

# Incorporating Hyper-Threading Technology into a NetWare-Based Platform

*By Dana Henriksen*

Introduced in the Intel® Xeon™ processor family, Hyper-Threading technology presents an alternative to traditional processing configurations. One Hyper-Threading processor comprises two virtual processors, which can simultaneously process different threads of a program. The Novell® NetWare® 6 operating system supports this processor technology.

Intel® Hyper-Threading technology enables two logical processors to perform different tasks simultaneously while sharing hardware resources. According to Intel, this technology can increase CPU resource utilization an average of 40 percent, yielding higher processing throughput.<sup>1</sup> By taking advantage of idle on-die resources, Hyper-Threading technology can boost the performance of multithreading and multitasking operations running on Intel NetBurst™ microarchitecture-based processors.

Hyper-Threading incorporates two logical processors in one physical processor package. The logical, or virtual, processors share most of the resources on the chip such as the local caches, execution units, and performance monitor registers. Because only one instance of the shared resources must be purchased, a Hyper-Threading package costs less per processor. These virtual processors also have resources dedicated to them such as general and local APIC (advanced programmable interrupt controller) registers. The kernel treats the virtual processors in almost the same manner it treats dedicated processors.

## Supporting Hyper-Threading in Novell NetWare

Support Pack 1 (SP1) of the Novell® NetWare® 6 operating system provides full kernel support of Hyper-Threading technology.

Administrators implementing a NetWare-based Hyper-Threading platform also need the ACPI (Advanced Configuration and Power Interface) Platform Specific Module (PSM). NetWare 5.1 does not yet support Hyper-Threading.

### Licensing requirements for multiprocessors

NetWare does not require special licenses or additional fees for multiprocessor support. Whether adding Hyper-Threading virtual processors or fully dedicated processors, the additional software license cost is the same: zero.

### ACPI tables for discovering processors

Because some operating systems are not enabled for Hyper-Threading, Intel prevents those operating systems from encountering the virtual processors by excluding the processors from the BIOS Multi-Processor Specification (MPS) tables. Instead, Intel lists only one processor per package, rather than two, in the MPS tables. It lists all processors in the BIOS ACPI tables.

To read the ACPI tables, Novell has developed a PSM that is available in NetWare 6 SP1. This ACPI PSM reads the ACPI tables, discovers all the processors, and registers them with the operating system. The ACPI tables are then available for use by the kernel.

<sup>1</sup> For more information about Hyper-Threading technology, please visit <http://www.intel.com/technology/hyperthread>.

Novell has not yet developed an ACPI PSM for NetWare 5.1.

Discrepancies in the ACPI tables may occur in different vendors' BIOS. Because ACPI is relatively new, compared to MPS, it has not been fully tested. Although Microsoft® Windows®, Linux®, and NetWare operating systems use the same ACPI specification, they do not always use the same fields in the tables. A problem that does not occur with one operating system may occur with another one. Until the ACPI BIOS of different hardware platforms have been tested with NetWare 6.1 and the ACPI PSM, some ACPI BIOS problems may be encountered.

### Load balancing with Hyper-Threading technology

Because Hyper-Threading processors share execution units, systems can achieve better performance results if the thread load balancing is sensitive to which virtual processors are in which processor packages.

For example, in a scenario where two threads are running on a system with four virtual processors in two processor packages, the system will perform better if one virtual processor in each package handles a thread while the other virtual processor in each package remains idle. If both threads are executed in the same package, they will compete for shared resources and experience a performance penalty while the other package is completely idle.

The Novell NetWare scheduler is sensitive to Hyper-Threading and balances the threads across the processors by using one virtual processor in each package before assigning threads to the other virtual processors in the packages.

### Initial APIC IDs

Using the CPUCHECK console command or the processor information page in the NetWare Remote Manager, system administrators can view the initial APIC ID of each virtual processor. This ID number helps identify which virtual processors are in the same processor package.

The initial APIC IDs are grouped in even-odd pairs, with an even number and the next sequential odd number paired in the same physical processor package. Thus, zero and one are in the same package, two and three are in the same package, and so forth. With this information, administrators can monitor the thread load balancing, and by using the start and stop processor commands, they can create test configurations that measure how Hyper-Threading contributes to performance.

### Testing the performance of Hyper-Threading systems

When executing instructions, virtual processors often do not use all the execution units, leaving the remaining execution units

By taking advantage of idle on-die resources, Hyper-Threading technology can boost the performance of multithreading and multitasking operations.

available for the other virtual processor in the package. Instructions that stall the processor, such as those that cause cache contention, wait for microcode support, or wait for memory access, leave shared resources available for the other virtual processor to use. The cache contention caused by two processors fighting for the same piece of memory disappears when the two processors are in the same package. Programs that cause significant cache contention and perform poorly on multi-processor (MP) systems perform better on Hyper-Threading systems because of the reduced cache contention.

Conversely, when code is highly optimized and seldom stalls the processor, fewer shared resources are available for the other virtual processor in the package. The virtual processors must wait for each other when using the shared resources, and the resulting performance is much less than that achieved with two dedicated processors.

Most code falls somewhere between never and always stalling the processor. The performance improvement depends on the nature of the code. Figure 1 shows the results of code execution tests using different processor configurations. The tests compared the performance of three configurations: 2 dedicated processors, 2/2 processors (two Hyper-Threading processor packages with two virtual processors each, for a total of four virtual processors), and 4 dedicated processors. The values shown in the figure represent the percentage performance gain or loss when compared to another processor configuration.

### Code execution tests

The Context Switch test used highly optimized code and avoided stalling the processor. It kept the shared execution units busy;

Test	Effect on system performance when shifting processor configuration from:		
	2 to 2/2 (From two dedicated processors to four virtual processors)	2 to 4 (From two dedicated processors to four dedicated processors)	4 to 2/2 (From four dedicated processors to four virtual processors)
Context Switch	9% decrease	78% increase	49% decrease
L2 Cache Read/Write	15% increase	39% decrease	89% increase
Spinlock	8% increase	27% decrease	47% increase
Cache Contention	9% decrease	42% decrease	57% increase
Checksum	45% increase	52% increase	4% decrease
Expensive Instruction	53% increase	63% increase	6% decrease

Figure 1. Comparing the performance of a Hyper-Threading configuration with other processor configurations


the virtual processors waited for each other to use this shared resource. Performance drops slightly when shifting from two dedicated processors to four virtual processors, yet rises when shifting from two dedicated processors to four dedicated processors. When comparing the four-dedicated-processors configuration with the four-virtual-processors configuration, the Hyper-Threading configuration performs 49 percent worse.

The L2 Cache Read/Write test, Spinlock test, and Cache Contention test all cause significant cache contention and memory stalls. These tests stall the processor frequently, leaving the shared resources available for the other virtual processor. When shifting from two dedicated processors to four virtual processors, performance rises (except on the Cache Contention test). Also, because of the high amount of cache contention in these tests, the four-dedicated-processors configuration performs worse than the two-dedicated-processors configuration. More importantly, four virtual processors perform these tests significantly better than four dedicated processors. Thus, applications that do not scale well on an MP system may likely scale much better on a Hyper-Threading system.

The Checksum test and the Expensive Instruction test both stall the processor with expensive work, but they avoid cache contention. These tests scale well when shifting from two to four dedicated processors as well as when shifting from two dedicated to four virtual processors. As expected, the four-dedicated-processors configuration performs slightly better than the four-virtual-processors configuration.

In summary, the code execution tests showed that processor configuration performance depends on the nature of the code run on the processors.

### Introducing Hyper-Threading technology into a NetWare environment

By taking advantage of idle hardware resources, Intel Hyper-Threading technology can offer advantages over dedicated processor configurations. For many programs, Hyper-Threading technology can increase CPU performance and achieve greater processing throughput. Novell NetWare 6 supports this technology and can help build the ideal platform for a Hyper-Threading system. 

**Dana Henriksen** ([dana@novell.com](mailto:dana@novell.com)) is the kernel architect for Novell NetWare 6. He has worked at Novell for 15 years and plays a key role in the development of the NetWare operating system.

#### FOR MORE INFORMATION

**Intel Hyper-Threading technology:**  
<http://www.intel.com/technology/hyperthread>

**Novell NetWare:**  
<http://www.novell.com/products/netware>