What Is PAE X86?

Article10/08/2009

Applies To: Windows Server 2003, Windows Server 2003 R2, Windows Server 2003 with SP1, Windows Server 2003 with SP2

What Is PAE X86?

In this section

- Common PAE X86 Scenarios
- Technologies Related to PAE X86
- Related Information

Physical Address Extension (PAE) X86 is one of two technologies that increase the amount of physical or virtual memory available to user mode applications. The other technology is 4-gigabyte tuning (4GT). For more information about 4GT, see "4GT Technical Reference."

By default, servers running the 32-bit editions of Windows Server 2003, Enterprise Edition, and Windows Server 2003, Datacenter Edition, support only 32-bit memory addressing and thus are able to access only 4 gigabytes (GB) of physical memory. However, hardware that offers much larger amounts of physical memory is available. Both the operating system and many applications can benefit from using this additional memory.

PAE X86 allows servers running Windows Server 2003, Enterprise Edition, and Windows Server 2003, Datacenter Edition, on a 32-bit hardware platform to access physical memory beyond 4 GB. To do this, PAE changes the memory addressing from 32-bit addressing mode to 64-bit addressing mode, allowing the operating system and high performance drivers and applications access to the additional physical memory. When using the larger amount of physical memory made available by PAE X86, operating system performance can be increased, particularly when the server is hosting multiple applications. Applications can also have direct access to the physical memory beyond 4 GB provided they use the Address Windowing Extensions (AWE) API set.

When using PAE X86, all of the physical memory in the system is treated as general purpose memory. The operating system is able to use this memory for caching and virtual memory management without major changes.

Applications can also use this memory without major changes. The most noticeable difference is that more applications can be running before paging occurs because there is more memory for each process and its associated virtual address space. This also means that more processes can be resident in memory. Because applications not needing larger physical memory do not need to be modified or use the AWE API set, virtually all applications can benefit from PAE X86.

Common PAE X86 Scenarios

PAE X86 allows the Windows executive software (also known as the kernel) to take full advantage of all available physical memory, including memory above 4 GB, up to 64 GB. This can result in a significant reduction in paging operations and can improve the performance of the server when multiple applications are hosted on a single computer, such as in application consolidation scenarios.

Additionally, certain data-intensive applications, such as database management systems and scientific and engineering software that need access to very large caches of data can also use PAE to address memory beyond 4 GB. These

applications can use the AWE API to access up to 64 GB of memory using a memory window inside their 4 GB process. Just four system calls are used to declare the AWE window and to map additional physical memory inside that window.

Note

- PAE X86 is not required on the 64-bit versions of the Windows Server 2003 family.

Technologies Related to PAE X86

The following technologies and components are integral in understanding PAE X86.

Memory Manager

The memory manager translates virtual memory addresses used by the operating system and applications to actual physical memory locations. The translation of virtual memory to physical memory is transparent to the application. User mode processes are never able to directly write to real memory and never actually know where their data resides. A user mode process can request a block of memory and write to it. The data written to the memory location might be written to real memory, or might be written to a paging file. A paging file (also known as a swap file) is a file on the hard disk that the memory manager uses to hold data that does not fit in memory. The memory manager moves data from the paging file to memory as needed and moves data from memory to the paging file to make room for new data.

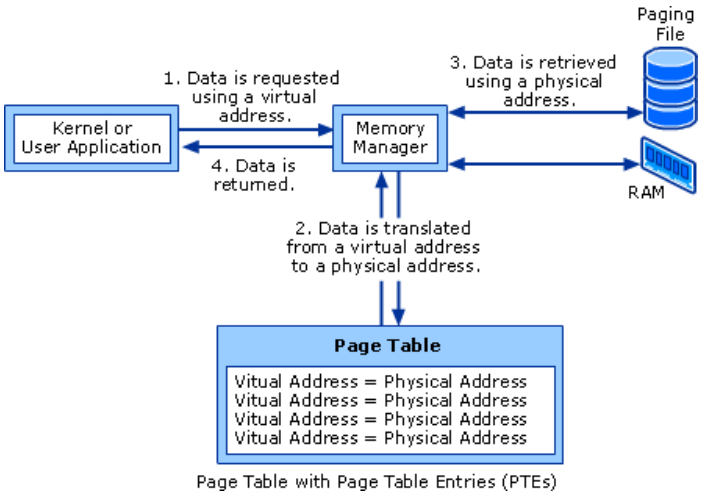
The 32-bit versions of Windows are able to address 232 bytes (4 GB) of memory by default. PAE X86 changes that addressing from 32-bit to 64-bit addressing mode. Even with the additional memory made available through PAE X86, if the total memory requirements of all applications exceeds the physical memory available, the memory manager uses the paging file to substitute for real memory; overcoming the physical memory limitation.

The paging process allows the operating system to overcome the real physical memory limits. However, it also has a direct impact on performance because of the time necessary to write or retrieve data from disk. PAE X86 modifies the way the memory manager translates the 32-bit virtual memory address to a physical address, allowing the address to be translated into 236 potential physical memory addresses. Access to the additional memory then reduces the amount of paging operations needed.

Page Table Entries

As described in the previous section, the memory manager translates, tracks, and organizes real and virtual memory for applications and for portions of the kernel. Because of this, the memory manager must index where in real memory or where in a page file the data resides. When an application asks for the data, the memory manager can refer to the index to determine where that data actually resides and then retrieve it for the application. The following diagram shows a simplified explanation of the translation process.

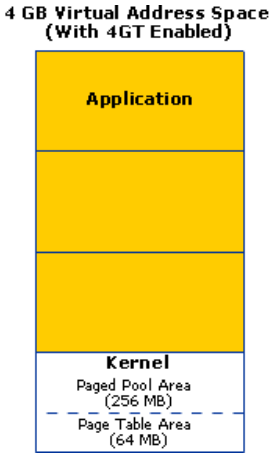
Use of Page Table Entries



Page Table with Page Table Entries (PTEs)

The Page Table, one of the internal tables used by the memory manager in translating virtual memory addresses to physical addresses, resides in the kernel's memory space. The kernel's allocated memory space (either 2 GB by default, or 1 GB with 4GT enabled) is divided by function into numerous sections, two of which are the Paged Pool area and the Page Table area, as shown in the following figure.

Page Pool and Page Table Areas



On a computer running a member of the Windows Server 2003 family, the Page Table area, which contains the Page Table Entries (PTEs), is 64 megabytes (MB) by default when 4GT is enabled. The Paged Pool space is approximately 256 MB. While these different allocations share the same memory area, the partition between them is fixed at startup. This becomes important when PAE X86 is enabled because the access to additional physical memory that PAE X86 allows requires additional PTEs for tracking and translation. If the operating system runs out of space in either the PTE area or the Paged Pool area, the other area cannot donate space to it, and programs may not receive requested resources, which can lead to unexpected errors for the application. By editing the registry, it is possible to limit the size of the Paged Pool space, which will then free additional memory for system PTEs.

Note

- Incorrectly editing the registry may severely damage your system. Before making changes to the registry, you should back up any valued data on the computer.

4-gigabyte tuning (4GT)

Another memory management technology is 4-gigabyte tuning (4GT) that can, in certain scenarios, but used in conjunction with PAE X86.

By default, the 32-bit editions of Windows Server 2003, Enterprise Edition, and Windows Server 2003, Datacenter Edition, provide applications with a flat 32-bit virtual address space that can address up to 4 GB of virtual memory. The address space is usually split so that 2 GB of virtual address space is directly accessible to the applications and the other 2 GB is only accessible to the kernel. Because of this 2 GB virtual memory limit, applications that are memory intensive and that manage their memory directly through their own caching methods, such as database management systems (DBMS), can experience reduced performance.

4GT can make more of the computer's virtual memory available to applications by making less virtual memory available to the operating system. With 4GT enabled, applications that are designed to be aware of this larger address space are able to address 3 GB of virtual memory instead of the 2 GB normally allocated for user mode processes. This is a 50 percent increase in virtual memory, allowing more data to be cached and potentially significantly increasing performance.

In many situations, it is valuable to use both 4GT and PAE X86. The combination of these two technologies maximizes the amount of physical memory available to applications. However, 4GT cannot be used with PAE X86 when the server has more than 16 GB of physical memory. In this situation, 4GT should be disabled and only PAE X86 should be used.

Application Windowing Extensions (AWE)

Because an application's virtual address space can only be extended to a maximum of 3 GB (using 4GT) in a 32-bit operating environment, in order to increase performance, applications (and database applications in particular because they use large datasets and do extensive swapping of data from disk to memory) need a method to map large portions of data into memory and keep that data in real memory at all times. This ability should also allow an application to map and remap data into its virtual memory space very quickly.

Application Windowing Extensions (AWE) is the solution to this need. AWE is an application programming interface (API) set that complements PAE X86. AWE is a set of APIs that allows software developers to create applications that use up to 64 GB of physical non-paged memory in a 32-bit virtual address space on 32-bit platforms. This technology allows for windowed views to this physical memory from within the application's virtual address space.

AWE allows an application to reserve sections of real memory that cannot be paged out to the paging file or otherwise manipulated, except by the reserving application. AWE will instead keep the data in real memory at all times. Because the memory manager does not manage this memory, it will never be swapped out to the paging file.

Unlike PAE, AWE is not an option set at startup. Applications must directly invoke the AWE APIs in order to use them, and the applications must be developed using the AWE APIs in order for the applications to benefit from the AWE APIs.

Related Information

The following resource contains additional information that is relevant to this section.

- "4GT Technical Reference"