

Introduction

Hyper-Threading technology is a groundbreaking innovation from Intel that enables multi-threaded server software applications to execute threads in parallel within each processor in a server platform. The Intel® Xeon™ processor family uses Hyper-Threading technology, along with the Intel® NetBurst™ microarchitecture, to increase compute power and throughput for today's Internet, e-Business, and enterprise server applications. This level of threading technology has never been seen before in a general-purpose microprocessor. Hyper-Threading technology helps increase transaction rates, reduces end-user response times, and enhances business productivity providing a competitive edge to e-Businesses and the enterprise. The Intel® Xeon™ processor family for servers represents the next leap forward in processor design and performance by being the first Intel® processor to support thread-level parallelism on a single processor.

With processor and application parallelism becoming more prevalent, today's server platforms are increasingly turning to threading as a way of increasing overall system performance. Server applications have been threaded (split into multiple streams of instructions) to take advantage of multiple processors. Multi-processing-aware operating systems can

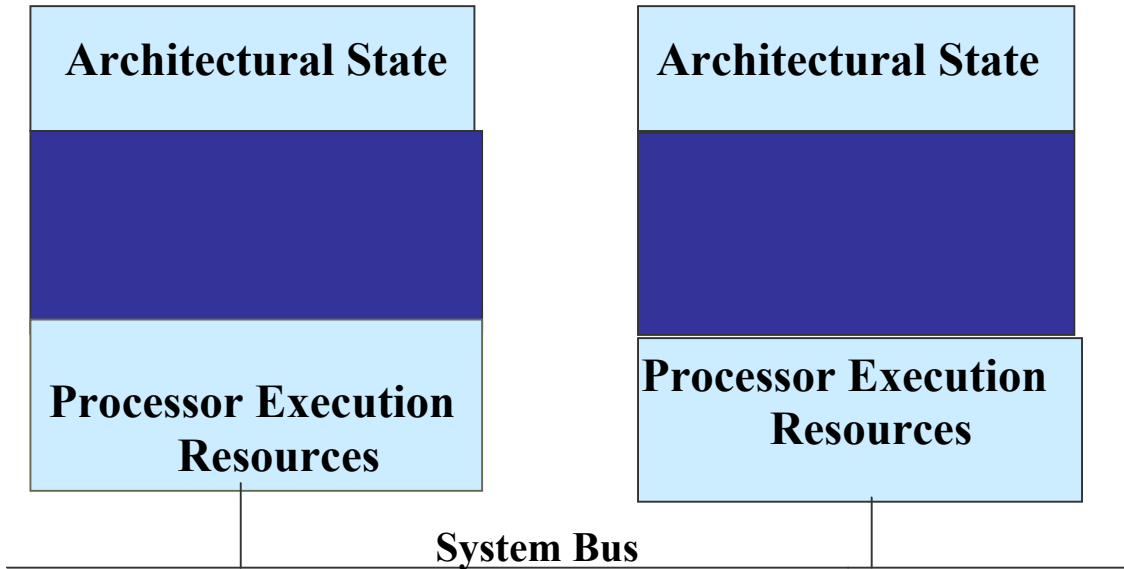
schedule these threads for processing in parallel, across multiple processors within the server system. These same applications can run unmodified on the Intel® Xeon™ processor family for servers and take advantage of thread-level-parallelism on each processor in the system. Hyper-Threading technology complements traditional multi-processing by offering greater parallelism and performance headroom for threaded software.

Overview of Hyper-Threading Technology

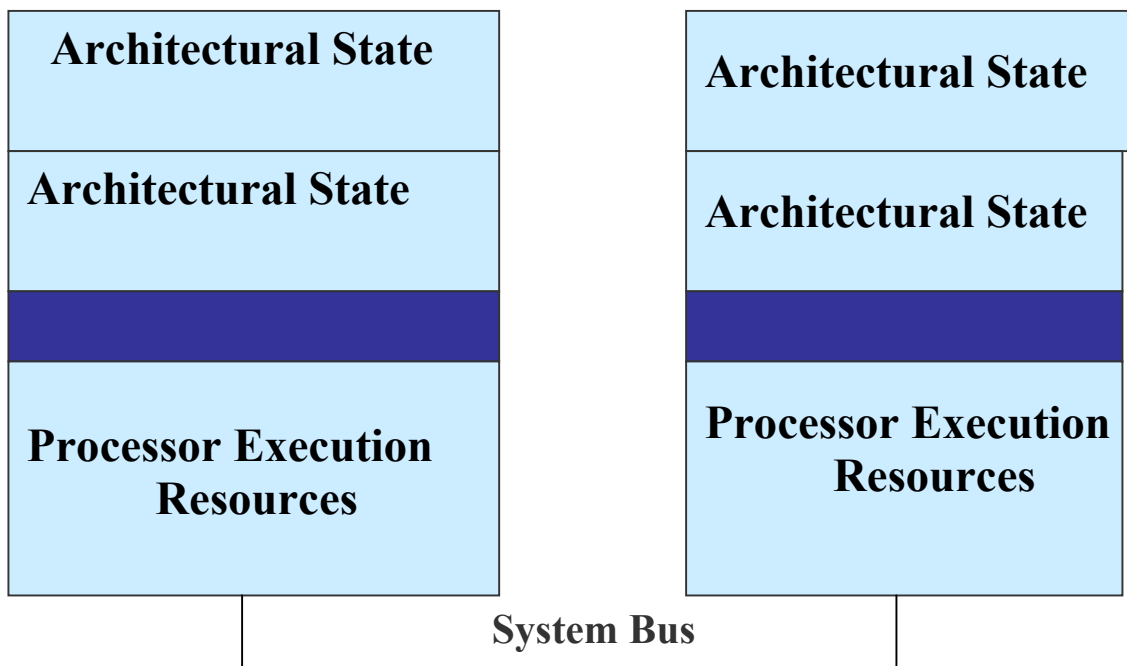
Hyper-Threading technology is a form of simultaneous multi-threading technology (SMT), where multiple threads of software applications can be run simultaneously on one processor. This is achieved by duplicating the architectural state on each processor, while sharing one set of processor execution resources. The architectural state tracks the flow of a program or thread, and the execution resources are the units on the processor that do the work: add, multiply, load, etc.

Dual-processing (DP) server applications in the areas of Web serving, search engines, security, streaming media, departmental or small business databases, and e-mail/file/print can realize benefits from Hyper-Threading technology using Intel® Xeon™ processor-based servers.

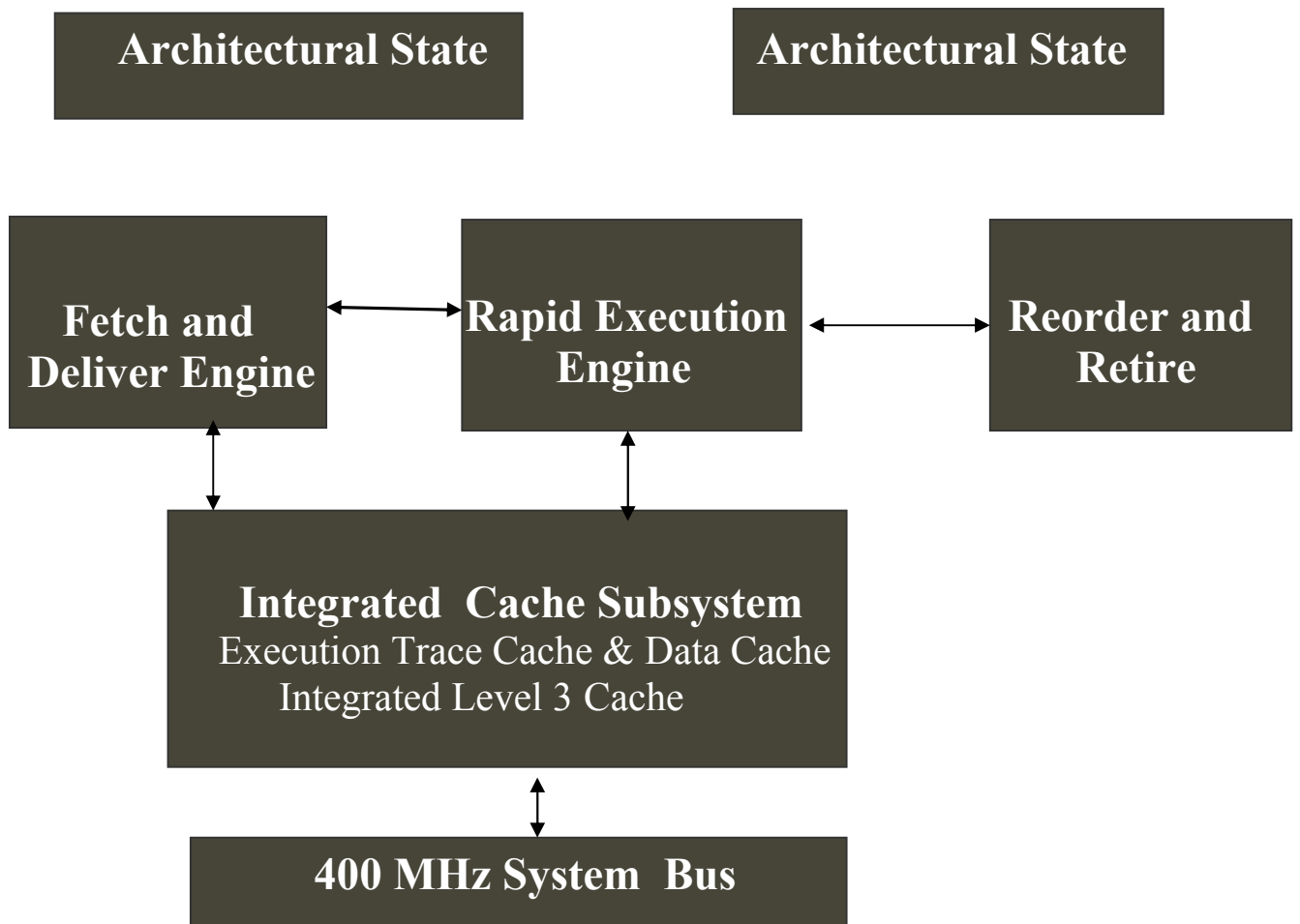
Traditional Multiprocessing System



A Dual Processor System With Hyper-threading Technology



Microarchitectural Details



With Hyper-Threading technology, the execution resources on the Intel®Xeon™ processor family are shared by two architectural states, or two logical processors. The majority of the execution resources are in the ***Rapid Execution Engine***, which process instructions from both threads simultaneously. The ***Fetch and Deliver engine*** and ***Reorder and Retire block*** partition some of the resources to alternate between the two threads.

Fetch and Deliver Engine:

The Fetch and Deliver engine alternates between fetching instructions from one logical processor and the other, and sends these instructions to the Rapid Execution Engine for processing. At the Level 1 Execution Trace Cache, one line is fetched for one logical processor, and then one line is fetched for the other logical processor. This continues, alternating back and forth, as long as both logical processors need to use the Execution Trace Cache.

Rapid Execution Engine:

At the Rapid Execution Engine, both logical processors execute simultaneously. The Rapid Execution Engine takes instructions from the instruction queues and sends them to the execution units as fast as it can. The instructions are selected based only on dependencies and availability of execution units. The instructions may be

selected out-of-order, meaning that later instructions that are independent can be scheduled before earlier instructions. The schedulers simply map independent instructions in the instruction queues to available execution resources.

Integrated Cache Subsystem:

The Integrated Cache Subsystem delivers data and instructions to the processor core at a high speed with larger cache lines than previous-generation processors. Because the Integrated Cache Subsystem is clocked at the same rate as the processor core, as faster processors are released, the cache speed can increase correspondingly providing high-speed access to key data. The larger cache line sizes also decrease average cache misses. A large, third-level integrated cache is available only on the Intel®Xeon™ processor MP for 4-way and above server platforms. This additional cache space is available for larger instruction and data sets, and significantly reduces the average memory latency, which improves performance for mid-range and high-end server applications. The caches are shared by both logical processors and are designed to minimize potential cache conflicts through a high level of set-associativity, which helps ensure data is well retained in the caches.

Reorder and Retire Block:

The Reorder and Retire block takes all the instructions that were executing out-of-order, and puts them

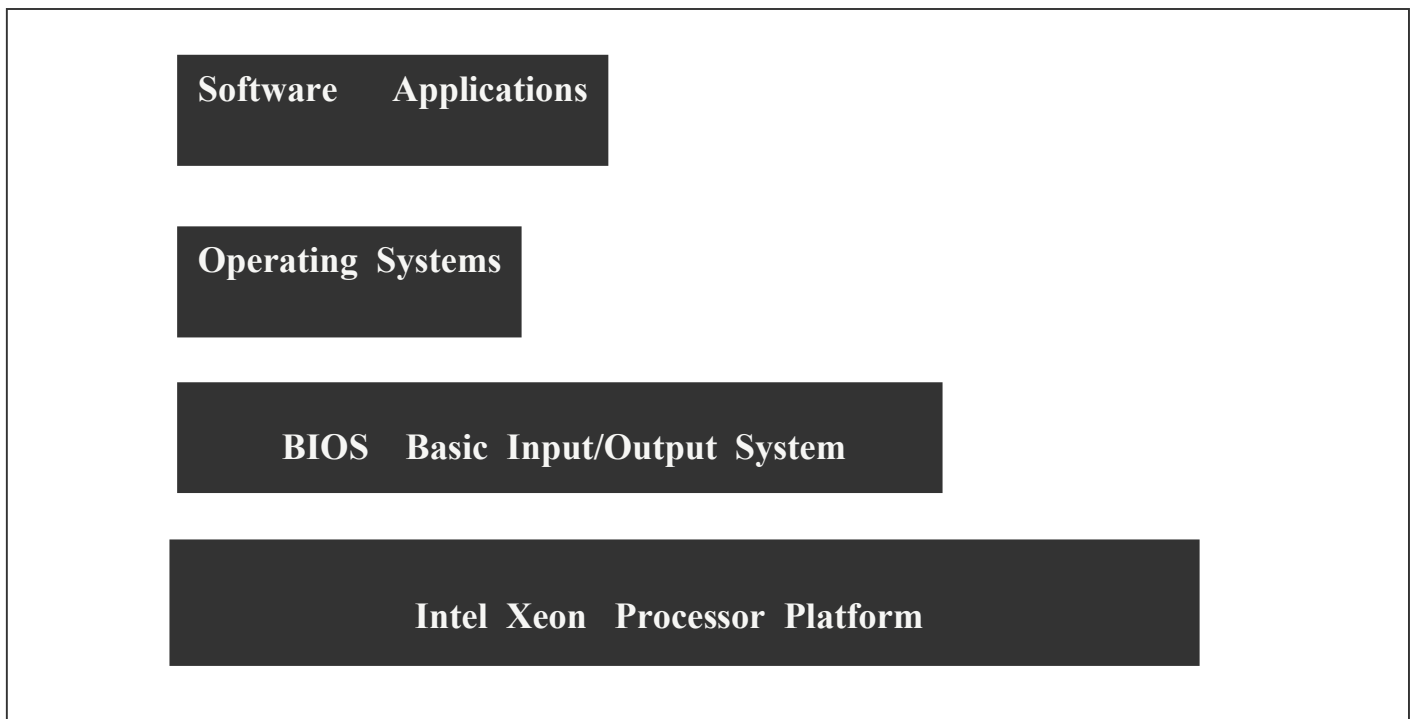
back into program order, then commits the state of those instructions in program order. Instruction retirement alternates between logical processors. Instructions are alternately retired for one logical processor, then the other.

400 MHz System Bus:

The 400 MHz System Bus is designed to increase the throughput of multi-processing and multi-threaded server applications and provide the necessary bandwidth for Hyper-Threading technology when accessing system memory. It uses signaling and buffering schemes that allow for sustained 400 MHz data transfers providing up to 3.2 GB/s bandwidth, which can be up to four times the previous-generation MP processor. When one of the logical processors cannot find the data it needs in the Integrated Cache Subsystem, then the data must be transferred over the system bus from memory.

How Server Platforms Use Hyper-threading Technology

Platform components of Intel® Xeon™ processor family for servers include the processor and other system hardware components, BIOS, operating system, and software applications



Server platforms based on the Intel®Xeon™ processor family have implemented the necessary changes in the platform BIOS in order to recognize the logical processors.

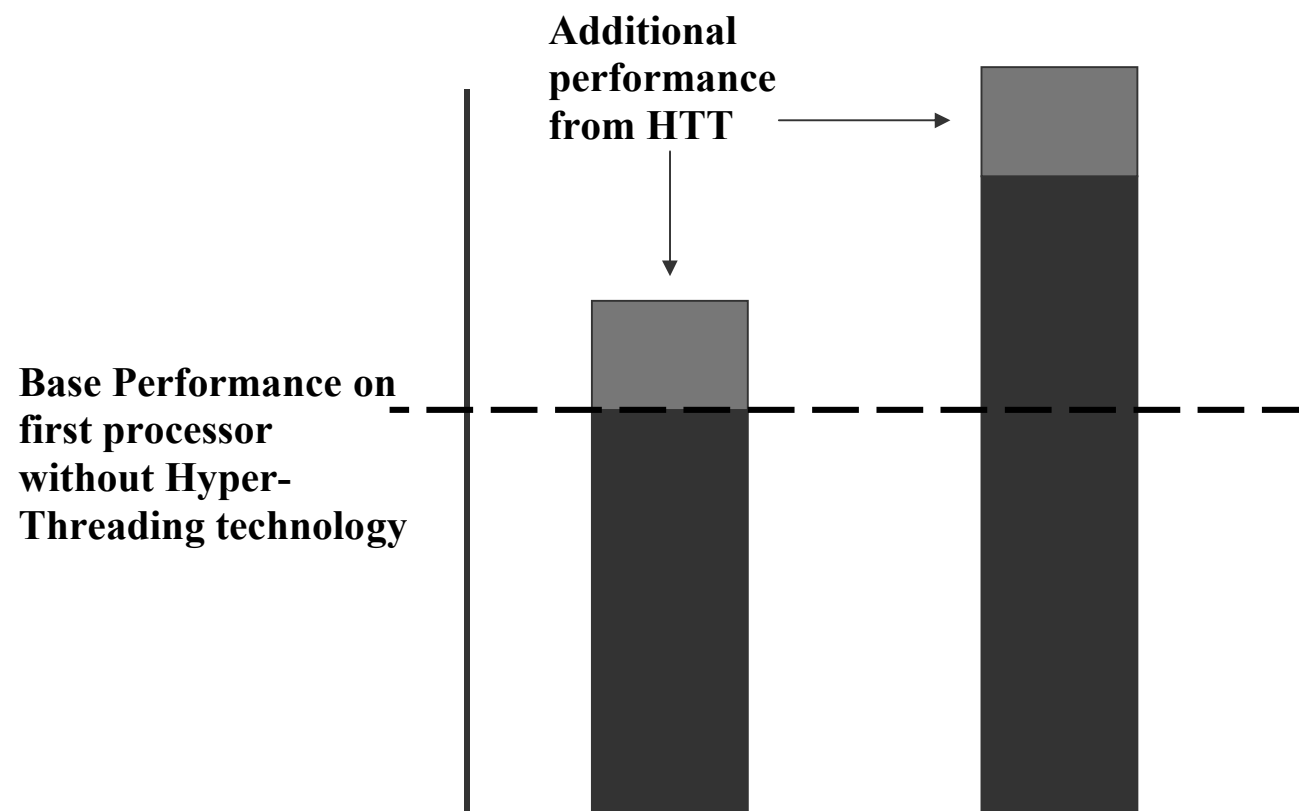
Within each Intel®Xeon™ processor in the system, the two logical processors appear to the BIOS and multi-processor-aware operating system (OS) as processors available to the system and software applications. During the multi-processor system initialization process (also called system boot), the BIOS counts and records the number of logical processors available in the system. The BIOS records only the first logical processor on each physical processor into the MPS (Multi-processor Specification) table to preserve backward compatibility with legacy operating systems. These legacy operating systems will use the MPS table only, and will not recognize the second logical processor on each physical processor. They will work as if Hyper-Threading technology is not there. The BIOS then records two logical processors for each physical processor into the ACPI (Advanced Configuration and Power Interface) table. Operating systems that use the ACPI table can then proceed to schedule threads onto the logical processors that the software license allows the system to use. The BIOS and OS may report only the number of logical processors available in the system.

The Intel®Xeon™ processor family with Hyper-Threading technology is fully backward compatible with existing operating systems and applications. That is, legacy multi-processor-aware operating systems can run unmodified on Intel®Xeon™ processor-based platforms. Some of these legacy operating systems (such as Windows NT*) may not recognize the second logical processor and may not take advantage of Hyper-Threading technology, but are compatible with the Intel®Xeon™ processor family

and will run unchanged. Today's current operating systems (such as versions of Windows* 2000 Server, Linux* and Novell NetWare*) can recognize two logical processors and can utilize Hyper-Threading technology, depending on OS license configurations. Many of the newer operating systems, either released or soon to be released (such as versions of Microsoft Windows* .NET Server, certain distribution versions of Linux, and Novell NetWare), are expected to include further optimizations for Hyper-Threading technology.

Performance with Hyper-Threading Technology

Applications that exhibit good threading methods and scale well on multi-processor servers today are likely to take advantage of Hyper-Threading technology. The performance increase seen is highly dependent on the nature of the application, the threading model it uses, as well as system dependencies.



Reduce Resource Competition

A platform with Hyper-Threading Technology-enabled processor(s) will automatically process two threads on each processor. Some code modules, however, might compete for shared resources, reducing the performance of the Hyper-Threading Technology processor. In a multi-processor system with Hyper-Threading Technology-enabled processors, the OS should dispatch code to different physical processors before scheduling multiple threads on the same processor. This kind of scheduling will keep resource competition to a minimum.

Spin-wait loops are an example where code written for a multi-processor (MP) environment might compete for resources in a Hyper-Threading Technology environment.

Spin-Wait Loops in MP Environment

In multi-processor environments, a thread might sit in a spin-wait loop while waiting for release of a lock that another thread holds. Until the lock is released, the thread in the loop will execute the loop as fast as it can. A simple loop might consist of a load, compare, and a branch else sleep(0). An efficient execution of such a loop by physical processor 0 can include simultaneous loads, compares, and branches - an endless consumption of the processors cycles

until the lock is released. In a multi-processor environment, one processor does not impact the resource availability to a code thread running on another processor.

| Processor 0 | Processor 1 |
|--|-------------|
| Spin wait: Try to get the lock If fail, Jump to spin wait | |

Spin-Waits in Hyper-Threading Technology

With Hyper-Threading Technology, spin-wait loops can delay the completion of one thread when another thread holds a lock, because the looping thread is repeatedly scheduled and executed, consuming vital shared resources in its loop. Considering our simple load, compare, and branch loop, logical processor 0 keeps executing its loop as fast as it can, consuming shared resources and clock cycles that logical processor 1 needs to complete execution of its thread. This effect impacts performance of multi-threaded code in a Hyper-Threading Technology processor.

Use the PAUSE Instruction to Optimize Code

Intel recommends that software writers always use the PAUSE instruction in spin-wait loops.

Starting with the Intel® Pentium® 4 processor, this recommendation was made for the benefit of power savings. Executing a PAUSE in a spin-wait loop forces the spin-wait to finish one iteration of the loop before starting the next iteration. Halting the spin-wait loop reduces consumption of non-productive resources - and thus power.

With a Hyper-Threading Technology processor, using the PAUSE instruction optimizes processing of two threads. Pausing the spin-wait loop frees more resources for the other processor, allowing faster completion of its thread.

Business Benefits of Hyper-Threading Technology

Hyper-Threading technology can result in many benefits to e-Business and the enterprise:

- Improved reaction and response times for end-users and customers
- Increased number of users that a server system can support
- Handle increased server workloads
- Higher transaction rates for e-Businesses
- Greater end-user and business productivity
- Compatibility with existing server applications and operating systems
- Headroom to take advantage of enhancements offered by future software releases
- Headroom for future business growth and new solution capabilities