical memory means the installed Random Access Memory (RAM) on the server. RAM is the short term data storage used by applications and the operating system to handle information. Low physical memory translates to more paging, which can be noticed as increased I/O activity on the hard disk drive containing the page file.

The basic measure of memory performance is the amount of available physical memory on the server. The operating system functions and applications all need RAM to function, the paging file is only used as a temporary warehouse for data not currently used and cannot be relied on to account for serious lack of RAM. Insufficient memory will lead to a lot of paging, as data needs to be constantly exchanged between the paging file and RAM to accommodate the needs of different processes.

4.1.2 Virtual Memory

Virtual memory consists of the physical memory and a predefined space on a hard disk called the paging file. This paging file is used as an extension of physical memory by the operating system when needed, by moving data from RAM to the hard disk when additional RAM is required. This is called *paging*. The size limit of the paging file can be set in the operating system parameters. The maximum amount of virtual memory is called the *commit limit*. This commit limit is limited only by the amount of physical RAM and the maximum size of the paging file. Applications, and even many of the operating system functions, do not differentiate between physical memory and the paging file, for them the only visible memory is the virtual memory.

4.1.3 Committed Virtual Memory

Virtual memory that has been reserved to a process is considered committed and may not be used by other processes. Thus available virtual memory is the difference between the amount of committed virtual memory and the commit limit. If the amount of committed virtual memory reaches the commit limit, no new processes may be created.

4.1.4 Application and Kernel Memory

Virtual memory is divided between application memory and kernel memory. Kernel memory is memory reserved for operating system functions, while application memory is used by applications.

4.1.5 Paging

Paging is the act of the operating system swapping data between physical memory and the paging file. A typical situation for this is when the required information is not found in the physical RAM and has to be loaded from the page file. Microsoft calls these events as *hard page faults*. [2.] Although these page faults are a normal event on a system utilizing virtual memory, excessive amount of them can be problematic. As paging utilizes the hard disk drive, a high paging count can often be seen as elevated hard disk I/O activity. Insufficient physical RAM is one of the possible causes for excessive paging, as the page file is constantly needed to compensate for the limited memory.

Memory leaks can also cause a lot of paging, as when a process keeps expanding its memory space, other processes' committed memory is constantly being pushed out of the physical RAM and into the page file. [5.]

4.1.6 Memory Leaks

Memory leak is a situation in which an application, after using a part of memory allocated to it, fails to release this part of the memory for further use, keeping this unused part reserved for itself. This will lead to a continuously rising memory allocation for the application in question. [5.]

4.2 Processor

The Central Processing Unit (CPU) is the core of the computer that performs all the calculations required by applications and operating system components. Nowadays most servers have several CPUs, which should be taken into account when measuring overall processing performance.

4.2.1 Processor Time Allocation

When measuring processor performance, an essential consideration is how a processor's execution time is being spent. A processor's time is roughly divided between operating system functions, application processes and the idle process.

Spending processor time on operating system and low level hardware functions is called working in *privileged space* or *privileged mode*, while the time spent in working for user applications is spent in *user space* or *user mode*. The Oracle database application can generally be considered to run in user space. [1, 5-5.]

The idle process is being processed when the CPU has no other functions to perform. For a processor it is the functional equivalent of doing nothing. Thus, a processor spending none of its processing time in the idle process would be considered fully utilized.

4.2.2 Hardware Interrupts and Deferred Procedure Calls

Hard disk drives, network interface controllers, peripheral devices and many other hardware components generate interrupts when they require a processor's time. Normal process execution is interrupted for the duration of the interrupt and will be continued after the service required by the device has been performed. Also most system clocks generate an interrupt every 10 milliseconds. Deferred Procedure Calls (DPCs) are lower priority interrupts that are not counted by performance counters measuring normal interrupts. Both are executed in privileged space by the processor. [2.]

Device interrupts and DPC:s are a normal function of computer systems, but if a large fraction of a CPU's execution time is spent servicing interrupts, this could indicate a problem with a hardware device. The device might be faulty, or it might have malfunctioning or outdated driver software.

4.2.3 Context Switches

A context switch occurs, when a processor switches from one thread to another, or from privileged space to user space [2]. Handling of context switches is an operating system function and thus executed in privileged space.

The rate context switching depends on the amount of threads being run, but also on the relative priorities of these threads. A high priority program might lower the amount of context switches by monopolizing the processor for extended amounts of time. [6]

4.3 Hard Disk

4.3.1 Spindle

A spindle is jargon for an actual physical hard disk drive, the hardware device itself [7]. The word is used in this thesis to differentiate it from the physical disk definition from the operating system perspective, which in reality could contain severals spindles.

Depending on the device in question, a spindle has a specific amount of space and expected performance characteristics.

4.3.2 Line Unit Number and Physical Hard Disk

A Line Unit Number (LUN) identifies a specific logical hard disk unit, which can consist of a part of a physical hard disk spindle, the whole spindle or several spindles treated as one hard drive unit. For the Windows operating system, a single LUN is seen as a single physical hard drive device and it is also measured as such [8;9]. If the amount of actual physical spindles behind a physical hard disk unit is unknown, no realiable performance data on the actual hardware device performance can be received by monitoring this unit.

4.3.3 Logical Hard Disk

A logical hard disk definition for Windows is essentially a partition, a part of one specific physical hard disk unit that has been defined as a single logical unit by the operating system. A logical hard disk unit has a predefined space allocation and cannot exceed this, even if the physical disk space would allow this.

5 Performance Counters

Windows Performance Monitor offers hundreds of different performance counters. One of the challenges in this project was to choose the most useful ones that could act as reliable indicators, either alone or in combination with each other, and could be used in a cursory analysis to define if there is an existing bottleneck. Many counters that would have at first seemed useful have been discounted due to unreliability or being useful in only very specific cases that would not fit into the scope of a first-time bottleneck analysis.

This chapter will introduce the performance counters which have been selected for the purposes of this project. Threshold values are given for each counter that should indicate a problem with the hardware component in question.

Most of the suggestions found for choosing and interpreting Performance Monitor counters presumed the monitoring to happen during regular server operations. As the focus of this work is to evaluate system performance only during resource intensive repetitive database operations, many of the recommendations found for counter interpretation and their threshold values could not be taken at face value. In these cases I have given my own recommendations.

5.1 Performance Counters Related to Processor Performance

5.1.1 Percentage of Processor Time

The *Processor: % Processor time* counter indicates the percentage of time the processor has been busy during the measuring interval. It is the general indicator on how much strain is put on the CPU in question.

Values ranging from 70% to 85% have been suggested to indicate an overworked processor [10;24;11]. As this work focuses on very resource intensive operations instead of routine server monitoring, it would be tempting to set the alarm threshold as high as 95%. However, as I have never encountered a genuine processor bottleneck in my time at Process Vision Oy when running this kind of operations, even a processor usage of 70% could be considered a sign of something else using the processor time and thus worthy of investigation. After checking other relevant performance counters for possible root cause indicators, individual processes could then be monitored to establish that it is indeed the database operation being tested using most of the processor time.

5.1.2 Percentage of Interrupt Time

The *Processor: % Interrupt Time* counter shows the percentage of processor time that is used for servicing hardware interrupts. Hard disk drives, network interface controllers, peripheral devices and many other essential hardware components generate these interrupts. Also most system clocks generate an interrupt every 10 milliseconds. Normal process execution is interrupted for the duration of an interrupt. [2.]

A high percentage of time spent for servicing interrupts for a CPU could indicate a problem with a hardware device. The device might be faulty, or it might have a malfunctioning or outdated driver software.

A limit of 15-20% interrupt time or higher is recommended as an indicator of a possible hardware problem [11;12].

5.1.3 Percentage of Privileged Time

The *Processor: % Privileged Time* counter shows the percentage of processor time spent in privileged mode.

If this counter value is higher than 30%, it indicates that the processor is spending an excessive amount of time performing system functions such as I/O and thread context switching [12;13].

If also either the Processor: % Interrupt Time performance counter or the Processor: % DPC time counter exhibit abnormally high values, a faulty hardware component or device driver is a likely root cause and a deeper analysis is recommended [14].

Otherwise there might be a problem with excessive context switching, which would be indicated by the System: Context Switches / Sec performance counter [15].

5.1.4 Context Switches

The value on the *System: Context Switches/sec.* counter shows the sum of context switches performed on all processors on the server per second [2]. It is thus a good indicator of the level of multitasking performed by the system. A context switch is also recorded when a processor running a thread changes its state between user and privileged modes [15].

Sources differ on the amount of context switches that should be considered excessive. Guy Thomas in his book Performance Monitor for Windows suggests that 3,000 context switches per second would be perfectly normal, recommending investigation only when the amount exceeds 10,000 switches per second. [10;27.] Clint Huffman in his Microsoft TechNet blog post regarding his experiences on excessive context switching would consider 1,000 context switches a normal value, but no clear limit for an alarming situation is given [15].

In my opinion, even a very high amount of context switches per second should not be automatically considered a problem, as powerful systems with many processors are able to run very large amounts of threads without significant problems. Based on this it is recommendable that this counter is only used as a problem indicator if there is supporting evidence available from other relevant counters. When testing on a virtual server, more than 6,000 context switches per second have been encountered in several hours' time, with peak values on a 2 minute PAL tool monitoring interval going as high as 15,000 context switches, with no other processor related problem indicators being evident. Results of these tests are presented in chapter 6.

However, if a high usage percentage of CPU privileged mode is evident in a Processor: % Privileged Time counter, this would indicate that the high amount of context switching is putting a strain on the processor, which in turn would indicate an excessive amount of threads being utilized.

5.1.5 Percentage of Deferred Procedure Call Time

The *Processor: % DPC Time* counter shows the percentage of processor time spent handling Deferred Procedure Calls (DPC). DPCs are lower priority interrupts that are not counted by performance counters measuring normal interrupts. They are executed in privileged mode by the processor. [2.]

A value of 20% or more should be considered as indicative of possible hardware or driver software problems and further analysis is recommended [14].

5.2 Performance Counters Related to Memory Performance

5.2.1 Available Memory

The *Memory: Available Mbytes* counter is the most basic performance counter for memory, showing the amount of free memory in megabytes (MB). The percentage of available memory can be calculated with the following formula

$$\frac{mem\_available}{mem\_total} \cdot 100$$

where *mem_total* is the total amount of RAM on the server in megabytes and *mem_available* the value in the Memory: Available Mbytes counter.

Having 10% or more of total memory available is considered to be acceptable. [16;17.]

5.2.2 Percentage of Committed Memory in Use

The percentage of committed memory in use is shown in the *Memory: % Committed Bytes In Use* counter. It is calculated with the following formula

$$\frac{committed\_memory}{commit\_lm} \cdot 100$$

where *committed_memory* is the total committed memory and *commit_lm* the commit limit.

High value on this counter would be indicative of low physical memory or too small a page file [16].

5.2.3 Pages Per Second

According to the Performance Monitor, the *Memory|Pages Per Second* counter indicates how many pages are read or written to page file every second to resolve hard page faults [2]. A hard page fault is defined as a situation where the required information is not found in the physical RAM and has to be loaded from the page file.

Insufficient physical RAM is one of the possible causes for excessive paging, as the page file is constantly needed to compensate for the limited memory. To confirm this, the Memory: Available Bytes counter should be checked. An insufficiently sized page file can also cause a high amount of pages per second [2].

A high paging count can also be caused by a memory leak. Memory leak is a situation where an application, after using a part of memory allocated to it, fails to release this part of the memory for further use, keeping this unused part reserved for itself. This will lead to a continuously rising memory allocation for the application in question, which causes paging activity even if the amount of physical RAM is not an issue. [2.]

According to Clint Huffman in the Microsoft TechNet blog, a high page file count can also be caused by applications using memory mapped files [18]. However, my research was unable to indicate that the Oracle database software would utilize this approach in an excessive manner, so I would consider it an unlikely source of a high paging count on a dedicated database server.

Suggestions for a pages per second limit that should be considered indicative of a problem vary considerably depending on the source. The Performance Monitor application help suggests a threshold as low as 20 pages per second [2], but I would speculate that this information is outdated. CC Hameed suggests in the blog of EPS Windows Server Performance Team values between 40-300 pages per second, depending on the speed of the page file hard drive [2]. Steven Choy suggests a limit of 1000 pages per second in the Microsoft TechNet Magazine to indicate a memory leak [11]. The same limit is suggested by Microsoft in their Exchange Server support blog [19]. As with almost all counters, this limit is dependent on the quality of the server hardware. Because of this, it is recommendable to consider only values exceeding 1000 pages per second as indicative of possible memory problems.

5.2.4 Page File Usage Percentage

The *Page File: % Usage* counter shows the usage percentage of the selected page file. If the system has several page files, a separate counter can be created for each of them.

This counter can be used to determine very easily if the page file is indeed too small in a situation where other performance counters would suggest this as a possible root cause.

The Performance Monitor documentation suggests an alert limit for this counter to be set at 70% [2]. The same view is shared by Guy Thomas in his e-book Performance Monitor for Windows [10].

5.3 Performance Counters Related to Hard Disk Performance

5.3.1 Physical Disk Idle Time Percentage

The *PhysicalDisk: % Idle Time* counter shows the amount of time in percentage the physical disk in question is spending idle. A physical disk having no time is working on something all the time and quite possibly is taxed beyond its capacity.

I would recommend using this counter as the first indicator of a possible hard disk problem in a case where there is only one spindle behind the physical disk unit. I have been unable to find reliable information concerning this performance counter's reliability in situations where there are several spindles. In these situations I would recommend looking at disk response times, presented by performance counters PhysicalDisk: Avg. Disk Sec/Write and PhysicalDisk: Avg. Disk Sec/Read.

In the Microsoft TechNet Magazine Steven Choy suggests that a counter value below 20% would indicate a saturated hard drive [11] and should be considered indicative of a problem. However, as the purpose of this work is to give suggestions for resource instensive database operations, I would consider anything above zero as acceptable, especially if other performance counters do not show alarming values.

5.3.2 Logical Disk Free Space Percentage

The *Logical Disk: % Free Space* counter indicates the percentage of free space on the partition being monitored. A very low amount of this on any partition in use is an obvious problem, as it would prevent further saving of data by processes utilizing it, even if the physical disk in question would have the required space available.

Other than for obvious reasons for monitoring partition space, this counter is essential for keeping an eye on the operating system partition. It is recommended that the counter value for this partition is not allowed to fall below 15%. [11]

5.3.3 Physical Disk Average Queue Length

The *PhysicalDisk: Avg. Disk Queue Length* performance counter shows the estimate of requests on the physical disk that are either in service or waiting for service [20]. A suggested alert limit for this value would be a queue length of 2 per physical disc spindle [11;20]. It is also noted that this value should be continuous, meaning an alarming value in a single interval should not be considered indicative of a problem. [20.]

5.3.4 Physical Disk Response Times

The *PhysicalDisk: Avg. Disk Sec/Write* and *PhysicalDisk: Avg. Disk Sec/Read* counters measure the response times of a physical hard disk unit in write and read operations, respectively. The value shown is the amount of seconds it takes to perform a write or read operation.

Unlike other hard disk performance counters, the threshold values of this counter are not dependent on the amount of spindles behind the physical disk unit. A slow response time indicates that the spindles trying to process an operation are too busy to respond immediately. [21.] Thus I would recommend giving a high priority to this counter, especially in situations where the amount of spindles behind the physical disk units is unknown, for example when using a virtual server.

Response times naturally differ depending on the hard disk model in use. Being aware of the expected response times of the model in question can help interpreting the values of this counter. Generally most modern hard disks can be expected to have response times below 10 ms and values consistently going much above this should be considered indicative of a problem [11;21].

6 Testing the Performance Monitor

Tests were run on a company test server, in a database domain dedicated to performance testing.

Server attributes:

Operating system	Windows Server 2003
CPU:s	8
RAM	12 GB
Database	Oracle v10g

The server was running on a virtual platform. This means that no accurate data on the true hardware configuration behind the virtualized hardware elements was available. For example the amount of spindles behind a physical disk unit was unknown, which makes the PhysicalDisk: Avg. Disk Queue Length counter practically useless.

This is unfortunate, but at this point it had become clear that virtual servers are the standard for company test servers and no other viable alternatives were available. These tests would also give a more accurate picture on the usefulness of this project in an authentic situation.

6.1 Test Runs

Four test runs were performed. Each test run lasted for an hour, where Windows Performance Monitor was activated a few minutes before the beginning of the run and closed a few minutes before its conclusion. This was done to prevent idle periods from corrupting the test data.

The test runs were performed at different times of the day, to allow for possible daily regular operations on other servers or domains sharing the same resources from affecting the results. One of the test runs had to be later disqualified due to misconfiguration of monitoring parameters.

As suggested in chapter 3.1, a sampling interval of 15 seconds was chosen for the Performance Monitor.

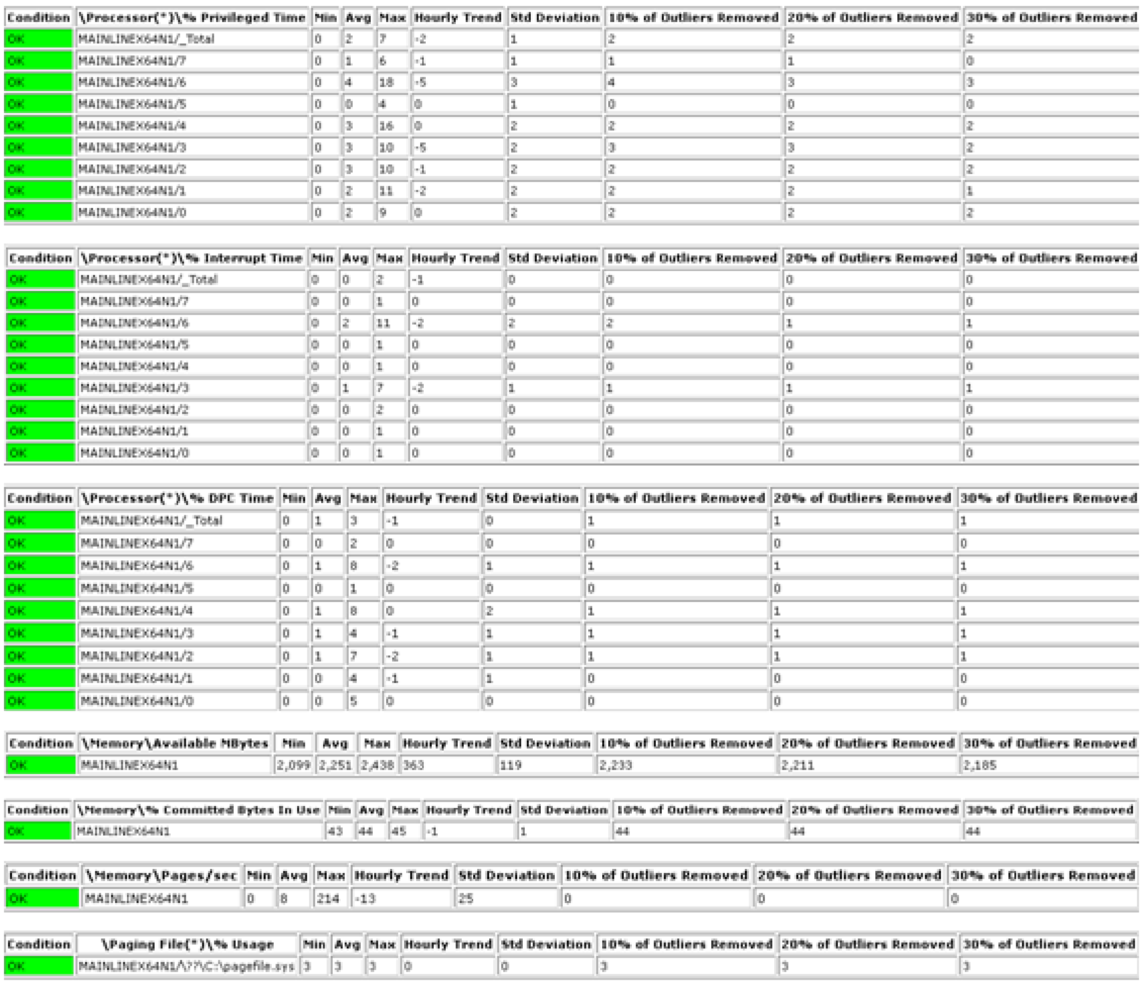
6.2 Handling of Test Data

The log files created by the Performance Monitor were read with the PAL tool to create reports that could be more easily analyzed. At the time of the tests no definite threshold values had yet been selected for different performance counters, so preselected thresholds from the PAL template for SQL Server were used, as this would be closest to an Oracle server in performance. All the threshold values configured in this template are lower or equal to those recommended in chapter 5 for the selected performance counters, which guarantees that readings that should be considered alarming within the parameters of this project will be highlighted.

6.3 Test results

These tests resulted in more than 4 MB of PAL tool reports in .htm files, comprising from over 30 MB of Windows Performance Monitor log data in .blg format. The extensive amount of test data makes it impractical to present it all in this report, especially as most of the reports had nothing significant in them. Thus in this chapter selected performance information will be presented using the counters introduced in chapter 5 and anomalies found in this data will be discussed.

The picture below shows the PAL tool conclusion reports for the performance counters Processor: % Interrupt Time, Processor: % Privileged Time, Processor: % DPC Time and all the memory related counters lister in chapter 5.



Condition	\Processor(*)\% Privileged Time	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1/_Total	0	2	7	-2	1	2	2	2
OK	MAINLINEX64N1/7	0	1	6	-1	1	1	1	0
OK	MAINLINEX64N1/6	0	4	18	-5	3	4	3	3
OK	MAINLINEX64N1/5	0	0	4	0	1	0	0	0
OK	MAINLINEX64N1/4	0	3	16	0	2	2	2	2
OK	MAINLINEX64N1/3	0	3	10	-5	2	3	3	2
OK	MAINLINEX64N1/2	0	3	10	-1	2	2	2	2
OK	MAINLINEX64N1/1	0	2	11	-2	2	2	2	1
OK	MAINLINEX64N1/0	0	2	9	0	2	2	2	2

Condition	\Processor(*)\% Interrupt Time	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1/_Total	0	0	2	-1	0	0	0	0
OK	MAINLINEX64N1/7	0	0	1	0	0	0	0	0
OK	MAINLINEX64N1/6	0	2	11	-2	2	2	1	1
OK	MAINLINEX64N1/5	0	0	1	0	0	0	0	0
OK	MAINLINEX64N1/4	0	0	1	0	0	0	0	0
OK	MAINLINEX64N1/3	0	1	7	-2	1	1	1	1
OK	MAINLINEX64N1/2	0	0	2	0	0	0	0	0
OK	MAINLINEX64N1/1	0	0	1	0	0	0	0	0
OK	MAINLINEX64N1/0	0	0	1	0	0	0	0	0

Condition	\Processor(*)\% DPC Time	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1/_Total	0	1	3	-1	0	1	1	1
OK	MAINLINEX64N1/7	0	0	2	0	0	0	0	0
OK	MAINLINEX64N1/6	0	1	8	-2	1	1	1	1
OK	MAINLINEX64N1/5	0	0	1	0	0	0	0	0
OK	MAINLINEX64N1/4	0	1	8	0	2	1	1	1
OK	MAINLINEX64N1/3	0	1	4	-1	1	1	1	1
OK	MAINLINEX64N1/2	0	1	7	-2	1	1	1	1
OK	MAINLINEX64N1/1	0	0	4	-1	1	0	0	0
OK	MAINLINEX64N1/0	0	0	5	0	0	0	0	0

Condition	\Memory\Available MBytes	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1	2,099	2,251	2,438	363	119	2,233	2,211	2,185

Condition	\Memory\% Committed Bytes In Use	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1	43	44	45	-1	1	44	44	44

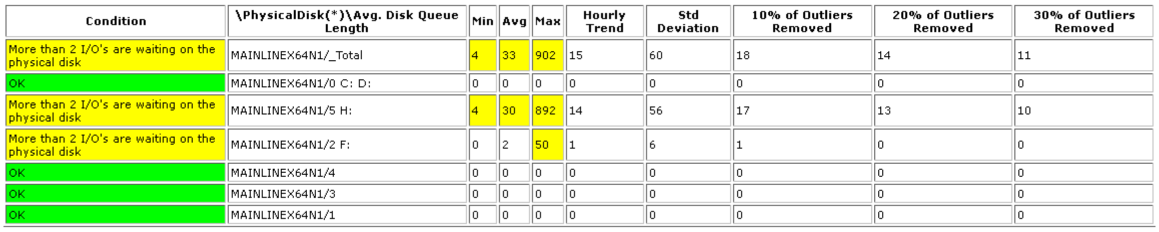
Condition	\Memory\Pages/sec	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1	0	8	214	-13	25	0	0	0

Condition	\Paging File(*)\% Usage	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1/\??\C:\pagefile.sys	3	3	3	0	0	3	3	3

Figure 3: Conclusion reports for performance counters with unremarkable values

As can be seen in figure 3, even the maximum values encountered for these performace counters are completely unremarkable. The processor works mostly in user space, Interrupt and DPC times give no reason to suspect hardware failures. There is plenty of RAM available at all times, and pages per second, even when spiking, do not come even close to the threshold value of 1,000 pages per second. The values for these counters were similar for all test runs and do not require any further analysis.

The picture below shows the PAL tool conclusion report for performance counter for PhysicalDisk: Avg. Disk Queue Length.

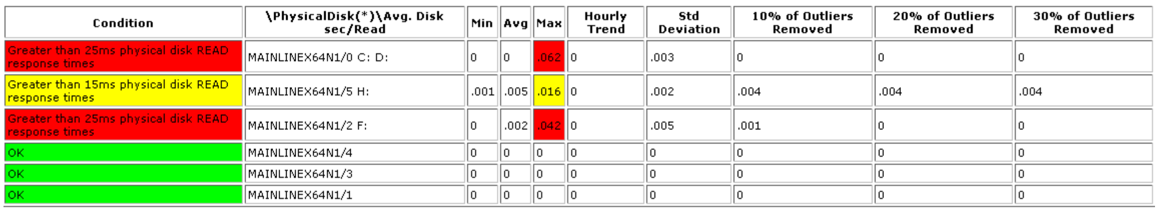


Condition	\PhysicalDisk(*)\Avg. Disk Queue Length	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
More than 2 I/O's are waiting on the physical disk	MAINLINEX64N1/_Total	4	33	902	15	60	18	14	11
OK	MAINLINEX64N1/0 C: D:	0	0	0	0	0	0	0	0
More than 2 I/O's are waiting on the physical disk	MAINLINEX64N1/5 H:	4	30	892	14	56	17	13	10
More than 2 I/O's are waiting on the physical disk	MAINLINEX64N1/2 F:	0	2	50	1	6	1	0	0
OK	MAINLINEX64N1/4	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/3	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/1	0	0	0	0	0	0	0	0

Figure 4: Conclusion report for the PhysicalDisk: Avg. Disk Queue Length counter

As can be seen from the average values for the physical disk unit employed by the database (MAINLINEX64N1/5 H:) in figure 4, either this disk unit is heavily overtaxed, average disk queue length being as high as 10 even when 30% of outliers have been removed, or then this particular disk unit is in reality being serviced by more than one spindle. As the server in question is a virtual server, the amount of spindles is unknown, which makes this performance counter practically useless in this occasion. However, performance counters that measure disk response times can still be used, as overworked physical hard disk components would show as increased response times, despite the amount of spindles behind a physical disk unit.

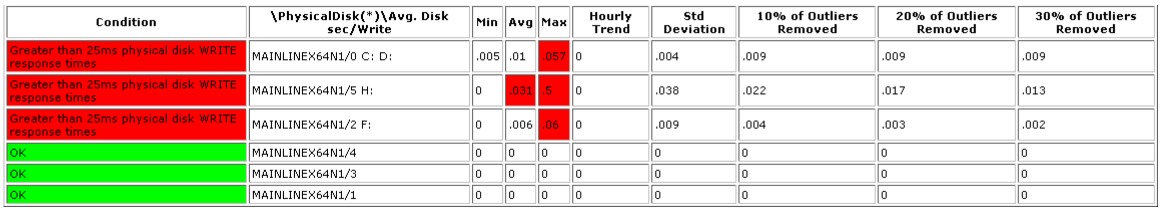
Below in figure 5 are shown the average disk response times for read operations during the test run in question.



Condition	\PhysicalDisk(*)\Avg. Disk sec/Read	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
Greater than 25ms physical disk READ response times	MAINLINEX64N1/0 C: D:	0	0	.062	0	.003	0	0	0
Greater than 15ms physical disk READ response times	MAINLINEX64N1/5 H:	.001	.005	.016	0	.002	.004	.004	.004
Greater than 25ms physical disk READ response times	MAINLINEX64N1/2 F:	0	.002	.042	0	.005	.001	0	0
OK	MAINLINEX64N1/4	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/3	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/1	0	0	0	0	0	0	0	0

Figure 5: Conclusion report for the PhysicalDisk: Avg. Disk Sec/Read counter

Figure 6 below shows the average disk response times for write operations during the test run in question.



Condition	\PhysicalDisk(*)\Avg. Disk sec/Write	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
Greater than 25ms physical disk WRITE response times	MAINLINEX64N1/0 C: D:	.005	.01	.057	0	.004	.009	.009	.009
Greater than 25ms physical disk WRITE response times	MAINLINEX64N1/5 H:	0	.031	.5	0	.038	.022	.017	.013
Greater than 25ms physical disk WRITE response times	MAINLINEX64N1/2 F:	0	.006	.06	0	.009	.004	.003	.002
OK	MAINLINEX64N1/4	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/3	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/1	0	0	0	0	0	0	0	0

Figure 6: Conclusion report for the PhysicalDisk: Avg. Disk Sec/Write counter

The test run in question is a writing operation, which can be seen in the read response times that on average are well below the threshold. The high maximum values are caused by occasional spikes that do not have a high impact on the overall performance.

The unmodified average value of 31 ms write response time should be a cause for alarm, as this exceeds the threshold of 25 ms by a considerable margin. Even by removing 10% of the outliers, which should eliminate any few extreme response time spikes possibly skewing the average, this value is as high as 22 ms. Only by eliminating 30% of the outliers can a less threatening value of 13 ms be achieved. This would indicate that the writing operation performed by the application is performed sequentially, short periods of low database activity being followed by periods of heavy writing. This can be confirmed by looking at the graph showing the write response times during the whole operation in figure 7. The unit for response times in the graph is one second.

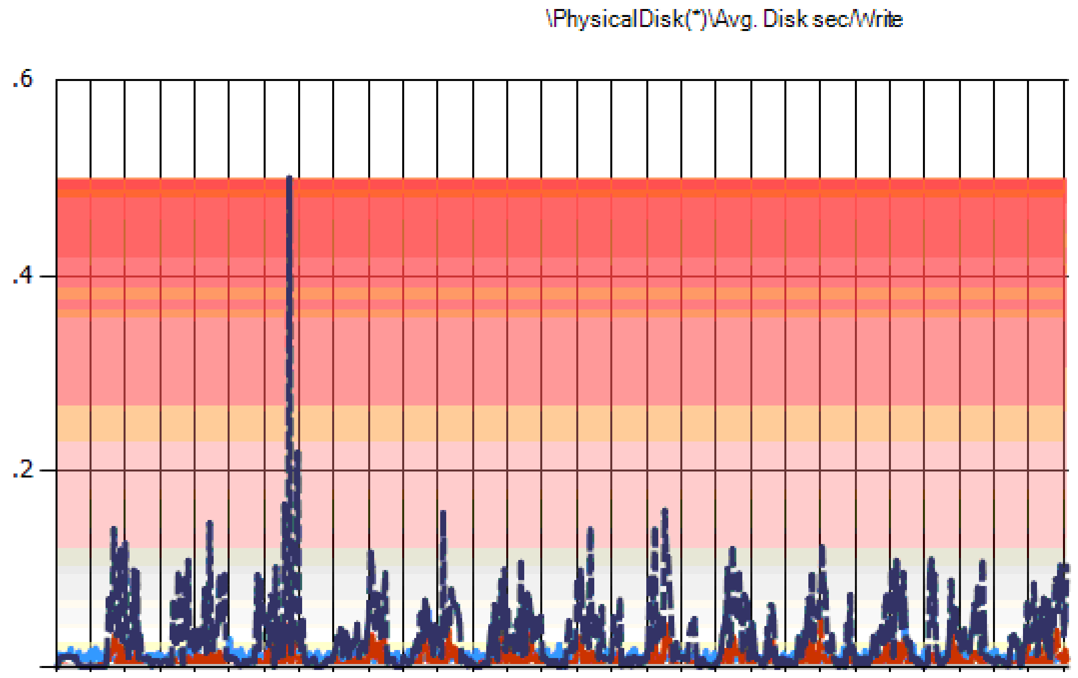
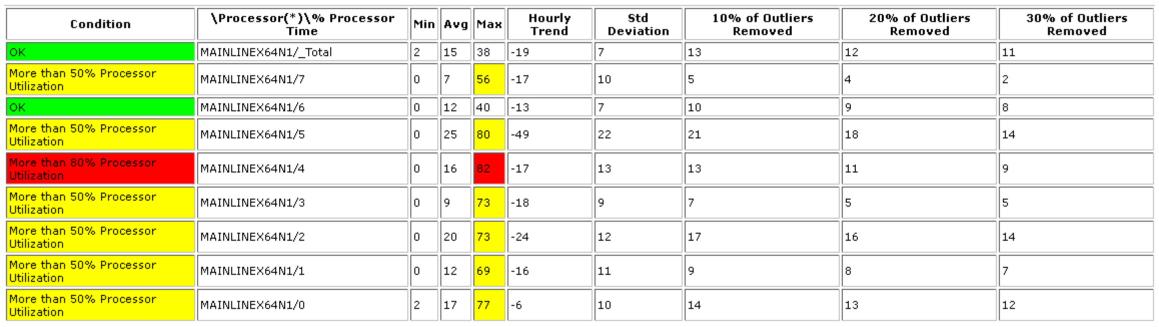


Figure 7: PhysicalDisk: Avg. Disk Sec/Write counter

On the basis of these response times I would conclude that the database server has insufficient hard disk resources allocated for this operation. There are sequential response lag spikes during the whole test run that clearly indicate a considerable response lag during periods of heavy writing. However, as the server in question is a virtual server, there is always the possiblity that the same resources are used by some other server situated on the same virtual platform.

The conclusion report for the performance counter for processor utilization, Processor: % Processor time, is shown below in figure 8.



Condition	\Processor(*)\% Processor Time	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
OK	MAINLINEX64N1/_Total	2	15	38	-19	7	13	12	11
More than 50% Processor Utilization	MAINLINEX64N1/7	0	7	56	-17	10	5	4	2
OK	MAINLINEX64N1/6	0	12	40	-13	7	10	9	8
More than 50% Processor Utilization	MAINLINEX64N1/5	0	25	80	-49	22	21	18	14
More than 80% Processor Utilization	MAINLINEX64N1/4	0	16	82	-17	13	13	11	9
More than 50% Processor Utilization	MAINLINEX64N1/3	0	9	73	-18	9	7	5	5
More than 50% Processor Utilization	MAINLINEX64N1/2	0	20	73	-24	12	17	16	14
More than 50% Processor Utilization	MAINLINEX64N1/1	0	12	69	-16	11	9	8	7
More than 50% Processor Utilization	MAINLINEX64N1/0	2	17	77	-6	10	14	13	12

Figure 8: Conclusion report for the Processor: % Processor Time counter

The average processor utilization is quite low, but sequential peaks of processor activity can be seen when high hard disk reponse times were evident, as can be seen in figure 9 below.

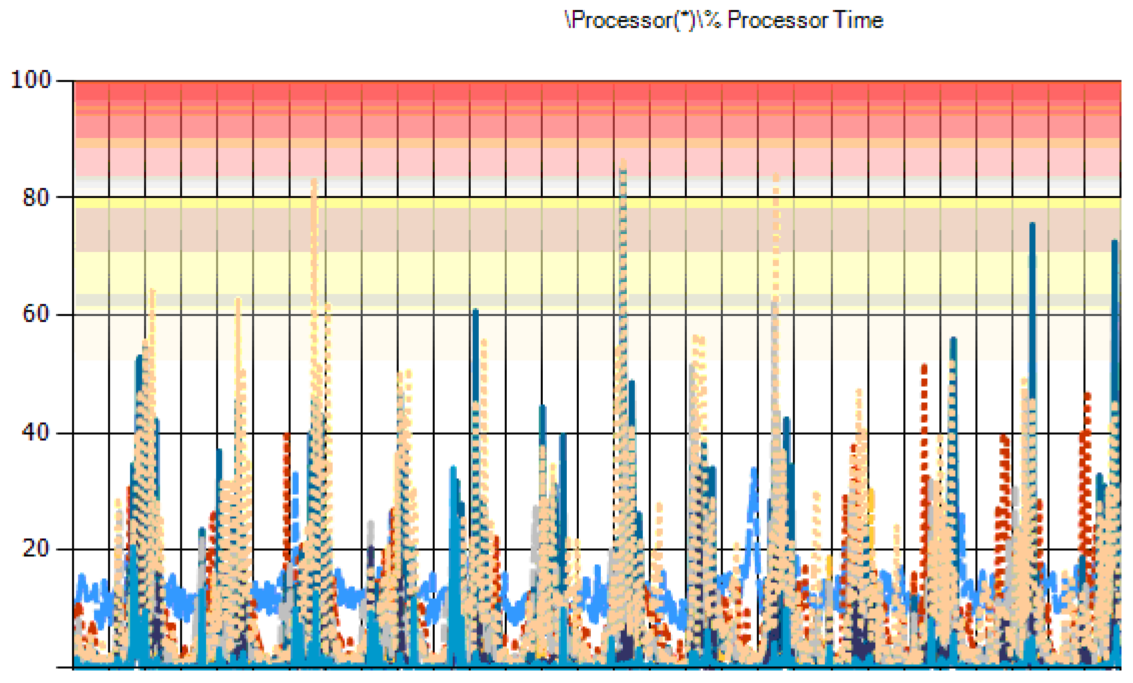
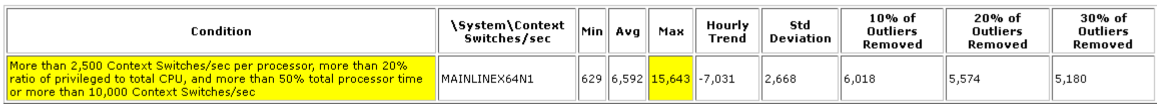


Figure 9: graph for the Processor: % Processor Time counter

By looking at the Processor: % Privileged Time counter, which is a component of Processor: % Processor time, it can be determined that the processors are mostly working in user space, also during these peaks. The timing of the peaks correlates loosely with the hard disk response time peaks shown in figure 7, which would indicate short but heavy writing periods by the application performing the test runs. The processor load during these peaks is relatively high, but this can only be expected during a heavy writing operation.

The results for the System: Context Switches/Sec counter can be seen in figure 10 below.

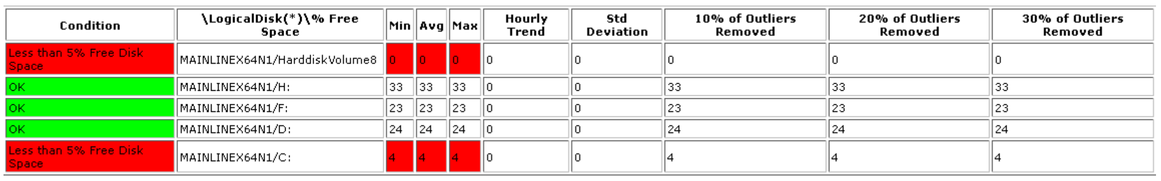


Condition	\System\Context Switches/sec	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
More than 2,500 Context Switches/sec per processor, more than 20% ratio of privileged to total CPU, and more than 50% total processor time or more than 10,000 Context Switches/sec	MAINLINEX64N1	629	6,592	15,643	-7,031	2,668	6,018	5,574	5,180

Figure 10: Conclusion report for the System: Context Switches/Sec counter

The values shown can be considered pretty high, even going above 10,000 context switches per second occasionally. However, as mentioned in chapter 5.1.4, I am unwilling to consider a high amount of context switching indicative of a real problem unless some related indicators, such as the Processor: % Privileged Time, are also showing high values. As could be seen earlier in this chapter (figure 3), this is not the case.

Perhaps the most confusing anomaly during these test runs can be seen in the Logical Disk: % Free Space performance counter, as shown in figure 11.



Condition	\LogicalDisk(*)\% Free Space	Min	Avg	Max	Hourly Trend	Std Deviation	10% of Outliers Removed	20% of Outliers Removed	30% of Outliers Removed
Less than 5% Free Disk Space	MAINLINEX64N1/HarddiskVolume8	0	0	0	0	0	0	0	0
OK	MAINLINEX64N1/H:	33	33	33	0	0	33	33	33
OK	MAINLINEX64N1/F:	23	23	23	0	0	23	23	23
OK	MAINLINEX64N1/D:	24	24	24	0	0	24	24	24
Less than 5% Free Disk Space	MAINLINEX64N1/C:	4	4	4	0	0	4	4	4

Figure 11: Conclusion report for the LogicalDisk: % Free Space counter

As can be seen in the picture above, the percentage free space on all logical disks on the server stays constant for the whole operation. As it was confirmed after the test, actual data was written in the database by the test run in question, and as the counter value even for the logical disk unit bearing the operating system stays constant for the whole test period, it can be concluded that this counter does not show reliable values on a virtual server. Also, all three successful test runs had this same anomaly and attempts to reproduce it on a physical workstation have been unsuccessful.

6.4 Interpreting the Results

From this test data it can be concluded that the traffic coming from the application server to the database server is very bursty, meaning most of the data delivery is concentrated on peaks of heavy writing operations with long periods of relative inactivity in between. The hard disk performance during these peaks is clearly a bottleneck, with response times exceeding the 25 ms threshold by a considerable margin in all three successful test runs. It should also be noted that should this hard disk bottleneck be resolved while keeping the application responsible for the data delivery intact, a processor bottleneck during these peaks can be expected, as improving the hard disk performance would help feed more data to the processors, increasing their load.

The resources of this server could be better utilized with less bursty traffic, placing a more even load on the hard disks and the processors during execution.

All the test results can, however, be considered at least moderately suspect due to the nature of virtual servers. As the hardware resources can be shared with unknown actors, and this resource allocation can change even during the test operation and also possibly depending on measured performance requirements of running applications, no definitive conclusions on the root causes of these aforementioned problems can be given.

7 Conclusion

This project was began with the idea of using a ubiquitous piece of software for performing a relatively quick check for a database server under a heavy load. Being able to quickly establish the existence of a performance problem in the hardware and getting a general idea of its nature can lead to considerable savings in time and resources, as more detailed and specialized investigations normally require more time and, especially, more competent personnel.

Most of the research done for this project did not, in the end, lead to very useful data for this work. During the hours dedicated for the thesis at work, the performance problems encountered often led to carrying out research more in the scope of the current problem and less for the benefit of my thesis, which would have required a more general approach. After a great amount of pages of miscellaneous notes, most of them quite unusable for this work but otherwise potentially helpful, it became apparent that the project had wandered quite far from the original idea of the thesis. I was thus forced to almost start from the beginning and start searching for information solely for the Windows Performance Monitor, and forget all the intricate details that had been gathered for improving Oracle server performance.

Some readers might find the lack of some specific counters surprising. The initial plan was to write a chapter explaining reasons why some counters were *not* included, but in the end it was not done, as it would not really have served the purpose of this work. The selected performance counters give a general idea of the performance of all three major hardware resource areas and should help rise a red flag in most bottleneck situations.

In hindsight I should have concentrated only on the Windows Performance Monitor from the beginning and avoided researching any specific area too deeply, thus distracting from the main work. Also the lack of usefulness of this work on virtual servers came as a bit of a shock. When the actual hardware configuration is hidden, getting any real information on the underlying problems can be problematic. Some research on virtual server architectures would have been warranted, but would have escalated the amount of work required by this project to an unacceptable level.

Although the project had its difficulties, working on it was found educational. Some of the information that was not included in the thesis has found use in daily operations at Process Vision Oy and lessons learned in documentation should not be discounted.

References

1 Chan, I. Oracle database performance tuning guide 10g release 2 (10.2) [PDF]. Redwood City, California, United States: Oracle; March 2008. http://docs.oracle.com/cd/B19306_01/server.102/b14211.pdf. Accessed 26 January 2011.

2 Microsoft. Windows Performance Monitor [computer program]. Redmond, Washington, United States: Microsoft; 2003.

3 Huffman, C. Clint Huffman's Windows [online]. Redmond, Washington, United States: Microsoft; 22 November 2013; http://blogs.technet.com/b/clinth/. Accessed 19 December 2013.

4 Performance Analysis of Logs (PAL) Tool [computer program]. Version 2.0.7. Redmond, Washington, United States: Microsoft; 2011.

5 Hameed, CC. An overview of troubleshooting memory issues [online]. Redmond, Washington, United States: Microsoft; 2008. http://blogs.technet.com/b/askperf/archive/2008/01/25/an-overview-of-troubleshooting-memory-issues.aspx. Accessed 2 February 2011.

6 Microsoft TechNet. Monitoring context switches [online]. Redmond, Washington, United States: Microsoft; http://technet.microsoft.com/en-us/library/cc938606.aspx. Accessed 26 January 2011.

7 Huffman, C. PerfGuide: Analyzing poor disk response times [online]. Redmond, Washington, United States: Microsoft; 29 September 2010. http://social.technet.microsoft.com/wiki/contents/articles/perfguide-analyzing-poor-disk-response-times.aspx. Accessed 28 December 2012.

8 Muratore, F. Windows performance monitor disk counters explained [online]. Redmond, Washington, United States: Microsoft; 16 March 2012. http://blogs.technet.com/b/askcore/archive/2012/03/16/windows-performance-monitor-disk-counters-explained.aspx. Accessed 28 December 2012.

9 Rouse, M. Logical unit number (LUN) [online]. Boston, Massachusetts, United States: TechTarget; http://searchstorage.techtarget.com/definition/logical-unit-number. May 2011. Accessed 28 December 2012.

10 Thomas, G. The art and science of performance monitoring [e-book]. United Kingdom: Computer Performance LTD; 27 July 2006. http://www.computerperformance.co.uk/Poll/Sales_Perfmon.htm. Accessed 26 January 2011.

11 Choy, S. Taking your server's pulse [online]. Redmond, Washington, United States: Microsoft; 2008. http://technet.microsoft.com/en-us/magazine/2008.08.pulse.aspx?pr=blog. Accessed 4 February 2011.

12 Huffman, C. User mode versus privileged mode processor [online]. Redmond, Washington, United States: Microsoft; 28 Sep 2010. http://blogs.technet.com/b/perfguide/archive/2010/09/28/user-mode-versus-privileged-mode-processor-usage.aspx. Accessed 28 December 2012.

13 Huffman, C. The case of the high CPU web server [online]. Redmond, Washington, United States: Microsoft; 9 July 2009. http://blogs.technet.com/b/clinth/archive/2009/07/09/the-case-of-the-high-cpu-web-server.aspx. Accessed 4 February 2011.

14 Huffman, C. Choose your own adventure: high deferred procedure calls (DPCs) or High Interrupts [online]. Redmond, Washington, United States: Microsoft; 28 September 2010. http://blogs.technet.com/b/perfguide/archive/2010/09/28/choose-your-own-adventure-high-deferred-procedure-calls-dpcs-or-high-interrupts.aspx. Accessed 2 February 2011.

15 Huffman, C. The case of the 2 million context switches [online]. Redmond, Washington, United States: Microsoft; 28 Oct 2009. http://blogs.technet.com/b/clinth/archive/2009/10/28/the-case-of-the-2-million-context-switches.aspx. Accessed 26 January 2011.

16 Tulloch, M. Key performance monitor counters [online]. San Francisco, California, United States: TechGenix Ltd; 10 May 2005. http://www.windowsnetworking.com/articles_tutorials/Key-Performance-Monitor-Counters.html. Accessed 26 January 2011.

17 Huffman, C. PerfGuide: Low available RAM [online]. Redmond, Washington, United States: Microsoft; 18 September 2010. http://social.technet.microsoft.com/wiki/contents/articles/perfguide-low-available-ram.aspx. Accessed 7 February 2011.

18 Huffman, C. The case of the phantom hard page faults [online]. Redmond, Washington, United States: Microsoft; 16 July 2009. http://blogs.technet.com/b/clinth/archive/2009/07/16/the-case-of-the-phantom-hard-page-faults.aspx. Accessed 7 February 2011.

19 Microsoft. Microsoft Technet: Exchange server 2010 support: performance and scalability [online]. Redmond, Washington, United States: Microsoft; 23 July 2012. http://technet.microsoft.com/en-us/library/ff367896%28v=exchg.141%29.aspx. Accessed 2 January 2013.

20 Stinson, C. High avg disk queue length and finding the cause [online]. Edmonton, Canada. 12 September 2008. http://www.iishacks.com/2008/09/12/high-avg-disk-queue-length-and-finding-the-cause/. Accessed 1 February 2011.

21 Newton, T. Performance tuning Windows Server 2008 R2 Pt 2 [online]. Redmond, Washington, United States: Microsoft; 5 November 2010. http://blogs.technet.com/b/askperf/archive/2010/11/05/performance-tuning-windows-server-2008-r2-pt-2.aspx. Accessed 4 February 2011.