

The Performance Monitoring Poster

Performance monitoring is a mysterious domain. Often an issue is easily traced to a certain subsystem but sometimes a bottleneck is much harder to uncover. This poster offers a thorough overview of the relations between the individual subsystems and provides a guide to support the analysis of the more complex and hard to narrow down issues.

The five subsystems covered on this poster are ordered by complexity. To begin with, the famous processor, memory and disk subsystems are covered followed by a dive into the analysis of running processes and the kernel. The analysis for each of the subsystems is structured into steps with increasing complexity. If necessary or appropriate, a step contains references to other subsystems which may be participating in the issue.

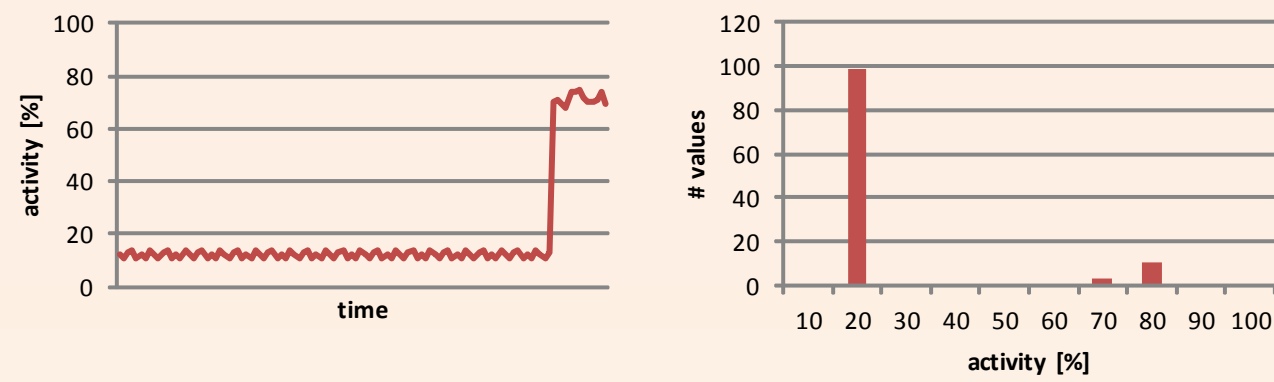
This poster describes performance analysis for the most important subsystems by providing vital insights, the relation to other subsystems and a step-by-step guide how to analyze the subsystem. It is recommended that you follow the five steps in the analysis of performance issues.

Updates to this poster will be published at <http://www.sepago.de/d/sepago-backstage/blogs/tags/poster>.

Introduction

Many of the described checks mention a threshold without providing a value. This is due to the fact that the threshold depends on the environment as well as the workload. For example, a threshold for the processor activity may be 80% to account for expected peaks. In another environment, the threshold describes a rather constant upper limit for the load and therefore may be set to 90% processor activity.

Several steps ask for a metric to be high but do not specify what high means. Again the notion of a high value depends on the analyzed environment. As a rule of thumb, the value for a metric can be considered high when it deviates strongly from a typical corridor. The left graph below displays a characteristic increase in processor activity by 50% due to a process gone astray and claiming a single CPU core on a two-core system. In such a case, activity changes by (1/# cores). Resolving the issue (often by killing the responsible process) results in an equivalent decrease of activity.



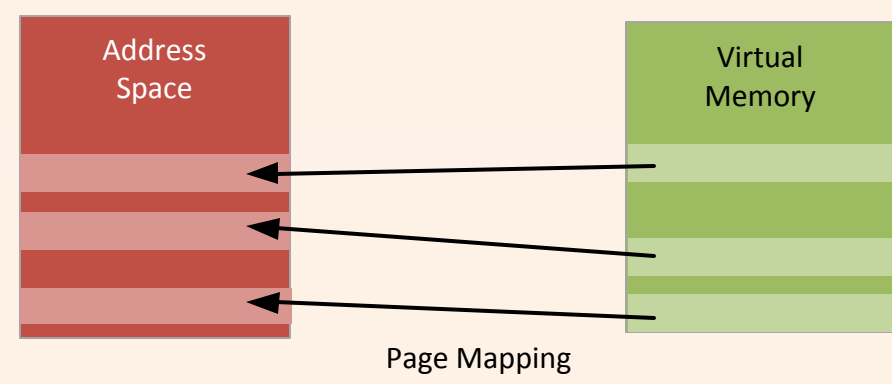
Humans are very good at spotting such peaks by merely looking at a graph. But at the same time, humans are easily distracted by values seeming extreme at first glance. This process can be (somewhat) formalized by plotting a histogram of all values contained in a series of values. An especially high or low value is set apart from the cluster of values. The right graph above shows that the activity is around 20% most of the time exposing values of 70-80% to be unusually high in a few cases. This should result in a detailed analysis when and why these high values are observed.

Virtual Memory

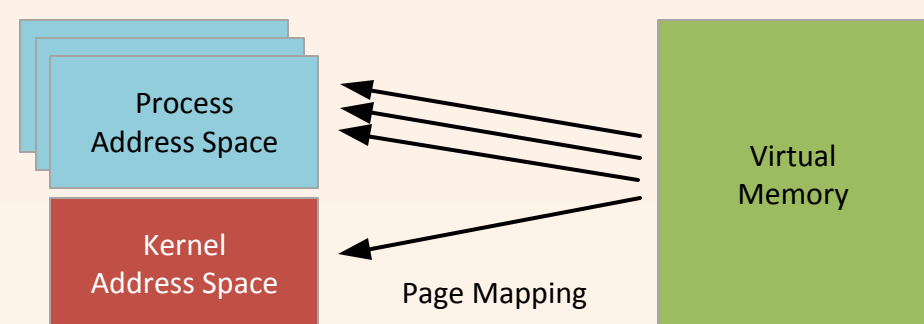
It is a very common mistake to confuse the address space with the overall memory. The address space denotes the number of bytes addressable by the system. Inside resides the memory available to the system which may be taking up only a fraction of the address space or even all of it. For example, with PAE a 32 bit system is able to address 64GB of memory but it may be equipped with only 4GB of physical memory.

System memory is addressed in pages which are mapped into the address space as required by kernel or processes through allocation.

The address space is split in two equally sized parts – one is reserved for the kernel and the other one is presented to processes as its own virtual address space. But due to the mapping of memory into the address space there is no such segmentation of system memory. Rather, it is reserved as necessary by either kernel or process.



As explained in the box „Memory“, only half of the address space is available to processes. In fact, every process works in its own virtual address space as if half of the address space was exclusively available to the process.



Read more about this topic in the Windows Internals books (<http://technet.microsoft.com/en-us/sysinternals/bb963901>).

About the Author

Nicholas Dille, MVP for Remote Desktop Services
(<http://blogs.sepago.de/nicholas>, @NicholasDille, nicholas.dille@sepago.de)

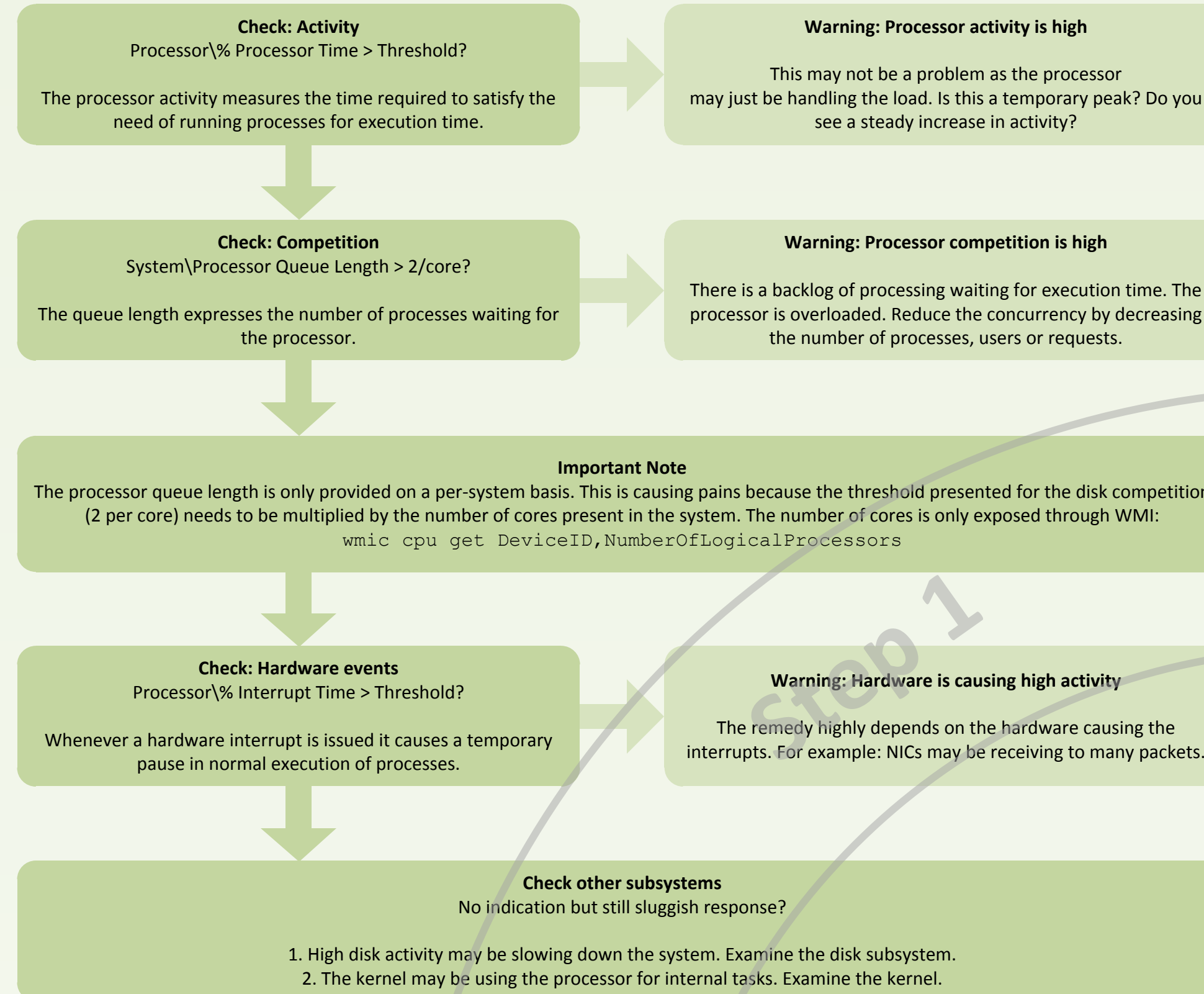
Nicholas Dille is an IT architect at sepago GmbH and has been engaged in enterprise projects for many years. He specializes in centralizing IT infrastructures, consolidating resources and managing capacity. In his community blog, he writes articles with deep technical insight around Remote Desktop Services and related technologies. sepago is an IT consultancy located in Cologne, Germany (<http://www.sepago.de>) and has achieved Microsoft Gold partner status as well as Citrix Platinum partner status.

Copyright © 2011 by Nicholas Dille

Processor

All processes running on a system are competing for execution time. As modern processors have several cores, the operating system does a very good job of moving processes between cores to spread the load equally. The load on the processor is monitored by two metrics: the activity and the competition. The activity of the processor is expressed by the percentage of time it was actively servicing processes. The competition is determined from a queue containing processes waiting for execution time. Often, only the processor activity is examined resulting in an incomplete view of this subsystem.

In addition to these parameters, the processor also services hardware interrupts (caused by notification sent by hardware components) and internal kernel tasks.



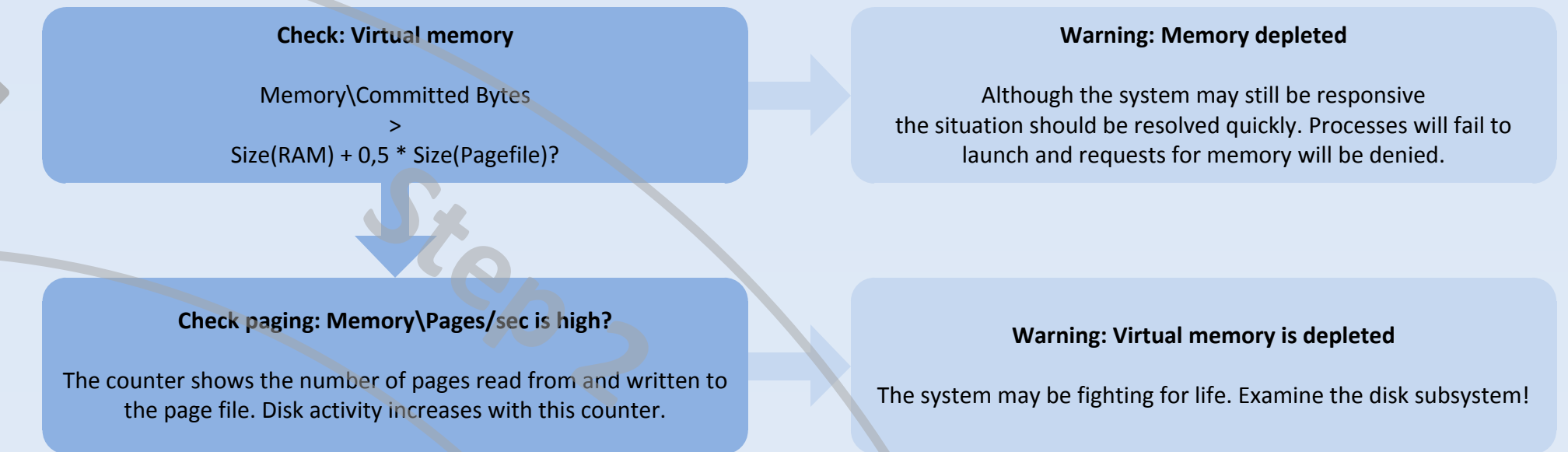
Memory

The memory subsystem maintains the available virtual memory consisting of the physical memory built into the system and the pagefile(s) configured for the system. To gain a valid overview of the state of the memory subsystem, both of these aspects need to be regarded. Windows begins paging memory to the pagefile soon after system startup to maintain as much unused physical memory as possible so that processes can quickly allocate a maximum amount of memory. Therefore, monitoring the pagefile usage is as important as the available memory. But using the page file causes I/O operations with the corresponding physical disk, so that the disk subsystem needs to be examined as well.

The amount of physical memory is well presented through WMI because it is a hardware asset. It can be retrieved by the following command:
`wmic memphysical get Name,MaxCapacity`

Judging by name, a 32-bit system can only address a maximum of 4GB of physical memory. By using the Physical Address Extension (PAE) a 32-bit system is able to use up to 64GB of memory. The maximum amount of usable memory also depends on the Windows edition used. Although PAE can be enabled on Windows Server 2003/2008 Standard Edition, it only serves to counter the negative effects of drivers masking physical memory. Drivers often require I/O address ranges to access the hardware devices. These address ranges are usually placed below the 4GB limit rendering portions of the physical memory unusable. By enabling PAE on Windows Server 2003/2008 Standard Edition, these address ranges are moved beyond the 4GB border and additional physical memory is made available to the kernel and processes. This does not work for Windows clients as these are limited to an address range of 4GB regardless of PAE. The different behaviour is caused by the more strict requirements for drivers on Windows servers. As PAE makes driver development more complex, it is not enforced for consumer devices. On a typical, physical server, PAE can make up to 650MB of physical memory available to the system. On modern laptops, more than 1GB is masked by address ranges for device drivers.

Examining the memory subsystem by looking at the available bytes and the pagefile usage is not conclusive, because Windows begins paging memory to the pagefile as soon as it has booted. Therefore, this poster is based on examining the commit charge. The operating system maintains a metric called Commit Limit which denotes the overall amount of virtual memory available to the system. It is represented by the sum of physical memory built into the system and pagefile(s) available to the system. The Commit Limit may change while the system is running because Windows may be allowed to manage the size of the pagefile(s) automatically and decide to shrink or enlarge it. By using the metric Committed Bytes, performance monitor exposes the number of bytes used by the system regardless of the located (physical memory or pagefile). Using the commit charge instead of the available physical bytes gives an impression when the system will hit the hard limit of the available virtual memory. In addition, monitoring the effect of paging on the performance of the physical disk is essential because paging activity is the primary metric to determine whether the memory subsystem is overloaded. If paging activity increases, the disk subsystem needs to be analyzed.



Don't use the metric Page Faults/s instead of Pages/s because it also includes soft page faults which are resolved without reading pages from the physical disk. Whenever the operating system flags a page to be removed from physical memory, it is not immediately written to the pagefile instead rather the page reference is added to the standby list maintained by the kernel. As long as the page is only flagged for paging out, a request for the process memory mapped to the page is called a soft page fault because it can easily be resolved by restoring the page to the active page list. And even after the page has been written to the pagefile, it may still be recovered quickly if the corresponding physical memory has not been reused. Only when the page has been assigned again will a hard page fault be caused by accessing the originally virtual memory area. It will then have to be retrieved from the physical disk resulting in a time consuming process delaying the normal execution of the corresponding process.

Kernel

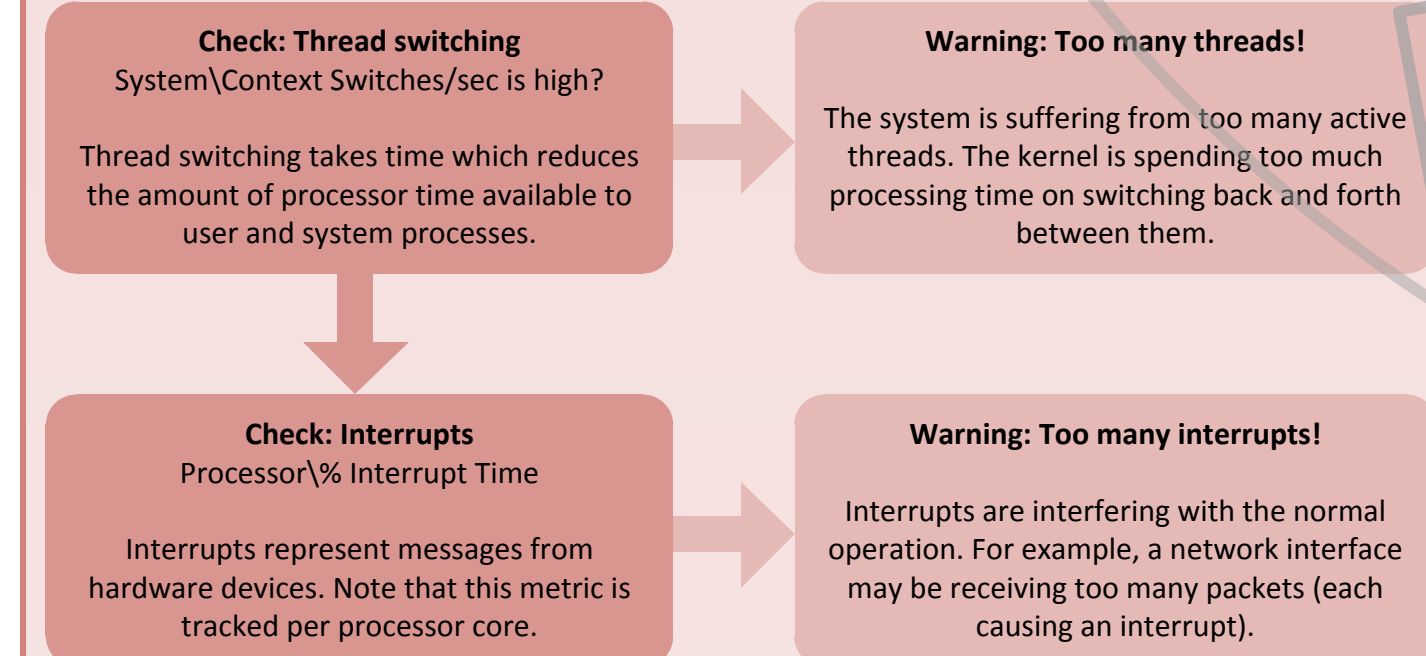
The kernel represents the core of the operating system. In general it is not necessary to monitor the operation of the kernel on a regular basis. But there are issues that are caused by bottlenecks in the kernel. This is often the case when a bottleneck cannot be traced to a specific subsystem.

The kernel maintains several memory areas for data structures that are required for the proper operation of the system. The Non-Paged Pool is used for storing data that cannot be paged out to the disk as it is crucial to the operation of the kernel (like the data structures relevant to process and memory management) and therefore uses precious space in physical memory. The Paged Pool – in contrast - contains data that may be paged to the disk.

Both metrics are expressed by an upper limit and the current usage which can be displayed using Process Explorer when debugging symbols are configured. Many performance issues arise from these memory areas being depleted.

Due to the architecture, 32-bit systems have inherently low upper limits for (Non-)Paged Pool. Even when preset to the maximum value, these memory areas are easily exhausted on highly loaded systems. For example, terminal servers suffer from the effects when the user density is pushed. On a 64-bit system, these limits have been raised so high that it is not perceivable today when they will be reached but both metrics are primarily limited by the amount of physical memory available to the operating system.

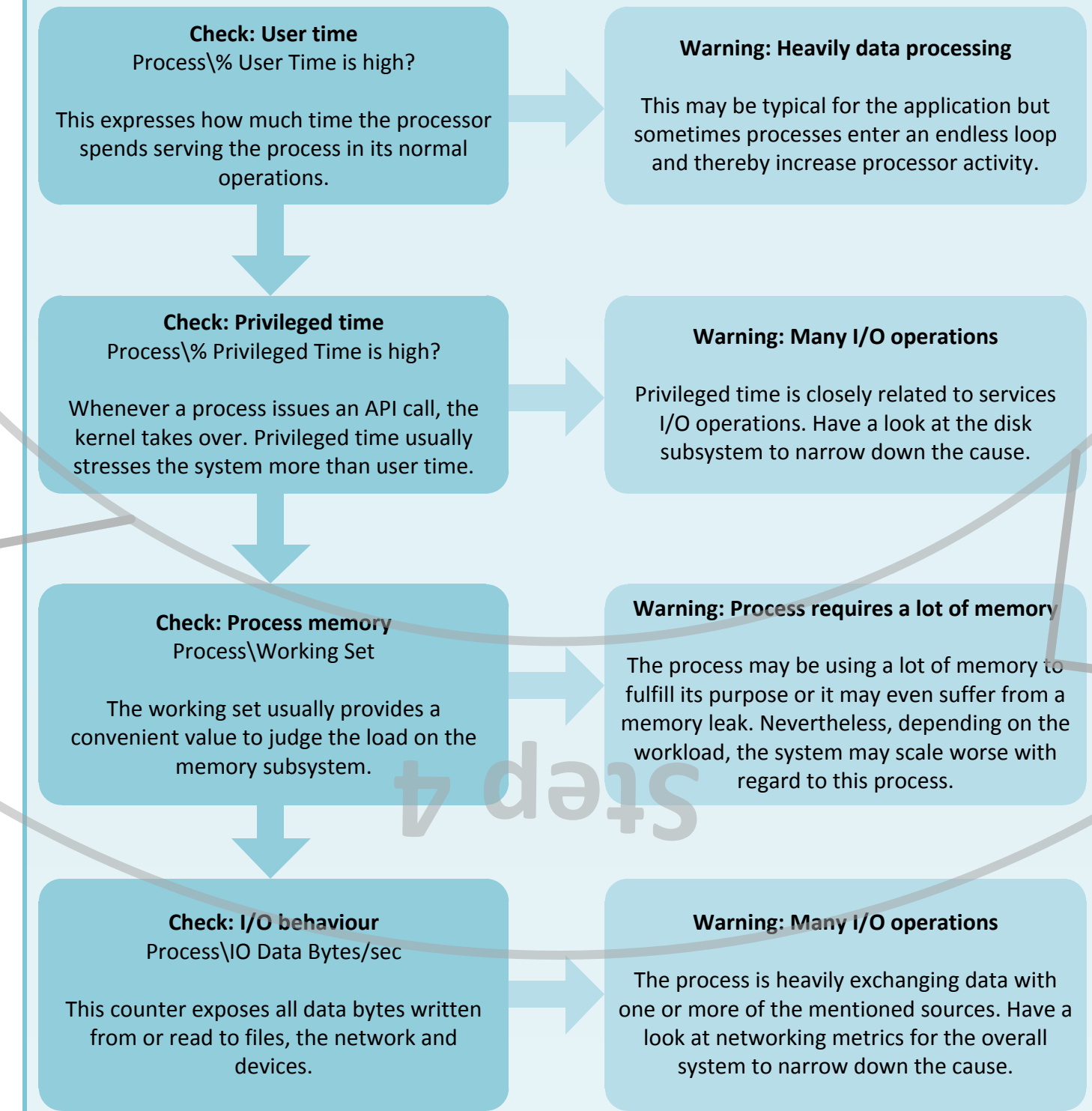
Apart from these special memory areas, the kernel also collects data about itself servicing requests (e.g. interrupts) and processes (context switches). The latter provides a very interesting piece of information as the number of context switches per second measures how often a new thread is granted a time slice of a processor core. The higher this metric the more time needs to be invested by the processors to switch between threads. As this task is expensive (in terms of processing time), the processor may be eating precious time switching between competing threads causing the processor resource to perform worse. It is important to note that this is an internal task which is not showing on processor activity.



Processes

Monitoring processes is very similar to watching the whole system. As each and every process utilizes all subsystems mentioned on this poster, the impact of a process on the system can be determined by analyzing the impact on every subsystem. But usually a process exerts most of its load on a single subsystem.

The processor can serve a process to perform complex calculations on data and thereby cause an increased load on the processor. When a process is working on a large amount of data, this may increase the memory footprint. Also the exchange of data with other processes or systems can raise the traffic on physical disks or the network.



Physical Disk

The physical disk is responsible for storing various kinds of data. First of all, the operating system causes load itself by accessing vital files, libraries and configuration data. Next, applications are launched from the disk and require more or less frequent reads and writes. Last, the pagefile increases disk activity because it is used to extend the physical memory built into the system.

It is inherently hard to separate the load caused from these three domains. Although disk I/O can be measured for a process, it is not practical to add all those values to a performance analysis. In Windows Server 2008/Vista, Microsoft introduced the Resource Monitor (perfmon/res) which (above other metrics) displays the disk activity caused by every process. It is a very handy tool for a quick overview how running processes use the disk.

The fact that the pagefile is stored on the physical disk creates a close relationship with the memory subsystem. Whenever the memory subsystem is under load, paging causes disk activity by accessing the pagefile and thereby increasing the load on the physical disk. The box labeled „Memory“ contains hints how to measure the effect of the memory subsystem on the disk subsystem.

Performance data is exposed for all physical disks individually. A list of local hard drives can be retrieved from WMI with the following command:
`wmic pagefile get Name,AllocatedBaseSize`

Two metrics are of high interest when examining the physical disk: the activity and the competition. The activity is expressed in a percentage of time and the competition in a number of processes waiting for execution time. To determine the load caused by the memory subsystem, refer to the details about paging in the box „Memory“.

