Principled Technologies

White Paper
June 2004

Effects of Intel® Hyper-Threading Technology on the Response Time of Business Office Applications

For Intel Corporation

Executive summary

Intel® Corporation (Intel) commissioned Principled Technologies SM (PT) to examine the effects of Intel's Hyper-Threading (HT) Technology on the response time—how quickly applications respond to user commands—that PC users experience when running multiple common business applications. By providing the ability to run two instruction streams simultaneously, HT Technology has the potential to improve the response time of systems running two or more applications at the same time. We investigated how well HT Technology delivered on that potential by examining work scenarios in which users would commonly be running and switching among multiple applications. Business applications are among the most common programs in use in enterprises today, and Microsoft's Office 2003 is the most current version of the office suite that is in the broadest usage today, so we used that product as the basis for our investigations and tests.

We found that HT Technology frequently improved a critical aspect of PC multitasking behavior: the response time of foreground applications. We discussed typical office application usage with IT managers and conducted some additional informal research in application usage. We then examined different scenarios in which a user would launch an application or perform a time-consuming function in one application (the background) and then, while waiting, switch to another application (the foreground) to perform another task. In such situations, we found that HT Technology typically improved the response time of the foreground application. This response-time improvement typically occurred with little to no penalty on the time the background task took to complete. In some cases, HT Technology even improves overall system throughput, the amount of work the system performs in a given unit of time.

Key findings

- Intel's HT Technology frequently improves the response time that users experience when running multiple applications simultaneously.
- The response-time improvement from HT Technology typically occurs with little to no penalty on the time that background tasks take to complete.
- In some cases, HT Technology can also improve overall system throughput.
- The response-time improvement from HT Technology in multitasking situations has the potential to be much greater than the throughput improvement.

To ensure that our tests did not reflect only a particular vendor's systems, we set up a test bed of three PCs from three different top computer vendors. We did not aim for the fastest systems available, but instead specified the following configuration:

- Intel Pentium® 4 processor with HT Technology, 2.80GHz
- 512MB of RAM
- 40GB hard disk
- on-board graphics

Vendors typically configure similar system models slightly differently, so our test systems have similar but not identical configurations. Because of those differences and because our goal is to focus on HT Technology, not to compare individual system performance, we refer to these PCs as System A, System B, and System C. Appendix A provides configuration details for all three systems. We used the BIOS of each system to enable and disable HT Technology.

Our tests showed that HT Technology improved multitasking response time on all three systems on multiple tests of multiple application scenarios.

We examined multitasking application scenarios that our research and experience suggested were like those many real users perform today with standard office applications. After identifying scenarios we thought would be representative and useful for this paper, we manually tested the scenarios by launching combinations of foreground and background application tasks, then timing the foreground response with a stopwatch. We then chose three representative scenarios that we discuss in detail below.

We wanted to be able to test our scenarios repeatedly and accurately on our test systems, so in addition to hand-timing the application functions we created scripts to automate and time the application functions. Though such scripts unavoidably add a small amount of overhead to the application functions and do not perfectly represent the hand-timed scenarios, they provide reasonable, automated, and repeatable mechanisms for running the scenarios on multiple systems. We created the scripts in IBM's Rational Visual Test 6.5 (more on the scripts below). We use the times from these scripts in our analysis below.

As Figure 1 shows, HT Technology delivered significant improvements for all three of our multitasking scenarios, with an average response-time improvement of 55% across all three scenarios and all three systems. (We discuss those scenarios in more detail in the next section.)

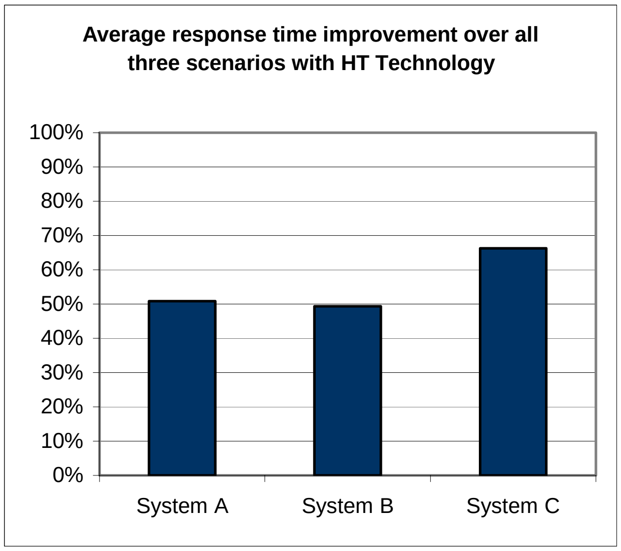


Figure 1: Average response time improvement over all three test scenarios with HT Technology. Improvement is the percentage faster the test applications responded to user commands.

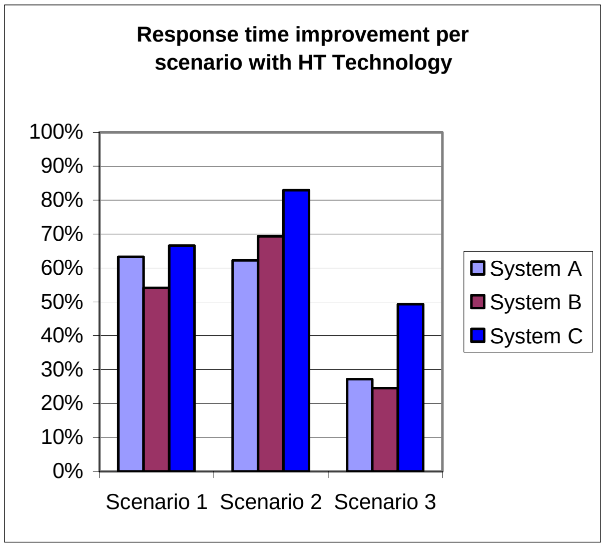


Figure 2: Average response time improvement for each of the three test scenarios with HT Technology. Improvement is the percentage faster the test applications responded to user commands.

Figure 2 shows the percentage of response time improvement for each system with each of the three different scenarios. Though the different systems showed different degrees of improvement in our scenarios, HT Technology delivered significant response-time benefits in all three scenarios on all three test systems.

In the following sections we discuss these scenarios (Application scenarios), some useful basics about HT Technology (HT Technology background), the results of our tests with those scenarios (Test results and analysis), and how we tested (Test methodology). In the appendices we provide details about the configurations of the test systems, explain how to manually execute the application functions in our scenarios, and discuss some issues in the developments of the scripts.

Application scenarios

We began the process of selecting application scenarios by discussing typical office application usage with IT managers. We also conducted some additional informal research in common office application usage. We examined different multitasking scenarios using Microsoft Office 2003 applications and such other common business programs as Adobe's Adobe Reader 6.0. We typically examined scenarios initially with HT Technology disabled, and we ran the scenarios manually, as users would. We monitored the processor usage of both the background and foreground tasks with the CPU Usage feature of the Windows Task Manager and the Windows Perfmon utility. We then ran those same multitasking scenarios manually with HT Technology enabled.

After examining different multitasking scenarios, we settled on three that we felt represented a range of common user activities in a variety of different common office applications.

In Scenario 1 a user is working on a PowerPoint 2003 presentation about the market for digital products of all types. The user wants to email the presentation to a group. To shrink the file prior to emailing it, she has instructed PowerPoint to compress its images, a task that takes a fair amount of time to complete. Rather than waiting for that task to finish, she uses Adobe Reader 6.0 to read a .PDF document about some aspects of the equipment and infrastructure necessary in a digital home.

Scenario 2 involves two different applications, Access 2003 and Outlook 2003. In this scenario a user who wants to examine a sales-analysis Access database opens that database and tells it to run all the analysis macros he has prepared to retrieve and format the data that interests him. While waiting for this task to complete, he decides to check his email and launches Outlook.

We chose our final example, Scenario 3, to represent tasks that are not as lengthy as those in the first two scenarios. In this case, the user first launches a Word document about media center PCs. After launching the document, the user decides to see how the document would look in Web layout. This operation causes a delay, so the user decides to launch an Excel spreadsheet that provides a very simple model of a call-management system.

For more details on how to execute each of these scenarios and the files they involve, see Appendix B.

HT Technology background

Before proceeding to the analysis of our test results, some HT Technology background is useful. Multitasking scenarios that typically benefit the most from HT Technology are those in which the background task appears to be using all or nearly all of the processing capacity of the system with HT Technology disabled. We use the term "appears" because even though performance-monitoring tools will correctly show the processor as being fully or nearly fully subscribed, in most cases multiple execution units within the processor will be idle. (Modern processors, such as the Intel Pentium 4 processor, contain multiple execution units, the parts of the processor that actually execute program instructions. Many of execution units in a processor are typically inactive at any given time.) By making a single processor appear to the operating system as two logical processors, HT Technology lets the operating system schedule additional work on the processor—by scheduling what it sees as the second processor. That extra work often ends up using additional—and otherwise idle—execution units.

The effect on system throughput—the amount of work the system performs in a given unit of time—of being able to keep more execution units busy can be significant. The effect on system response time in multitasking scenarios, however, has the potential to be much greater than the throughput improvement.

To see why, consider the following hypothetical example, first on a system without HT Technology. A user launches a background task that if running alone on the system would take 20 seconds to complete. The task appears to use 100% of the system's processing power for most of those 20 seconds. As users frequently do, rather than wait for the background task to complete, the user wants to keep working and so, after a wait of a few seconds, launches another application. The launch of that application if running alone on the system would take five seconds to complete, during which time it would normally use much of the system's processing power. Without HT Technology, in such cases the operating system frequently either does not schedule the second task until the first completes or at best allocates the second task such small amounts of processor time that it can accomplish little. Thus, the user cannot begin to work in the second, foreground application until the first, background application has completed its work and the second has had a chance to launch. The foreground task's response time—the time it takes before responding to the user's command to start—is over 20 seconds.

Consider the same example on a system with HT Technology. The user launches the same background application. This time, however, that application appears to be using all of only one of the two (logical) processors of the system; the other processor appears available. So, when, a few seconds later, the user launches the second (foreground) application, the operating system can schedule it on the second (logical) processor. The foreground application then launches almost as quickly as it would if running alone on the system. In this case, the foreground task might complete in six seconds, just a little more time than it would take if running alone. From the user's perspective the response time of that application is six seconds—a vast improvement over the more than 20-second response time without HT Technology.

Because HT Technology in this case is letting the system use otherwise idle processor execution units, it also typically would have little to no effect on the time the background task took to complete. So, the background task would still complete in roughly the same 20 seconds it would have required if running alone. The system's total throughput for those 20 seconds is greater with HT Technology than without it, because with HT Technology the system both finished the background task and launched the foreground application during the same time period.

During our investigation we encountered many cases in which HT Technology led to improvements on this scale. Our first two application scenarios serve as examples of the large response-time improvements possible. We also encountered situations in which HT Technology delivered smaller but nonetheless important and useful improvements in response time in the face of smaller background tasks. Our third application scenario is an example of the smaller but still significant HT Technology benefits available.

Finally, we must, of course, note that HT Technology does not always improve response time. When each of the application tasks running concurrently does not fully or nearly fully subscribe the processor, HT Technology may not improve response time.

Test results and analysis

As we noted earlier, we automated each of the three test scenarios with Visual Test 6.5 scripts that both perform and time the application functions in each scenario. In this section we examine the timings for each of the scenarios in turn and discuss the effects of HT Technology on the response time and any other relevant aspects of each scenario.

Figure 3 shows the results of our tests of our Scenario 1 Visual Test scripts. Each of the times we present in this table and in the corresponding tables for the other scenarios is the median of five runs, in seconds. We show three decimal places of timing results because the timers in our Visual Test scripts report in milliseconds. Because we're measuring response times, smaller times are better.

Test PC	**Foreground (PDF read)**		**Background (PowerPoint)**	
	HT On	**HT Off**	**HT On**	**HT Off**
System A	6.031	16.422	16.843	16.360
System B	7.344	16.016	16.938	15.688
System C	7.172	21.453	16.968	15.719

Figure 3: Median foreground and background test times for the Scenario 1 script on all three test systems. Smaller is better.

As these results (and Figure 2 above) show, HT Technology provides a dramatic improvement in foreground application response time on all three test systems. This improvement comes at a slight cost to the background application, which finishes on average a little over a second slower with HT Technology enabled. Despite this cost, the system's overall throughput during the entire time period is higher with HT Technology, because the system accomplishes both the foreground and the background work in about the same time the background task alone takes to complete without HT Technology.

To see why HT Technology improves response time so much in this case, examine Figures 4 and 5 below. Each of these figures shows the processor utilization of one of the two tasks running alone on a system without HT Technology enabled. (For these processor utilization measurements and for those in the rest of this paper, we used System A. The task times in these processor utilization charts will not match the times in the test results because we captured processor utilization of each task running alone with HT Technology disabled.)

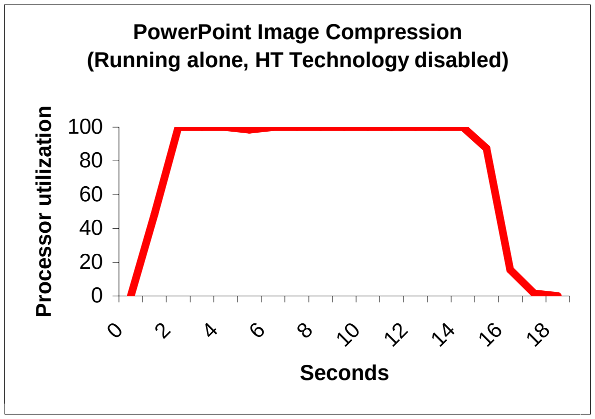


Figure 4: System processor utilization, without HT Technology, of the Scenario 1 background task running alone.

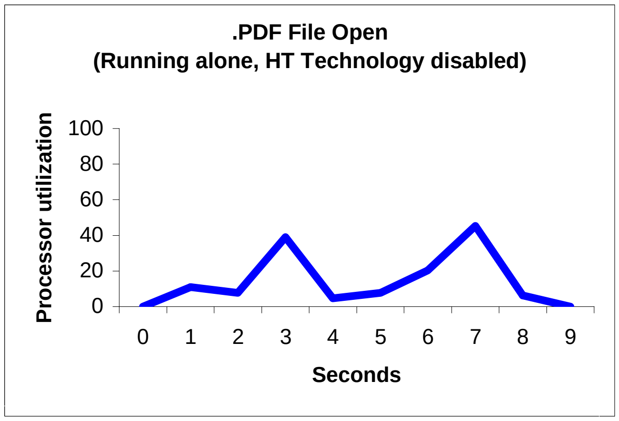


Figure 5: System processor utilization, without HT Technology, of the Scenario 1 foreground task running alone.

As Figure 4 demonstrates, the background task—the PowerPoint image compression—is so computationally intense that it uses 100% of the available processing power for most of the time it is running. Consequently, without HT Technology when the background task is executing and the operating system wants to schedule the foreground task, little processor time is available. Though the foreground task—opening the .PDF file—is nowhere near as demanding of the CPU, it nonetheless at two key points consumes about 40% of the system's processor and so needs significant system resources.

With HT Technology, the operating system appears to have a second (logical) processor available to run the foreground task. By running the foreground task on this second logical processor, the system is able to take advantage of the idle execution units in the processor and run the two tasks concurrently.

Thus, scenario 1 is an example of response-time improvement at a small cost in background task performance.

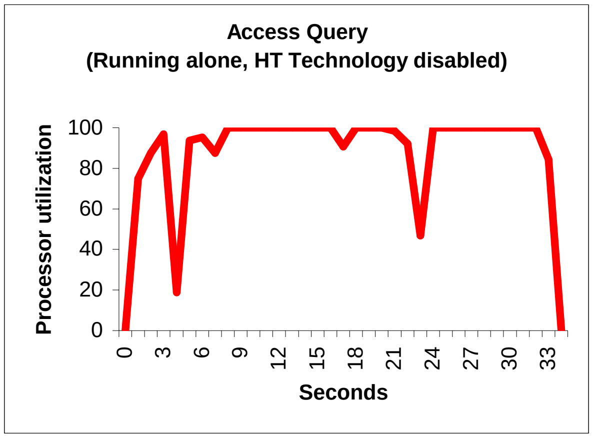
Scenario 2, as Figure 6 shows, is a case in which HT Technology delivers both vastly better response time and significantly better overall throughput.

Test PC	**Foreground (Outlook Startup)**		**Background (Access Query)**	
	HT On	**HT Off**	**HT On**	**HT Off**
System A	6.344	16.797	35.266	37.078
System B	4.828	15.734	31.921	42.313
System C	5.047	29.500	34.125	41.469

Figure 6: Median foreground and background test times for the Scenario 2 script on all three test systems. Smaller is better.

The response-time win for the foreground application is large and easy to see. The times for the background task are also significantly better with HT Technology enabled, ranging from a low of about a two-second improvement on System A to an over 10-second improvement on System B. As these times show, in this case the user's experience is in all ways better with HT Technology: the foreground task finishes sooner, the background task finishes sooner, and, consequently, the system's overall throughput is higher.

The key to understanding why HT Technology was able to improve performance in all areas again is in the processor utilization characteristics of the background and foreground application tasks. Figures 7 and 8 show the percentage of processor utilization over time for those two tasks.



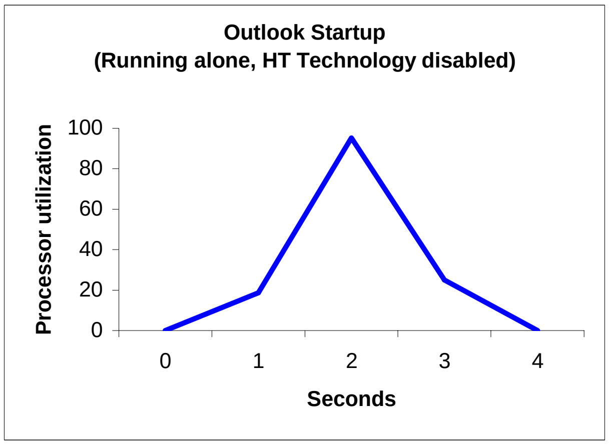


Figure 7: System processor utilization, without HT Technology, of the Scenario 2 background task running alone.

Figure 8: System processor utilization, without HT Technology, of the Scenario 2 foreground task running alone.

Figure 7 shows that the Access query background task, like the PowerPoint image-compression background task of Scenario 1, for much of its duration uses 100% of the processor on a system without HT Technology. Unlike that earlier background task, however, this one has multiple points during which the processor utilization dips below 100%. As Figure 8 shows, the foreground task spikes at one point to 100% processor utilization, but that point is brief. The foreground's processor utilization is over 20% only for a couple of seconds.

Without HT Technology, the background task is still so demanding that the foreground task ends up getting very little processor time. With HT Technology, however, the combination of the multiple dips in the processor requirements of the background and the relatively short duration of high processor usage of the foreground means that the system's processor needs to find available execution units only for a few short blocks of time to complete the foreground. In our tests the system's Pentium 4 processor with HT Technology was able to keep running the background activity at roughly full speed while also handling the foreground work. The result was, as our tests show, both dramatically better response time and better throughput.

As we noted in the Application scenarios section, we did not want to focus only on scenarios with relatively long background tasks. We also wanted to examine scenarios with shorter foreground tasks. Scenario 3 provides an example of this situation. Figure 9 shows the results of our tests of our Visual Test scripts of that scenario. Here, the response time benefit of HT Technology is a large percentage—an average of about 34% across the three systems—and a small but definitely noticeable amount of time, on average over a second and a half. The effect on the performance of the background task varies by system, with two systems (System C and, to a lesser degree, System A) showing an improvement in background times with HT Technology, while one system (B) showed a faster background time without HT Technology.

Test PC	Foreground (Excel Spreadsheet Open)		Background (Word Document Open)	
	HT On	HT Off	HT On	HT Off
System A	3.563	4.891	31.203	31.828
System B	3.313	4.391	37.046	31.140
System C	3.563	7.031	39.422	47.906

Figure 9: Median foreground and background test times for the Scenario 3 script on all three test systems. Smaller is better.

In this scenario, as you can see in Figures 10 and 11, the background task at some points does not consume all of the system's processing power. The processor demands of the foreground task are relatively low, with that task wanting more than 20% of the processing capacity only for about a second.

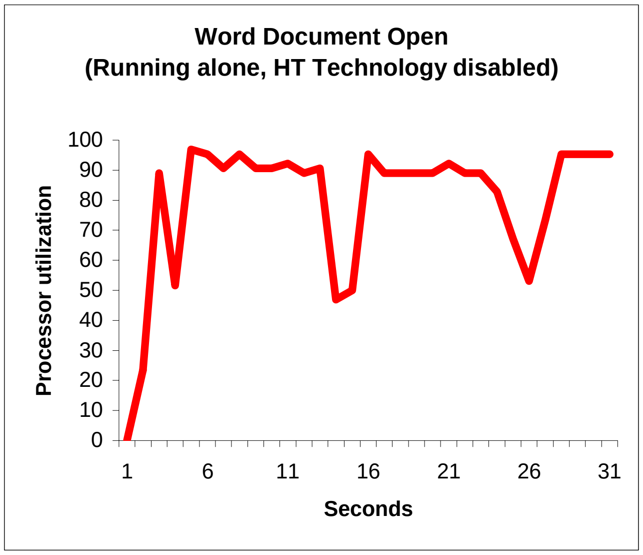


Figure 10: System processor utilization, without HT Technology, of the Scenario 3 background task running alone.

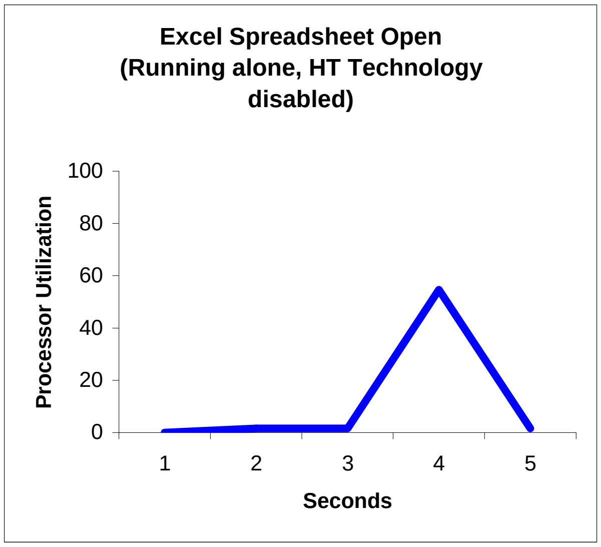


Figure 11: System processor utilization, without HT Technology, of the Scenario 3 foreground task running alone.

HT Technology does not show as large a win in this scenario as in the previous scenarios for two reasons. First, the combined processor utilization of the two applications is generally lower than in the previous scenarios. In addition, these two tasks are easier to schedule together than the background and foreground tasks of the previous scenarios. Put differently, when running without HT Technology the operating system has more opportunities with this background task to schedule the work of the foreground task than it did with the previous scenarios. Despite the less demanding nature of these two tasks, HT Technology still delivers a response-time benefit.

An obvious corollary to our results and analysis is that HT Technology is unlikely to deliver response-time improvements in multitasking situations in which neither the foreground nor the background tasks are computationally intensive. Though some multitasking scenarios obviously do not place heavy demands on the processor, others do involve computationally intensive application tasks. In situations with computationally intensive tasks, our investigation and results suggest that HT Technology will deliver significant response-time improvements.

Test methodology

We measured the effects of HT Technology on our sample multitasking application scenarios (see “Application scenarios”) both with hand timings of the application functions and with automated test scripts that performed those functions. Appendix B details the steps we followed in our hand timings. We developed the test scripts using IBM’s Visual Test 6.5. We focus in most of our discussions here on the results of the test scripts, because those results are generally more repeatable than hand timings. We collected results for five executions of each script with HT Technology enabled and five with HT Technology disabled on each of our test systems.

We set up the test systems as follows:

- We completed the manufacturer’s Windows XP setup and initialization process.
- We installed a purchased copy of Office 2003.
- Using the standard Windows Update and Office Update Web sites, we applied all current Windows XP and Office 2003 updates except those, such as a new version of Windows Moviemaker and Windows Messenger 4.7, that were totally unrelated to the goal of this paper. (The tests in this paper reflect all updates as of May 15, 2004.)
- To reflect the emphasis of our focus on enterprise IT users, we then acted as an enterprise IT department might and removed from all the systems all the applications that were unique to a particular system.
- Both to reflect this enterprise focus and to speed system boot for testing, we removed all non-essential startup applications using the standard Windows MSCONFIG tool.
- To ensure as consistent a starting point as possible for our performance measurements, we defragmented the disk of each system.
- Using Symantec’s Ghost utility, we then made an image of each system’s hard disk so we could return to that clean image whenever necessary in our testing.

To obtain each test result we cite here, we followed the same basic process:

1. Reboot the system.
2. Run the test script (or hand-timed application functions, as appropriate).
3. Record the result.
4. Repeat this process five times. (If any test or script failed, we discarded that test and ran the test again).
5. Take the median result of the five iterations as the representative of the group.

We consequently discuss here only the median results of five runs of any system state (i.e., with or without HT Technology enabled).

We also computed the median performance improvement HT Technology delivered. We computed this by comparing the median time for a script (or hand-timed application function) with HT Technology enabled to the median result with HT Technology disabled. To reflect the results of all three systems, we sometimes note the average HT Technology benefit; we computed this by taking the arithmetic average of the median HT Technology benefits on each of the three systems.

Appendix A: System configurations

This appendix lists the key configuration information for each of the three test systems we used. We specified the test configurations. Intel then purchased all three systems and had them shipped directly from the vendors to us.

Our goal was to have three systems that were as similar as their manufacturers' offerings reasonably permitted. We chose the following target specifications to represent common business PC configurations:

- Intel Pentium 4 processor with HT Technology, 2.80GHz
- 512MB of RAM
- 40GB hard disk
- on-board graphics running at a resolution of 1024 X 768 with 32-bit color

We installed purchased copies of the Microsoft Office 2003 applications we used in our tests. Because IT organizations frequently choose standard configurations and to make the systems as fairly comparable as reasonably possible, we manually uninstalled from each system any vendor-supplied applications we did not need for our testing. We also used Microsoft's MSCONFIG utility to remove all unnecessary startup tasks and to make sure all three systems had the same startup tasks. We then defragmented the disk of each system and saved the resulting disk image. We performed all tests on these "clean" disk images.

We used the BIOS of each system to turn HT Technology on and off. All three systems offered that ability.

Vendors typically configure similar systems slightly differently, so our test systems have similar but not identical configurations. Because of those differences and because our goal is to focus on HT Technology, not to compare individual system performance, we refer throughout this paper to the three systems as System A, System B, and System C.

System A	
Vendor	Dell Computer Corp.
Model	Optiplex GX270
Processor	Intel Pentium 4 processor with HT Technology, 2.80GHz
L2 cache	512KB
Front bus speed	800-MHz
Motherboard	Dell 0U1324
Chipset	Intel 82865G
BIOS	A03
RAM	512MB of PC2700 (333-MHz) RAM in slots 1 and 3
Hard disk	Maxtor 6E040L0 41110MB (7,200 rpm)
Hard disk buffer	2MB
Disk controller	Intel 82801 EB Ultra ATA Storage Controller
Network adapter	Intel PRO/1000 MT
Graphics chipset	Intel 82865G
Graphics RAM	96MB
Graphics driver	6.14.10.3762
Test graphics resolution	1024 X 768
Test color depth	32-bit
Sound chipset	SoundMax Integrated Digital Audio
CD-ROM drive	Samsung SC-148C
Diskette drive	None
Operating system	Microsoft Windows XP Professional Build 2600 Service Pack 1

System B	
Vendor	HP
Model	d330 ST
Processor	Intel Pentium 4 processor with HT Technology, 2.80GHz
L2 cache	512KB
Front bus speed	800-MHz
Motherboard	HP 085Ch
Chipset	Intel 82865G
BIOS	786B2 v.2.18
RAM	512MB of PC3200 (400-MHz) RAM in slots 1 and 3
Hard disk	Seagate ST340015A 40000MB (5,400 rpm)
Hard disk buffer	2MB
Disk controller	Intel 82801EB Ultra ATA Storage Controller
Network adapter	Broadcom NetXtreme Gigabit Ethernet for HP
Graphics chipset	Intel 82865G
Graphics RAM	96MB
Graphics driver	6.14.10.3762
Test graphics resolution	1024 X 768
Test color depth	32-bit
Sound chipset	SoundMax Integrated Digital Audio
CD-ROM drive	HLDS GCR-8482B
Diskette drive	Samsung SFD-321B
Operating system	Microsoft Windows XP Professional Build 2600 Service Pack 1

System C	
Vendor	IBM
Model	ThinkCentre 8189E8U
Processor	Intel Pentium 4 processor with HT Technology, 2.80GHz
L2 cache	512KB
Front bus speed	800-MHz
Motherboard	IBM 865G
Chipset	Intel 82865G
BIOS	IBM 2AKT38AUS
RAM	512MB of PC2700 (333-MHz) RAM in slot 1
Hard disk	Maxtor 6E040L0 41110MB (7,200 rpm)
Hard disk buffer	2MB
Disk controller	Intel 82801EB Ultra ATA Storage Controller
Network adapter	Intel PRO/1000 MT
Graphics chipset	Intel 82865G
Graphics RAM	96MB
Graphics driver	6.14.10.3762
Test graphics resolution	1024 X 768
Test color depth	32-bit
Sound chipset	SoundMax Integrated Digital Audio
CD-ROM drive	LiteOn LTN486S
Diskette drive	Samsung SFD-321B
Operating system	Microsoft Windows XP Professional Build 2600 Service Pack 1

Appendix B: Instructions for running the application scenarios

This appendix explains how we manually tested each of the application scenarios and how we manually measured the response times of their foreground applications. Though the vast majority of our discussions here focus on the results of the automated tests, we felt it was also important to verify that manually performing the same functions showed the same type of results as those the scripts produced.

As the instructions below reflect, to get the most consistent possible timings and to make our hand-timed actions more like the ones the automated scripts perform, we sometimes chose to follow procedures for launching applications that were different from those typical users would follow. (See Appendix C for additional information on scripting issues.) For example, we typically launch the foreground application from the Run prompt rather than by double-clicking on a desktop icon. When we made such choices we also independently verified that the typical user procedures would still show similar results with HT Technology enabled and disabled.

We consequently are confident that the benefits HT Technology demonstrated in these (and other) scenarios are ones users would realize in real work situations and are not artifacts of the measurement or scripting Technology.

We ran all tests five times with HT Technology enabled and five times with HT Technology disabled.

<u>Scenario 1</u>

This scenario uses two files:

- content.ppt – a 34MB PowerPoint presentation the background task uses. This 36-slide presentation examines the market for digital products of all types. Like many modern presentations, it contains many graphical images.
- Digital Home.pdf – a 749KB PDF document the foreground task uses. This 12-page document discusses some aspects of the equipment and infrastructure necessary in a digital home. It contains a mixture of graphics and text.

The scenario requires two applications:

- Adobe Reader 6.0 (foreground)
- Microsoft PowerPoint 2003 (background)

We used the following process in our manual tests of this scenario.

First, we set up the scenario by doing the following steps, which are only necessary once:

1. Create a desktop shortcut on the left side of the screen to the file content.ppt.
2. Create a desktop shortcut on the left side of the screen to the file Digital Home.pdf.
3. Right click on the Digital Home.pdf desktop shortcut and select Properties.
4. Copy the “Target” string. (It should look something like “C:\Whitepaper1\Sc1\Content\Digital Home.pdf”.)
5. Click the Start button and open the Run command prompt.
6. Paste the target string into that prompt, and remove the apostrophes at the string’s beginning and end.
7. Click “OK” to ensure that the command works. Adobe Reader should open the Digital Home.pdf file.
8. Exit Adobe Reader.

To run each timed test, we then did the following:

1. Reboot the system.

2. As soon as Windows XP will allow you to do so, click the Start button and open the run command prompt. It should still have the target string from setup step 6 above.
3. As soon as the Windows hourglass disappears and Windows XP has completed its startup sequence, double-click on the content.ppt shortcut on the desktop.
4. When the content.ppt file is open, click on the "Compress Picture" icon on the PowerPoint picture toolbar.
5. Select the "Web/Screen" option, and click "OK".
6. Immediately move the mouse cursor over the Run command prompt and click "OK".

Start timing the foreground task (opening the .PDF file) as soon as you click the "OK" button on the Run command prompt. Stop timing when the Digital Home.pdf file is completely open.

<u>Scenario 2</u>

This scenario uses one file, SxStiming.mdb, a 24.5MB Access database the background task uses. This file is a sales database that launches a set of queries when you click an OK button after opening the file. This scenario also uses Outlook's standard .pst data file. For ease of testing and to get the most repeatable results reasonably possible, we tested Outlook with an empty .pst file. Our experience suggests that this is a reasonable approach and that HT Technology would probably yield a larger response-time improvement in this scenario with a .pst file that contained more email and other items.

The scenario requires two applications:

- Microsoft Access 2003 (background)
- Microsoft Outlook 2003 (foreground). Outlook should have as empty a .pst file as reasonably possible and nothing in its Inbox.

We used the following process in our manual tests of this scenario.

First, we set up the scenario by doing the following steps, which are only necessary once:

1. Create a desktop shortcut on the left side of the screen to the file sxstiming.mdb.
2. Start Access and set its security level to permit the execution of macros, which SxStiming.mdb uses.
3. Click the Start button and open the Run command prompt.
4. Type "outlook:inbox" into that prompt and click "OK".
5. Verify that Outlook opens with no messages in the Inbox.
6. Exit Outlook.

To run each timed test, we then did the following:

1. Reboot the system.
2. As soon as Windows XP will allow you to do so, click the Start button and open the run command prompt. It should still have the "outlook:inbox" string from setup step 3 above.
3. As soon as the Windows hourglass disappears and Windows XP has completed its startup sequence, double-click on the sxstiming.mdb shortcut on the desktop and click "OK" in response to the prompt that will appear.
4. Immediately move the mouse cursor over the Run command prompt and click "OK".

Start timing the foreground task (launching Outlook) as soon as you click the "OK" button on the Run command prompt. Stop timing when Microsoft Outlook completely opens and there is a "0" in the lower left hand corner of the Outlook window. (The "0" is an easy visual indicator that Outlook is completely open.)

<u>Scenario 3</u>

This scenario uses two files:

- Excel2Min.xls – a 446KB spreadsheet the foreground task uses. This file contains three worksheets that provide a very simple model of a call-management system.
- test.doc – a 19.7MB document the background task uses. This 43-page file focuses on media center PCs and, like many modern business documents, contains multiple graphical images on its first page, a cover sheet

The scenario requires two applications:

- Microsoft Excel 2003 (foreground)
- Microsoft Word 2003 (background)

We used the following process in our manual tests of this scenario.

First, we set up the scenario by doing the following steps, which are only necessary once:

1. Create a desktop shortcut on the left side of the screen to the file test.doc.
2. Create a desktop shortcut on the left side of the screen to the file Excel2Min.xls.
3. Double-click the test.doc desktop icon to open that Word document.
4. Size it so it does not take the whole screen or obscure the icons on the left of the screen.
5. Exit Word.
6. Right click on the Excel2Min.xls desktop shortcut and select Properties.
7. Copy the "Target" string. (It should look something like "C:\Whitepaper1\Sc3\Content\Excel2Min.xls".)
8. Click the Start button and open the Run command prompt.
9. Paste the target string into that prompt.
10. Click OK.
11. Verify that Excel opens Excel2Min.xls.
12. Exit Excel.

To run each timed test, we then did the following:

1. Reboot the system.
2. As soon as Windows XP will allow you to do so, click the Start button and open the run command prompt. It should still have the "C:\Whitepaper1\Sc3\Content\Excel2Min.xls" string from setup step 7 above.
3. As soon as the Windows hourglass disappears and Windows XP has completed its startup sequence, double-click on the test.doc shortcut on the desktop to open that file.
4. Wait for Word to completely open the file. You will know it has done so when the full page count of 43 appears at the bottom of the Word window.
5. Wait four seconds, as a user often will while deciding what to do next.
6. Change the document to the Web layout view.
7. Wait two seconds, as a user typically will when surprised that an operation is taking longer than he expected.
8. Immediately move the mouse cursor over the Run command prompt and click "OK".

Start timing the foreground task (the opening of the Excel file) as soon as you click the "OK" button on the Run command prompt. Stop timing when Excel has completely drawn the first page of the first worksheet.

Appendix C: Issues in script development

To the best of our knowledge, IBM's Visual Test 6.5 is the tool in the widest use today for constructing application-based benchmarks and performance tests for PCs running variations of Microsoft Windows. We have used this product (and previous versions of it) for many years in the construction of performance tests. The tool also, however, has some stated limitations that unavoidably affect the way you develop performance tests with it.

First, the tool's own documentation notes that its primary goal is to be an application testing automation tool, not a benchmark development tool. Consequently, the granularity of some of its functions and the way some functions behave are not ideal for benchmark development.

IBM also does not officially support Visual Test 6.5 for the Windows XP operating system. Because that is the leading and most current desktop version of Windows today, we nonetheless felt it was essential to use that operating system for our tests here.

The presence of any scripting tool also has the potential to affect the performance of a system. The tool unavoidably must occupy some memory and consume some processing power, so developing a performance-measurement script with such a tool involves maintaining a delicate balance between using the tool to automate typical real user behavior and minimizing the effects of the tool on system performance.

The combination of these limitations means that in some cases functions that are simple for a human to perform are either not possible or not appropriate for a Visual Test script whose goal is to accurately measure performance. For example, though a user can easily launch any of the background applications we used in the test scenarios here and then double-click on a desktop icon of a data file (which a foreground task would then launch), a Visual Test script might not be able to do the same, especially without HT Technology, until the background task terminated. The Visual Test script would instead end up waiting until the operating system was able to allocate processor resources to it—a wait that would essentially serialize functions real users would be performing in parallel.

Consequently, we sometimes had to use scripting techniques that would achieve the same results as typical user behavior and not encounter any of the above limitations or problems. Such techniques include inserting delays to mimic user think time and launching applications with a click on the OK button of a pre-filled Run command line. The hand timing instructions we provide in Appendix B reflect those techniques, so following those instructions will yield results similar to those the scripts produce. We should note here, as we did in Appendix B, that whenever we had to use one of these alternative techniques we manually verified that doing so did not significantly change the way the system behaved and that real users performing the same actions in more typical ways would see the benefits of HT Technology that we describe.

The timings the scripts produce also inevitably contain some variability. This variability is a result of the combination of the tool's limitations and the generally asynchronous nature of the many functions Windows XP and other modern operation systems have running.

One of our goals was to produce scripts that Intel could own and distribute. To make such distribution as legally simple as possible, Intel needed to own all rights to the data files in the scripts. At the same time, we needed to be able to do all our research and analysis using data files we provided. To address both these requirements, we did all initial research and scripting for this White Paper using data files of our choice. When we had settled on the three scenarios we wanted to use here, we gave Intel the specifications—such as size, type of content, and graphical and computational intensity—for all the files we used. Intel then provided files that met those specifications. We verified that all the files Intel provided did indeed meet our specifications and satisfied ourselves that the files were appropriate for our use. We then used those files in our final test scripts and hand timings, in the process further verifying that the HT Technology benefits we found with these files were basically the same as those we found with our original files.

Finally, though one of the goals of this effort was to produce distributable scripts, we were not trying to build bulletproof benchmarks intended for wide distribution. We developed the scripts to mimic user behavior on our specific test systems; on significantly faster or slower systems the scripts might show different levels of HT Technology benefits or even possibly fail to work. So, though the scripts are as reliable, self-contained, and free of system dependencies as we could reasonably achieve within the project's timeframe, they do sometimes fail or encounter problems. Should a problem occur, rebooting the system and running the script again will generally yield a good result.

Principled Technologies

Principled Technologies, Inc.
10024 Sycamore Road
Raleigh, NC 27613
info@principledtechnologies.com