Research on a system performance with parallel modes of execution in a multiprocessor computer system with Hyper-Threading Technology

Ognjan Nakov, Stefan Stojchev

Abstract: *This paper represents a research into the new Hyper-Threading Technology (TH) being introduced with the Intel Xeon family processors. Many applications expect maximum efficient exploitation of the hardware capability in the computer configuration. That depends on the ability of the operating system to manage concurrent executions of multiple instruction streams. The object of this paper is to provide results, analyses and conclusions from a research into the mechanism for managing threads of execution and common system performance.*

Key words: *Operating System, Hyper-Threading Technology (HT).*

INTRODUCTION

The Hyper-Threading technology was released from Intel in the first half of the year 2002. The Windows operating systems immediately implemented the support of the HT from Windows2000 to all nowadays 32-bit versions of the Windows operating family. Hyper-Threading (HT) is a microprocessor simultaneous multithreading technology that supports the concurrent execution of multiple separate instruction streams, reffered to as threads of execution, on a single physical processor. In other words if the system is HT, each physical processor hosts two threads of execution. The main difference between the execution environment provided by the HT processor, compared with that provided by two traditional single-threaded processors (it is shown in figure 1), is that HT shares certain processor resources: there is only one execution engine, one on-board cache set, and one system bus interface. This means that the logical processors on an HT processor must compete for use of these shared resources. As a result, an HT processor will not provide the same performance capability as two similarly equipped single-threaded processors. It is important to note that the two logical processors on an HT processor are treated equally with respect to access to the shared resources.

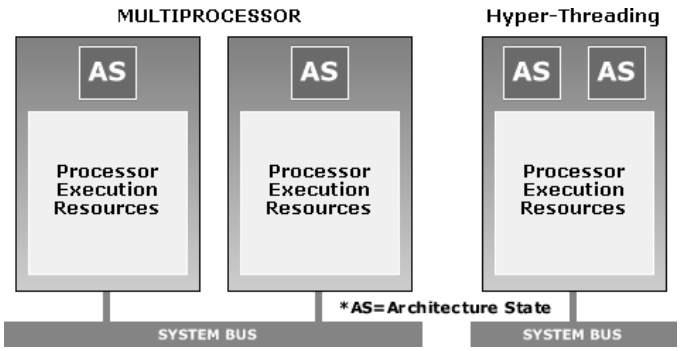


Figure 1. The main differences in architectures.

This research has the purpose to investigate the fundamentals in the utilizations of the logical HT processors, execute tests in several modes, analyze the results generated from the tests and make conclusions.

About test configurations and implemented ideas in the testing modes.

Software environment – The operating system is Windows 2000 Advanced Server, the most stable operating system from the Microsoft family yet. Service Pack for the operating system is SPIV. Programming language for developing test application is C++ - provides powerful and fast execution. Integrated Development Environment is VC++ 6.00.

Test hardware configuration – The system hardware configuration is:

CPU: Intel® Xeon™ CPU 2.00 GHz, SMT support – 2 units, Generation – 7th (7x86), Model information – P4 Prestonia Xeon, Hyper-Threading Technology – supported, L2 On-board Cache – 512 KB.

System MainBoard: Vendor – Dell Computers, MultiProcessor support – 2 units.

RAM: Installed memory – 768 MB DIMM Synchronous DDR-SDRAM 3*256.

Figure 2 shows the situation with the Windows 2000 Advance Server running on our test HT-enabled system (described already above) that has two physical HT processors. The numbers indicate the sequence in which the logical processors are listed by the BIOS. The only correct way for that is to list the first logical processor on each of the physical HT processors before listing any of the second logical processors. In this figure logical processors are listed by the BIOS in the recommended sequence. The operating system will attempt to utilize the logical processors in the same sequence as the BIOS listed them. The sequence in which logical processors are started is really very important because the operating system scheduler for managing threads always will utilize the next numbered inactive logical processor and that is its only one optimization. This causes a decreased system performance in comparison with other scheduling situations and it can be seen on this research.

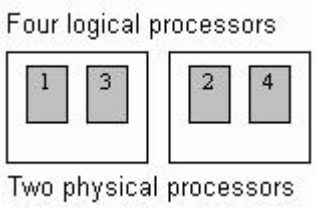


Figure 2. The test system with Two Physical HT Processors

Test modes - With an understanding of how application threads and process utilize the shared resources on the HT processors, the tests in this research will take place in five different modes. Each of them represents specific circumstances of the execution process. Notify that it is assumed that the whole amount of executable instructions in each mode stays unchanged. In other words the "work" remains constant for every different test mode. The main difference between the five modes is the way of managing the execution process. This is done using threads which are exclusively forced to be executed on a specific logical processor on an HT physical processor. All versions of the Windows operating family submit system functions for doing this.

First test mode: In this test mode all the threads are started on the first logical processor on the first HT physical. This simulates computer configuration with one traditional single-threaded processor. Note that this case is the base mode. Afterward the resulting system performance from this mode will be compared with those from the other modes.

Second test mode: In this test mode the operating system is absolutely free to allocate the streams of instructions grouped in separate threads to logical processors. This mapping depends only on the operating system. Assuming that no processor affinity has been set, the operating system scheduler is free randomly to schedule the next available thread or stream of instructions to any of the inactive logical processors.

Third test mode: The specific is that the threads are forced to be executed only on the first and the second logical processors, that is, one from each physical processors. It is assumed that each logical processor (from the mentioned above) executes the half of the instructions. This simulates computer configuration with two traditional single-threaded processors. Figure 1 shows the utilization of the logical processors in this test mode. Note that in all of the figures in this paper, the logical processors shown as shaded are the ones that are utilized by the operating system.

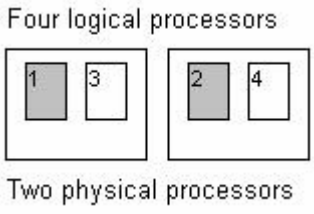


Figure 3. The utilization in mode 3

Forth test mode: Here the threads are forced to be executed only on the first and the third logical processors, that is, the both logical processors on the first physical processor. It is assumed that each logical processor executes the half of the instructions. Figure 2 shows the utilization of the logical processors in this mode. With an understanding the main idea of the HT it is expected that this test mode will generate a decreased system performance than the performance generated in the third test mode. For sure this will be caused by the competition for the shared resources on the first physical processor.

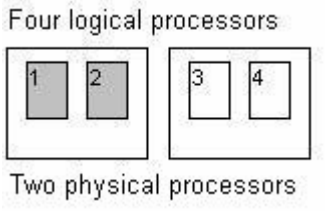


Figure 4. The utilization in mode 4

Fifth test mode: In this test mode each logical processor is forced to execute a quarter of the amount of all instructions. Figure 3 shows the utilization of the logical processors. It is logical (more active processors – more increased system performance) to make a guess that this mode should generate the highest possible system performance.

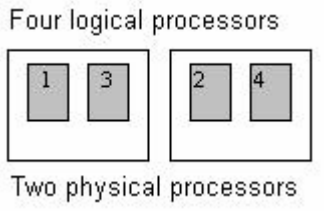
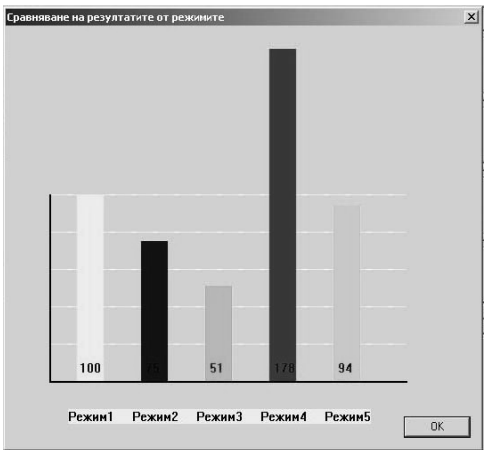


Figure 5. The utilization in mode 5

RESULTS FROM TEST MODES AND CONCLUSIONS.

Results – Figure 6 shows the results generated from all five test modes (mentioned before). It is very important to note that the picture represents the correlation in times needed for each test mode to finish the same constant work. The base mode is the first test mode. Taller colored bars means decreased system performance and the need of more time for the test mode to finish. The results are:



First test mode: 100 relative units. The base test mode.

Second test mode: 75 relative units. Obviously the operating system utilizes all four logical processors (it can be seen on Windows Task Manager Performance Tool) and the resulted performance is better than the resulted performance in the first test mode but not as better as it should be. The result generated in the third test mode points the optimal performance for Windows 2000 Advance Server on this hardware configuration. And the first conclusion is: **the Windows 2000 Advance Server doesn't provide optimal working mechanism for effective managing the separate threads of execution**.

Third test mode: 51 relative units. The optimal performance for Windows 2000 Advance Server on this hardware configuration. There is a small competition between the two active logical processors for the common resources. In that case this is the system bus only.

Forth test mode: 178 relative units. The most decreased system performance. In this case two logical processors work just like in the third test mode but the resulted performance is surprisingly decreased. The system performance is decreased with over 300% in comparison with the result in the third test mode. The reason for that is that the

active logical processors are situated on the same physical HT processor and they ferocious compete for the common resources. In that case the common resources are the on-board processor cache and the system bus. **The conclusion is: the on-board processor cache is a much more considerable critical resource than the system bus**.

Fifth test mode: 94 relative units (There is a big surprise). In this test mode all four processors are utilized and the difference between the generated system performance in this case and the generated system performance for the first test mode is almost unsignificant. In fact there is almost no difference between the two cases with the only one and the four simultaneous working logical processors. So what is the point of Hyper-Threading Technology? Is this a next commercial trick for Intel?!

Other Conclusions - Windows application software should run unmodified, and without error, on HT-enabled systems. The performance gain varies depending on the application. Microsoft tells that in general, multithreaded Windows applications perform better when running unmodified on an HT processor than they do on a similarly equipped single-threaded processor. But this research points different results! To use completely the hidden power of HT technology, programmers are supposed to modify their applications to support features such as - identifying the presence of HT processors and improving application performance on HT systems. There are some helpful instructions:

Application Identification - On HT-enabled systems, each logical processor is treated as an individual processor by the operating system and is represented by a bit in the system affinity mask. Applications must identify the presence of HT to perform HT-aware enforcement of per-processor licensing rules or to create an HT-aware execution environment for the application processes and threads. To perform these types of functions, applications use that system processor affinity mask. The Windows operating systems support a few system functions for working with the system processor mask. Some of them are: GetProcessAffinityMask(), SetThreadAffinityMask(), SetThreadIdealProcessor(). Their names show exactly what they do. Another new API function has been included only in all versions of the Windows Server 2003 family. It is called GetLogicalProcessorInformation(). The data returned can be used to create a list that relates the bits in the system processor affinity mask to the logical and physical processors in the system. It is important to note that application software that was released before the introduction of HT requires modification to support the identification of HT processors.

Improving application performance – To optimize the application performance benefit on HT-enabled systems, the application should ensure that the threads executing on the two logical processors have minimal dependencies on the same shared resources on the physical processor. With an understanding of how the application threads and processes utilize the shared resources on an HT processor, setting processor affinity to minimize competition for these system resources can increase much more the application performance. A good tip for the programmers is: where an application has threads that produce data and threads that consume data, setting affinities so that consumer/producer thread pairs run on the logical processors of the same physical processor should improve performance. This configuration allows the threads to share cached data and to overlap operation.

Idle loops - When a processor in a system running the Windows operating system has no work to do, it enters the idle loop. If the first logical processor on an HT processor is executing instructions in the idle loop, that is, if it is not doing any real work, it is competing for shared resources, which degrades the performance capability of the second

logical processor on the same physical processor. The result of this is decreasing the performance of the whole system. For that reason it is recommended to utilize a **YIELD (PAUSE)** and **HALT** instructions. HALT instruction makes the logical processor to be halted and it no longer executes instructions and no longer competes for shared resources. YIELD, which is functionally equivalent to the PAUSE instruction on the Xeon HT-processors, causes the logical processor to pause for a short period of time (approximately 50 clock cycles), and allows the other logical processor access to the shared resources on the physical HT processor.

FINAL CONCLUSION – The Windows2000 Advance Server operating system doesn't provide optimal system scheduler for managing threads of execution. This should be done "manually" by specific program codes in the application having in mind the main idea of HT technology. There is a great need for an absolute optimal operating system scheduler and the next research will be a step in this direction.

REFERENCES

[1] Intel Hyper-Threading Technology documentation – http://www.intel.com/technology/hyperthread/index.htm

[2] Microsoft Hyper-Threading Technology documentation – http://www.microsoft.com

[3] Microsoft Developer Network (MSDN).

[4] Introduction of the new Hyper-Threading Technology – www.setcom.com

[5] Kruglinski,D., G. Shepherd, S. Wingo. Programming Microsoft Visual C++ 6.0. Microsoft Corporation, USA, 1999

ABOUT THE AUTHORS

Assoc. Prof. Ognjan Nakov, PhD, Department of Computer Systems, Technical University of Sofia, Phone: +359 02 9653613 , E-mail: nakov@vmei.acad.bg

Stefan Stojchev, MS, Department of Computer Systems, Technical University of Sofia, Phone: +359 0887 153991, E-mail: stef_stoichev@abv.bg

FEEDBACK

For notes and asks for the open code testing application used in this research please write to: stef_stoichev@abv.bg