Intel® Hyper-Threading technology

technology brief

Abstract 2

Introduction 2

Hyper-Threading 2
- Need for the technology 2
- What is Hyper-Threading? 3
- Inside the technology 3
- Compatibility 4

Issues with Hyper-Threading 4
- Applications 4
- OS 5
- Licensing 5

Performance gains 6

Conclusion 7

For more information 8

Call to action 8

Abstract

Intel's introduction of Hyper-Threading technology represents a significant improvement in processor utilization and performance. This technology boosts system performance without going to a higher clock rate or adding more processors. This improvement is achieved by making multiple instruction streams, called threads, internally available to a single processor at the same time. These threads allow the processor the opportunity to better schedule the use of internal resources and improve utilization.

HP servers offer this new technology by making use of Intel® Xeon processors, which incorporate Hyper-Threading. This technology brief describes the Hyper-Threading concept, as well as its benefits and limitations for the user. Some HP performance test results are also included to show the improvement seen by the use of Hyper-Threading.

Introduction

Managers of today's enterprise environments continually face the need to lower costs and improve performance. A straightforward means to improve performance has always been to increase processor speeds, but this typically comes at a higher cost. To find a solution that attacks both cost and performance at the same time requires an improvement in the utilization of existing resources.

One of Intel's solutions in this regard was to focus on threading at the system level through software. Using parallel threads of instructions has worked well in multiprocessor systems, since different processors can simultaneously operate on different threads. Taking this a step further, Intel's Hyper-Threading solution offers a unique approach by providing thread-level parallelism on each individual processor.

This technology brief discusses the need for Hyper-Threading, the basics of the concept, the benefits it provides, issues that have arisen in regard to software licensing, and some results that HP has seen in laboratory tests.

Hyper-Threading

This new technology from Intel enables multi-threaded applications to execute threads in parallel on each individual processor. Available on Intel Xeon processors, Hyper-Threading provides the user with increased computing power to meet the needs of today's server applications.

Need for the technology

Improving processor utilization has been an industry goal for years. Processor speeds have advanced until a typical processor today can run at frequencies over 2 gigahertz, but much of the rest of the system is not capable of running at that speed. To enable performance improvements, memory caches have been integrated into the processor to minimize the long delays that can result from accessing main memory. Xeon processors, for example, now include three cache levels on the die.

Large server-based applications tend to be memory intensive due to the difficulty of predicting access patterns. The working data sets are also quite large. These two things can create bottlenecks, regardless of memory prefetching techniques. Latency due to these bottlenecks only gets worse when pointer-intensive applications are executed. Any mistake in prediction can force a pipeline to be cleared, incurring a delay to refill this data.

It is this latency that drives processor utilization down. Despite improvements in application development and parallel processing implementations, reaching higher utilization rates remained an unmet goal.

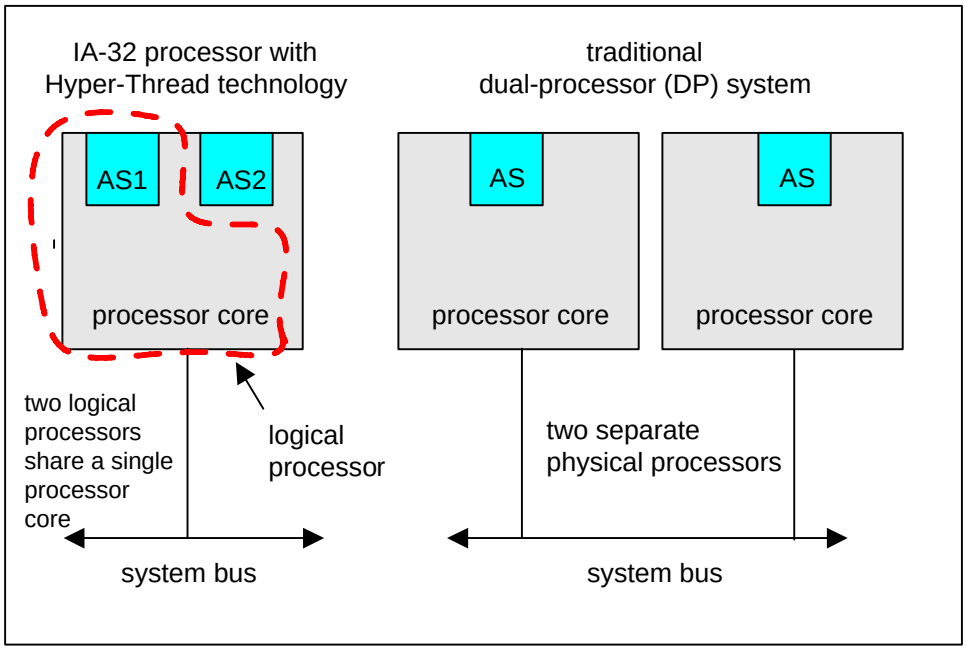
What is Hyper-Threading?

Hyper-Threading Technology enables one physical processor to execute two separate threads at the same time. To achieve this, Intel designed the Xeon processor with the usual processor core, but with two Architectural State devices (see Figure 1). Each Architectural State (AS) tracks the flow of a thread being executed by core resources.

After power-up and initialization, these two internal Architectural States define two logical processors. Individually they can be halted, interrupted, or can execute a specific thread independently of the other logical processor. Each AS has an instruction pointer, advanced programmable interrupt controller (APIC) registers, general-purpose registers, and machine state registers.

The two logical processors then share the remaining physical execution resources. An application or operating system (OS) can submit threads to two different logical processors just as it would in a traditional multiprocessor system.

Figure 1. Conceptual illustration of Hyper-Threading



Inside the technology

Looking inside the processor we find that the core contains subsystems to enhance performance. These subsystems control program execution, perform instruction fetching, integrate the on-die cache, and handle all the instruction reordering and retiring.

As threads are passed to the processor, the instruction fetching and reordering systems allocate resources to the incoming threads. The instructions in these threads are then sent to the execution system in an alternating fashion from the level 1 cache. This continues until one of the logical processors no longer needs information from the level 1 cache and then the entire cache resource is allocated to the other logical processor.

The execution core processes instructions in an order determined by dependencies in the data and availability. The processor is allowed to execute instructions out of order, that is, in a different order than the order in which they arrived. This means instructions can be executed in the order that will

yield the best overall performance. Schedulers inside the execution system handle the mapping and ordering, and they may send multiple instructions from one processor before executing instructions from the other.

The cache system provides data to the execution system at high speeds and with larger cache lines than previous processors used. The cache operates at the same speed as the execution core so that future versions of the processor will continue to operate at correspondingly faster rates. The Xeon MP, intended for systems with four or more processors, is equipped with an integrated third-level cache to reduce the competition for shared resources between processors in the same system.

Both of the logical processors inside the physical processor share all the internal caches. The cache design implements a high-level of set-associativity to minimize the possibility of the logical processors constantly throwing each other's data out of the cache to make room for their own data. In some cases, one processor may be able to fetch instructions or data into the cache for the benefit of the other processor in order to improve overall execution rates.

The instruction reordering and retiring system eventually completes all the out-of-sequence instructions that were executed, and then retires them in the original program order. Upon completion, instructions are retired much the way they were originally sent, with the logical processor taking turns.

Compatibility

Hyper-Threading on Xeon processors is fully backward compatible with current operating systems and applications. Legacy operating systems with multiprocessor capability can run unmodified on Xeon-based HP systems. Some legacy systems, such as Windows NT, may not recognize the additional logical processors, and therefore cannot take advantage of Hyper-Threading Technology.

Operating systems such as Windows 2000 Server, Linux, and Novell NetWare can recognize both of the logical processors and take advantage of Hyper-Threading Technology, depending on OS license configurations. Some of these same operating systems are expected to include optimizations for Hyper-Threading so they can distinguish between logical and alternate processors, enabling them to optimize scheduling and improve idle loops to maximize performance gains. For example, Windows 2003 Server (formerly referred to as Windows .NET) has this capability.

A recent paper published by Microsoft on the topic of software compatibility stated that, "Although Windows 2000 is compatible with Hyper-Threading Technology, we expect customers will get the best performance from Hyper-Threading Technology using Windows .NET Server. This is because the Windows .NET Server Family is engineered to take full advantage of the logical processors created by Hyper-Threading Technology. Microsoft expects to see positive performance gains with Windows .NET Server and Hyper-Threading Technology, while Windows 2000 performance gains are expected to be more modest."

Issues with Hyper-Threading

Reducing latency in a system by providing dual paths to an underutilized processor core should improve system performance. Still, there are some potential issues that should be examined, though these issues do not affect everyone.

Applications

Hyper-Threading Technology can actually produce a performance loss if the load at the logical processors is not balanced. Two logical processors share resources at the execution core and as a result no single processor is able to use all the resources that would normally be available to a single processor that did not implement Hyper-Threading. If one thread of an application were working and the other thread were waiting (spinning), the operating thread would still have less than 100 percent

of the resources. An effective load balance for a Hyper-Threading system is imperative to reduce the chances that only one thread will be active.

With two logical processors sharing execution resources, the effective size of the cache with which each can operate is approximately half the actual cache size. Applications written for multithreading should therefore expect to have only half the cache available for each thread. When considering code size optimization, for example, excessive loop unrolling should be avoided. Although cache sharing may be an issue for some applications, it does provide better cache locality for other applications. For example, an application might use one logical processor to fetch data into the shared caches to reduce latency for the other logical processor.

OS

Operating systems use logical processors on a Hyper-Thread processor just as they do any other processor, by scheduling threads to operate on each. For optimal performance in a Hyper-Threading system, the OS should provide these optimizations:

- Idle-loop and HALT: A logical processor that continually checks to see if work is available will needlessly consume resources. The OS should HALT the inactive processor so that executions resources are freed up for the logical processor operating.
- Thread scheduling: The OS should allocate threads to one logical processor in each physical location before assigning additional threads to the alternate logical processors. This will allow threads to execute with full resources when they are available.

Licensing

HP server platforms based on Hyper-Threading have implemented the required BIOS changes to recognize the logical processors so that this information can be passed to the OS or application.

During system boot and initialization, the multiprocessor system BIOS records only the first logical processor on each physical processor in the system and records that information in the Multiprocessor Specification (MPS) table to preserve backward compatibility. This aids any legacy OS that only uses the MPS for determining system configuration and is not capable of using the alternate logical processors.

Next, the BIOS records each of the alternate logical processors into the Advanced Configuration and Power Interface (ACPI) table. This allows an OS that uses the ACPI table to see and schedule threads for all of the logical processors. It is critical that the BIOS record the first logical processor of each physical processor before recording any alternates.

Windows 2000 Server does not distinguish between physical and logical processors on systems enabled with Hyper-Threading Technology; Windows 2000 simply fills out the license limit using the first processors counted by the BIOS. For example, when Windows 2000 Server (4-CPU limit) is launched on a four-way system enabled with Hyper-Threading Technology, Windows will use the first logical processor on each of the four physical processors; the second logical processor on each physical processor will remain unused, because of the 4-CPU license limit. (This statement assumes that the BIOS was written according to Intel specifications. Windows uses the processor count and sequence indicated by the BIOS.)

However, when Windows 2000 Advanced Server (8-CPU limit) is launched on a four-way system enabled with Hyper-Threading Technology, the OS will use all eight logical processors. Although the OS will recognize all eight logical processors in this example, in most cases performance would be better using eight physical processors.

When examining the processor count provided by the BIOS, Windows 2003 Server distinguishes between logical and physical processors, regardless of how they are counted by the BIOS. This

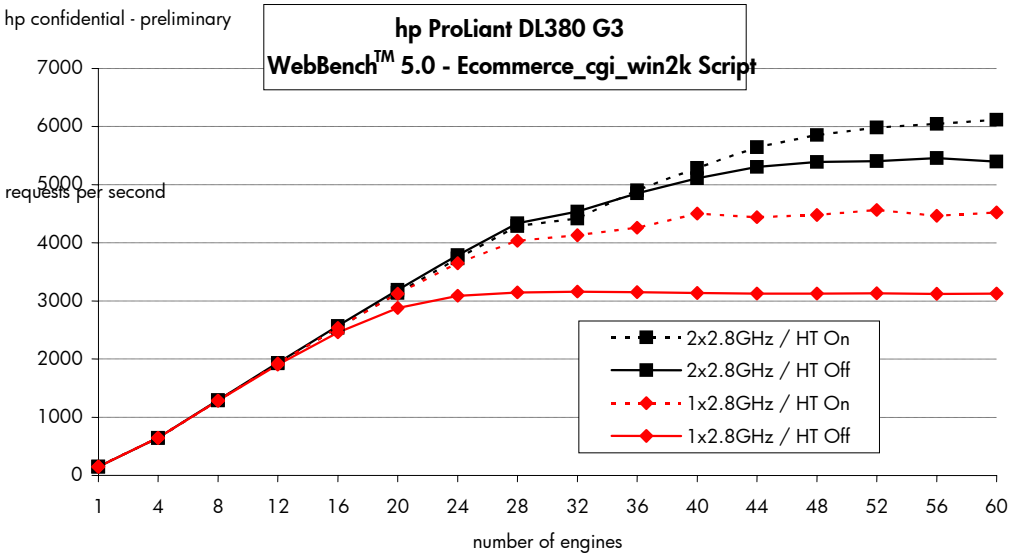
provides a powerful advantage over Windows 2000, in that Windows 2003 Server counts only physical processors against the license limit. For example, if Windows 2003 Standard Server (2-CPU limit) is launched on a two-processor system enabled with Hyper-Threading Technology, the OS will use all four logical processors. [1]

Performance gains

Intel Hyper-Threading Technology enables simultaneous multi-threading at the processor level. In the current implementation the two logical processors on each physical processor share most execution resources but maintain separate architectural states.

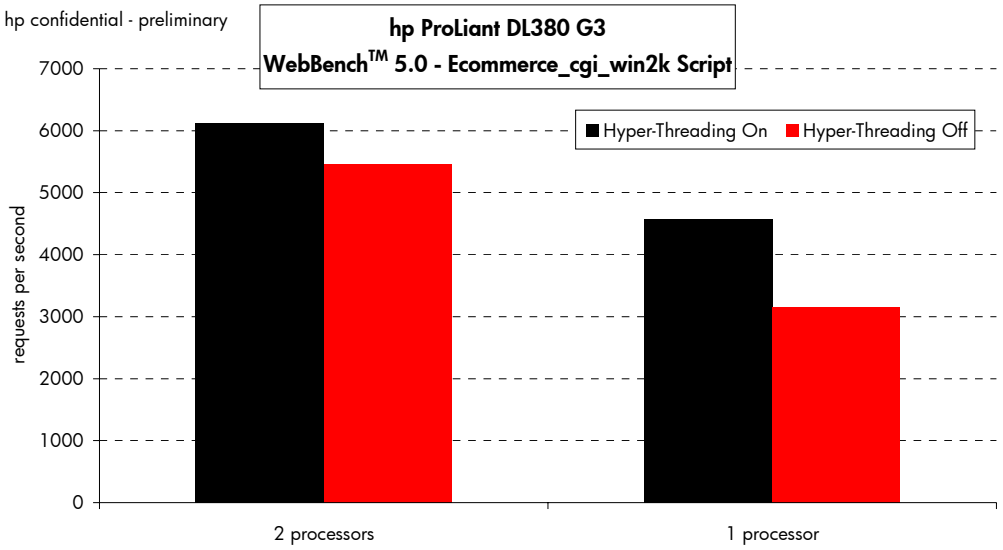
Tests run thus far in HP labs substantiate the expected performance gains for Hyper-Threading (Figures 2 and 3). For the standard workload of WebBench, the Hyper-Threading Technology showed a peak performance advantage of 12 percent in a dual processor server and a 44 percent performance advantage with a single processor. The differences observed in these cases are due entirely to the Hyper-Threading Technology, since the processors and systems were otherwise identical.

Figure 2. WebBench test results as a function of the number of physical clients (engines)



[1] *Microsoft Windows-Based Servers and Intel Hyper-Threading Technology, John Borozan, Microsoft Corporation, February 2002*

Figure 3. WebBench test results as a function of the number of processors



Performance results depend on many factors, of course, including installed memory, the application in use and the memory footprint it requires, as well as the number of simulated clients. Other tests results that were not ready in time for publication of this paper indicated that the performance gain was smaller in the on-line transaction processing application space, where the observed performance delta was between 5 and 14 percent.

Conclusion

Tests performed in HP labs indicate that Hyper-Threading fulfills much of its promise, as substantial performance gains were seen when using Xeon processors that incorporate Hyper-Threading. The gains reported from using Hyper-Threading ranged from as little as 5 percent in a multi-processor OLTP test to as high as 44 percent in a single-processor system running the WebBench benchmark test. This wide range emphasizes that performance is highly dependent on the type of application and other factors. As with any processor metric, actual performance benefits in the field will vary with system implementation choices such as installed memory, cache size, application type, and the memory footprint used by that application.

For more information

For additional information, refer to the HP ProLiant server website at www.hp.com/go/proliant.

Call to action

Send comments about this paper to: TechCom@HP.com.

TC030306TB, 03/2003