Principled Technologies

White Paper
September 2004

Effects of Hyper-Threading Technology on Response Time in Basic System Security Scenarios

For Intel Corporation

Executive summary

Intel® Corporation (Intel) commissioned Principled Technologies^{SM} (PT) to investigate the effects of Hyper-Threading (HT) Technology on response time—the speed with which applications respond to user commands—while doing basic computer system security functions such as virus scanning. Because HT Technology can run two instruction streams simultaneously, it can improve the response time of systems when executing more than one application at a time, including security applications. We looked at how well HT Technology fulfilled that possibility with basic security situations by running scenarios users might commonly execute, such as performing virus scans and encrypting email attachments for secure transmission. Business applications like the very popular Microsoft Office 2003 are among the most common programs in use in enterprises today. Consequently, we used applications from that suite in our testing. We also used Symantec's Norton AntiVirus program, a leading security product, for virus-scanning, and Adobe's popular PDF file-sharing standard in some of the tests.

We found that HT Technology frequently improved the response time of PCs in the security scenarios we examined. Before beginning our testing, we discussed typical security issues and application usage with IT managers and looked at other sources of information on those topics. We then investigated different scenarios in which users might address some basic aspect of system security. For each scenario, the user was working on one thing (the foreground task) while a bigger, more time-consuming function in another application (the background task) was executing. We studied both scenarios involving background security functions and those with security functions as part of their foreground activity. In both cases, we generally found that HT Technology significantly improved the response time of the foreground application. This response-time improvement usually occurred with little or no increase in the time for the background task to complete. In some cases the background task performance even improved. Finally, we found that HT Technology consequently generally improved overall system throughput, the amount of work the system can accomplish in a given amount of time.

Key findings

- Hyper-Threading Technology improved user response time in all three of the sample basic security multitasking scenarios we present here.
- The response-time improvement from HT Technology occurred with no penalty to the completion time of the background tasks.
- In some cases, HT Technology increased overall system throughput.
- The response-time improvement from HT Technology in these multitasking situations was generally much greater than the throughput improvement.

To ensure that our tests did not rely on characteristics specific to one vendor or system, we tested on PCs from two different leading computer vendors. We wanted to conduct our testing on the latest system technologies enterprise IT managers were evaluating, so we specified the following configuration:

- Intel® Pentium® 4 Processor with HT Technology 3.40 GHz
- Intel® 915G Express chipset with Intel® GMA 900 graphics
- 512MB of RAM
- 80GB Serial ATA (SATA) hard disk
- Windows XP Professional

We purchased both test systems from the vendor Web sites using their enterprise class system portals. As vendors usually configure similar systems slightly differently, our test systems have similar configurations but are not identical. In light of those differences and our goal to focus on HT Technology, not to compare different vendor's system performance, we refer to these PCs as System A and System B. Appendix A provides configuration details for each of these systems.

We found that HT Technology improved response time on both systems on multiple tests of application scenarios employing multiple applications.

We tested basic security scenarios that our research and experience indicated were akin to those real enterprise users might encounter. After we selected scenarios we felt were representative and appropriate for this paper, we manually tested and timed them by launching combinations of foreground and background application tasks. We chose three typical scenarios to examine closely in this paper.

To test our scenarios repeatedly and accurately, we took the hand-timed scenarios and automated them using scripts. Of course, such scripts do add a small amount of overhead to the application functions and are not exact duplications of the hand-timed scenarios. They provide, however, reasonable and repeatable mechanisms for accurately running the scenarios multiple times on multiple systems. We created the scripts in IBM's Rational Visual Test 6.5 (see below for more on the scripts). We used the times from our tests using these scripts in our analysis below.

Figure 1 shows that HT Technology delivered significant improvements for all three of the system security scenarios on both test systems. The average response-time improvement was about 80 percent on both systems for all three scenarios. (We look more closely at those scenarios in the next section.)

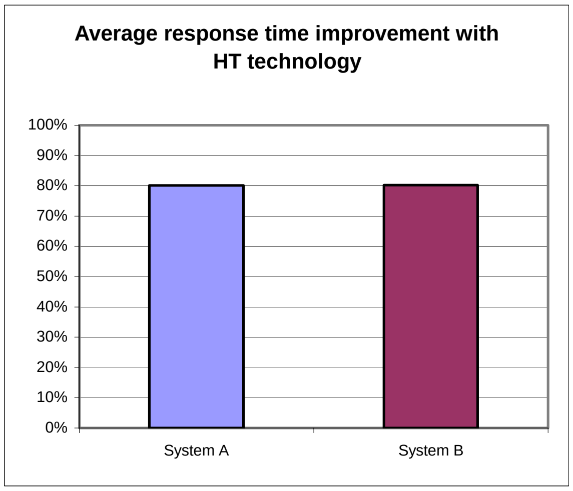


Figure 1: Average response time improvement over all three test scenarios with HT Technology. Improvement is the percentage faster the test applications responded to user commands with HT Technology enabled than without it.

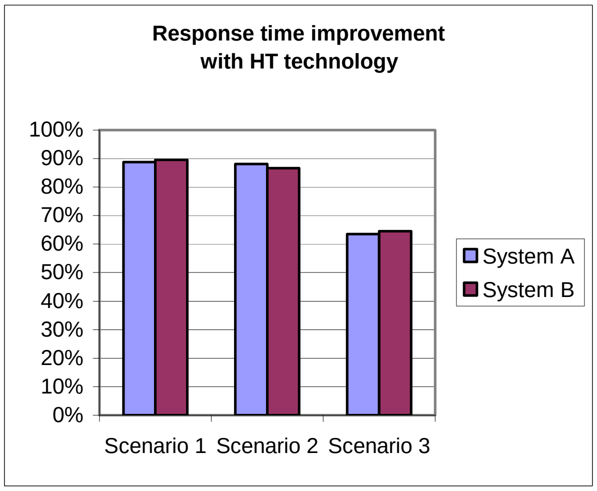


Figure 2: Response time improvement for each of the three test scenarios with HT Technology. Improvement is the percentage faster the test applications responded to user commands with HT Technology enabled than without it.

Figure 2 shows the response time improvement percentage for the two systems on the three different scenarios. Although the two systems exhibited different amounts of improvement in each of our test scenarios, HT Technology produced large response-time benefits in all three scenarios on both systems.

In the following sections we look more closely at these application scenarios (Application scenarios) and discuss some useful basics about HT Technology (HT Technology background). We then examine the results of our tests with those scenarios (Test results and analysis) and how we actually 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 development of the scripts.

Application scenarios

We began the application scenario selection process by speaking with IT managers about basic PC security issues and procedures. We also did some additional informal research on common security functions with typical office applications and other software in widespread use in enterprises. We looked at different multitasking scenarios using Symantec's Norton Antivirus product, Microsoft Office 2003 applications, and such other common business programs as Adobe's Adobe Reader 6.0. We usually first explored scenarios with HT Technology disabled, running them by hand as typical users would. As we ran the scenarios, 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 tried those same multitasking scenarios with HT Technology enabled.

After looking at a variety of scenarios with multiple applications, we decided on three we felt represented a range of common user activities involving basic system security functions.

In Scenario 1 a virus scan is running, and it is currently working on a Windows directory that contains compressed files. Such scans may occur during working hours because the user is concerned about her machine or because the IT department has remotely started the scan. Meanwhile, the user needs to continue working. She is working on a short (seven-slide) PowerPoint 2003 presentation about the market for digital products of all types, so she opens that file.

Scenario 2 involves three different applications—Access 2003, Outlook 2003, and Word 2003—plus a security certificate. 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 send an email message with an encrypted attachment, an image file. He has turned on encryption in Outlook, so by saving the message with the attached image file he starts the encryption.

Our final example, Scenario 3, involves on-demand virus scanning, something users regularly experience. A user is working on a Word document about media center PCs and decides to generate a PDF version of that document using Adobe's Distiller 6.0. After starting the PDF creation, the user waits a few seconds and decides to work on an Excel file. Opening that Excel spreadsheet, which contains a very simple model of a call-management system, causes Norton Antivirus to scan the file—a common example of on-demand virus-scanning.

For more details on how we executed each of the scenarios, our specific test functions, and the files the scenarios involved, see Appendix B.

HT Technology background

Before discussing and analyzing the results of our tests, some HT Technology background is useful. User computing situations that benefit the most from HT Technology are those in which a background task appears, on a system without HT Technology or one with HT Technology disabled, to utilize all or nearly all of the processing capacity of the system. 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, contain multiple execution units, the parts of the processor that actually execute program instructions. Many of the 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 using what it sees as the second processor. That extra work often ends up taking advantage of 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, such as a virus scan, that if running alone on the system would take 50 seconds to complete. The task appears to use 100 percent of the system's processing power for most of those 50 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 a significant amount 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 very little. Thus, the user effectively cannot begin to work in the second (foreground) application until the first (background) application has completed its work and the second has finally received its chance to launch. The foreground task's response time—the time it takes before responding to the user's command to start—is thus over 50 seconds. We saw delays of this type in our investigation of basic security scenarios.

Consider the same example on a system with HT Technology. The user launches the same background application. This time, however, that application appears to the operating system to be using all of only one of the two (logical) processors of the system; the other processor appears to the operating system to be 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 50-second response time without HT Technology. Our first scenario is an example of almost exactly this sort of improvement.

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 would take to complete. So, the background task would still complete in roughly the same 50 seconds it would have required if running alone.

In addition, the system's total throughput for those 50 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. All three of our sample scenarios, for example, show quite large response-time improvements.

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 results of tests with the scripts for each of the scenarios and discuss the effects of HT Technology on the response time and 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. We're measuring response times, so smaller times are better.)

Test PC	Foreground (PowerPoint open)		Background (Virus scan)	
	HT On	HT Off	HT On	HT Off
System A	6.344	56.312	49.437	56.375
System B	5.485	52.390	48.750	52.406

Figure 3: Median foreground and background test times for the Scenario 1 script on both 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 both test systems. HT technology also delivers a much smaller but still significant improvement in the performance of the background virus scan. In this case, HT Technology thus improved foreground response time, background throughput, and overall throughput.

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 for two reasons: we captured the processor utilization of each task running alone with HT Technology disabled, and we did not stop gathering processor utilization data at exactly the same points as the scripts considered the tasks complete.)

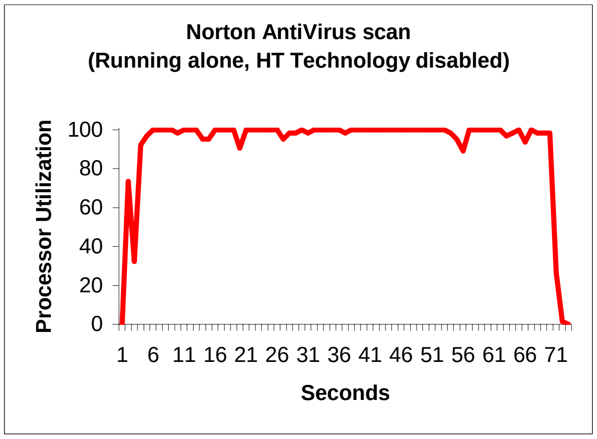


Figure 4: System processor utilization, without HT Technology, of the Scenario 1 background task running alone.

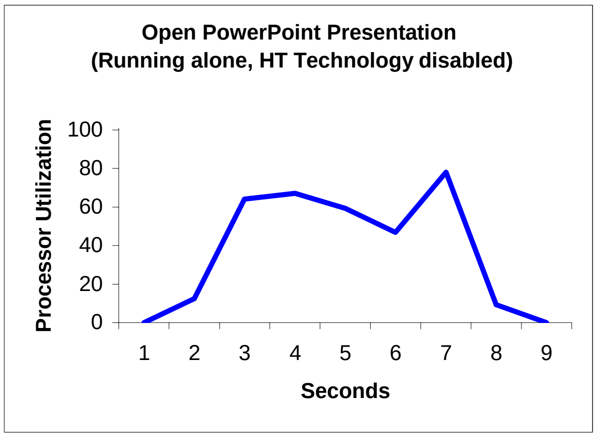


Figure 5: System processor utilization, without HT Technology, of the Scenario 1 foreground task running alone.

As Figure 4 demonstrates, the background task—the virus scan of a file that contains compressed files—is so computationally intense that it uses 100 percent of the available processing power for the vast majority 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 PowerPoint presentation file—is nowhere near as demanding of the CPU, it nonetheless spends much of its time consuming over half of the system's processor power and so needs significant processing 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.

As the results in Figure 3 show, HT Technology delivers this response time improvement in Scenario 1 while also improving the performance of the background application. 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 the system's overall throughput is higher.

Scenario 2, as Figure 6 shows, is another case in which HT Technology delivers both vastly better response time and an improvement, though in this case a very small and not significant one, in background task performance.

Test PC	Foreground (Outlook Startup)		Background (Access Query)	
	HT On	HT Off	HT On	HT Off
System A	2.406	20.187	96.015	97.937
System B	2.422	18.203	96.125	96.375

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

The response-time win for the foreground application is very large and something a user would greatly appreciate. The times for the background task are also slightly though not significantly better with HT Technology enabled.

The key to understanding why HT Technology was able to so greatly improve response time again stems from 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.

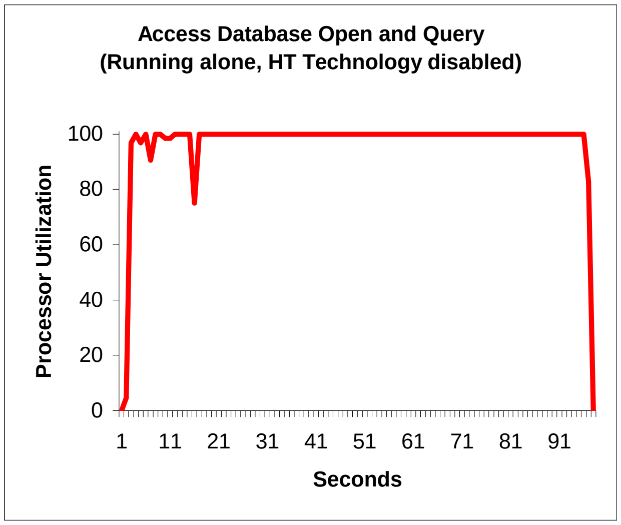


Figure 7: System processor utilization, without HT Technology, of the Scenario 2 background task running alone.

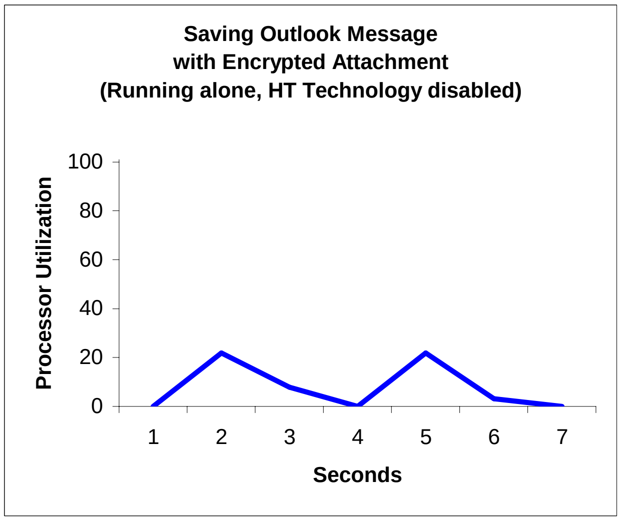


Figure 8: System processor utilization, without HT Technology, of the Scenario 2 foreground task running alone.

Figure 7 shows that on a system without HT Technology the Access query background task uses 100 percent of the processor for almost the entire time it is running, though it does dip below 100 percent processor utilization on multiple occasions. Figure 8 shows, by contrast, a relatively undemanding foreground task that rarely uses over 20 percent of the processor utilization—and then only for very brief times.

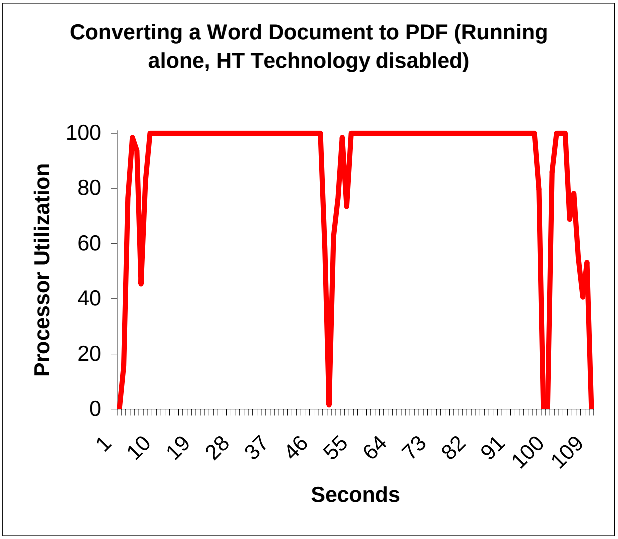
Without HT Technology, the background task is still so demanding that the foreground task ends up getting very little processor time. With HT Technology, however, the combination of the multiple dips in the processor requirements of the background and the relatively short duration of any significant processor usage of the foreground task 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 full speed while also handling the foreground work. The result was, as our tests show, dramatically better response time, a very slightly better background time, and thus better overall throughput.

As we noted in the Application scenarios section, we also wanted to examine scenarios that included on-demand security functions, such as virus scans that occur when a user opens a file. Scenario 3 provides an example of this situation. Figure 9 shows the median results of our Visual Test scripts of that scenario. Here, the response-time benefit of HT Technology is very large, though not quite as big as in the first two scenarios. The effect on the performance of the background task is negligible, with differences of under a second with and without HT Technology, and with the background being marginally faster with HT Technology on one system (System B) and marginally slower on the other (System A).

Test PC	**Foreground (Excel Spreadsheet Open and Virus Scan)**		**Background (Converting a Word Document to PDF)**	
	HT On	**HT Off**	**HT On**	**HT Off**
System A	16.016	43.844	110.140	109.578
System B	15.703	44.281	110.469	110.797

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

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



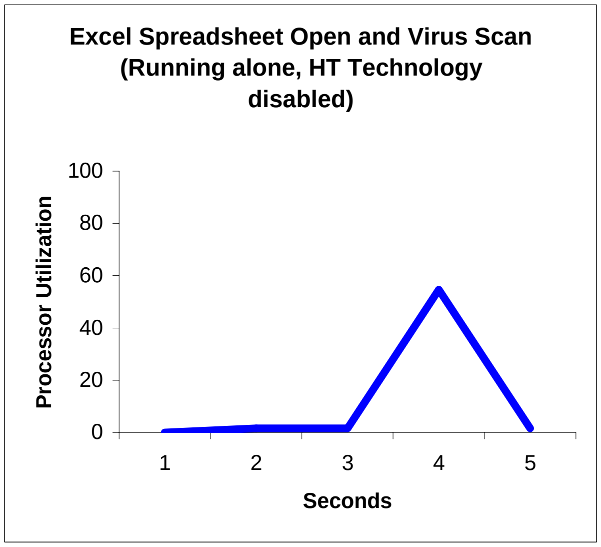


Figure 10: System processor utilization, without HT Technology, of the Scenario 3 background task running alone.

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

HT Technology delivers a very large response-time improvement in this scenario, over sixty percent and an average of over 27 seconds of time to the user. It does not show quite as large a win in this scenario as in the previous scenarios because the processor utilization of the background task is not at 100 percent for much of its initial time and because the processor utilization of the foreground task is significant only for a few seconds. Given the relatively undemanding nature of the foreground task, however, the HT Technology improvement is quite striking.

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 or background tasks place heavy demands on the processor. Though some multitasking scenarios obviously are not computationally intensive, others do involve such application tasks. In basic system security 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 multi-application scenarios (see “Application scenarios”) both by hand and with automated test scripts. Appendix B details the steps we followed when we hand-timed the scenarios. We developed the test scripts using IBM’s Visual Test 6.5. In this paper, we concentrated most of our discussions on the results of the automated test scripts, because those results are generally more repeatable than hand timings. We collected results for five runs of each script with HT Technology enabled and five with HT Technology disabled on each of the 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 Windows XP SP2 and those updates, 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 such updates, again except Windows XP SP2, as of September 13, 2004. We chose not to install Windows XP SP2 because it was publicly available for a relatively short time before our testing and enterprise IT managers were not yet widely deploying it.)
- To reflect the emphasis of our focus on enterprise IT users, we then acted as an enterprise IT department might and removed from both test systems all the applications that were unique to a particular system.
- Both to reflect this enterprise focus and to speed system boot times 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 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’s results 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 both 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 systems.

Appendix A: System configurations

In this appendix, we list the important configuration information for both of the test systems in this test. We purchased these systems in these test configurations from the enterprise sections of the Web sites of the two respective vendors.

Our goal was to have a pair of test systems that were as similar as their manufacturers' standard configurations allowed. We chose the following specifications to represent the latest technology that enterprise IT managers are evaluating:

- Intel® Pentium® 4 Processor with HT Technology 3.40 GHz
- Intel® 915G Express chipset with Intel® GMA 900 graphics
- 512MB of RAM
- 80GB Serial ATA (SATA) hard disk
- Windows XP Professional with SP1

Each vendor offered the options of having Norton AntiVirus and Microsoft Office 2003 Professional Edition pre-installed on the enterprise systems, and we chose those options. Because IT organizations often employ standard configurations and to make the systems as comparable as reasonably possible, we manually uninstalled from each system any vendor-supplied applications we did not need for our testing. We used Microsoft's MSCONFIG utility to remove all unnecessary startup tasks and to make sure both systems had the same startup tasks. We applied all Windows updates available as of September 13, 2004 except those unrelated to enterprise systems, such as Movie Maker 2, and Windows XP SP2. At the time of our testing, we felt that this service pack was not yet in widespread use in enterprises. We then defragmented the disk of each system and saved the resulting disk image. We performed all tests on these "cleaned" disk images.

We used the BIOS of each system to turn HT Technology on and off as needed for the testing.

Every vendor typically configures similar systems slightly differently. Consequently, 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 systems as System A and System B.

System A	
Vendor	Dell Computer Corp.
Model	Optiplex GX280
Processor	Intel® Pentium® 4 Processor with HT Technology 3.40 GHz
L2 cache	1MB
Front bus speed	800-MHz
Motherboard	Dell 0Y5638
Chipset	Intel® 82915G
BIOS	A02
RAM	512MB of PC3200 (400-MHz) RAM in slots 1 and 2
Hard disk	Maxtor 6Y080M0 (7,200 rpm) 80,000MB
Hard disk buffer	8MB
Disk controller	Intel® 82801 FB Ultra ATA Storage Controller
Network adapter	Broadcom NetXtreme 57xx Gigabit Ethernet Controller
Graphics chipset	Intel® 82915G
Graphics RAM	128MB
Graphics driver	6.14.10.3889
Test graphics resolution	1024 X 768
Test color depth	32-bit
Sound chipset	SoundMax Integrated Digital Audio
CD-ROM drive	Samsung SC-148A
Diskette drive	NEC FD1231M
Operating system	Microsoft Windows XP Professional Build 2600 Service Pack 1

System B	
Vendor	HP
Model	Dc7100 CMT (DX438AV)
Processor	Intel® Pentium® 4 Processor with HT Technology 3.40 GHz
L2 cache	1MB
Front bus speed	800-MHz
Motherboard	HP 0968h
Chipset	Intel® 82915G
BIOS	786C1 v.01.05
RAM	512MB of PC3200 (400-MHz) RAM in slots 1 and 2
Hard disk	Seagate ST380013AS (7,200 rpm) 80,000MB
Hard disk buffer	8MB
Disk controller	Intel® 82801 FB Ultra ATA Storage Controller
Network adapter	Broadcom NetXtreme Gigabit Ethernet Controller
Graphics chipset	Intel® 82915G
Graphics RAM	128MB
Graphics driver	6.14.10.3889
Test graphics resolution	1024 X 768
Test color depth	32-bit
Sound chipset	SoundMax Integrated Digital Audio
CD-ROM drive	MSI MS-8148C1
Diskette drive	Mitsumi D353M3D-5055
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 in this White Paper 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 launched 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 enabled and five times with HT disabled.

<u>Scenario 1</u>

This scenario uses two files:

- Content7.ppt – a 4.75 MB PowerPoint presentation the foreground task uses. This 7-slide presentation gives an overview of the market for digital products of all types. Like many modern presentations, it contains many graphical images.
- Driver.cab– a 73.1 MB Windows XP-provided cab file that the scenario scans during the background task. We do not include this file with the script; it is in C:\windows\driver cache\i386 by default on Windows XP systems.

The scenario requires two applications:

- Norton AntiVirus 2004 (background)
- Microsoft PowerPoint 2003 (foreground)

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

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

1. Create a desktop shortcut on the left side of the screen to the file content7.ppt.
2. Right-click on the content7.ppt desktop shortcut and select Properties.
3. Copy the “Target” string. (It should look something like “C:\Whitepaper2\SC1\Content\content7.ppt”.)
4. Click the Start button and open the Run command prompt.
5. Paste the target string into that prompt.
6. Click “OK” to ensure that the command works. PowerPoint should open the content7.ppt file.
7. Exit PowerPoint.
8. Copy the file C:\windows\driver cache\i386\driver.cab to C:\.
9. Click the Start button and open the Run command prompt.
10. Paste the following string into that prompt: navw32.exe /NORESULTS /HEUR:3 "C:\driver.cab"
11. Click “OK” to ensure that the command works. Norton AntiVirus 2004 should open, scan the driver.cab file, and then close.

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 11 above.
3. As soon as the Windows hourglass disappears and Windows XP has completed its startup sequence, move the mouse cursor over the Run command prompt and click "OK". This launches the background task.
4. Scroll through the pull-down menu for the Run command prompt to locate the Run command for PowerPoint that you created in step 5 above and click "OK".

Start timing the foreground task (opening the .PPT file) as soon as you click the "OK" button on the Run command prompt in step 4 above. Stop timing when all seven thumbnails for that presentation display in the left panel of the PowerPoint window. When you're done, exit from PowerPoint.

Scenario 2

This scenario uses two files

- digitalhome.jpg – an 81KB image file for use in a presentation. The foreground task uses this 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 each time with an empty .pst file. Our experience suggests that this is a reasonable approach because HT technology would probably yield a larger response-time improvement in this scenario with a .pst file that contained more email messages and other items.

The scenario requires three applications:

- Microsoft Access 2003 (background)
- Microsoft Outlook 2003 (foreground). As we noted above, Outlook should have as empty a .pst file as reasonably possible, with nothing in its Inbox.
- Microsoft Word 2003 (foreground). For this scenario, Outlook edits messages using Microsoft Word.

This scenario also requires a GlobalSign PersonalSign 30-day Demo digital certificate. To get the certificate: go to http://www.globalsign.com/digital_certificate/personalsign/index.cfm and request and then install a 30-day certificate.

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

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

1. Create a copy of c:\whitepaper2\sc2\content\original-sxstiming.mdb in the same location and name it SxStiming.mdb
2. Create a desktop shortcut on the left side of the screen to the file SxStiming.mdb.
3. Start Access and set its security level to permit the execution of macros, which SxStiming.mdb uses.
4. Position Access so its window displays on the left third of the screen.
5. Create a desktop shortcut on the left side of the screen for Outlook.
6. Verify that Outlook opens with no messages in the Inbox, and position Outlook so that its Window displays on the right third of the screen.
7. In Outlook go to Tools/Macro/Security and set the security level to low. Click "OK" to close the Security dialog.
8. In Outlook, go to the Tools/Options menu entry and then to the Mail Format tab of the dialog box that appears. Set the message format to HTML, and check the 'Use Microsoft Word 2003 to edit e-mail messages' checkbox.

9. Go to the Other tab in the Outlook Options dialog. Click on Advanced Options button. Check the ‘Startup in this folder’ setting. If Inbox is not in the list as the startup folder, click on the browse button and select ‘Inbox’ from the list of folders, then press OK to close the ‘Select Folder’ dialog and again to close the ‘Advanced Options’ dialog.
10. Go to the Security tab and check “Encrypt contents and attachments for outgoing messages.“ Click “OK” to close the Options dialog.
11. Click the “New” button on the Outlook window to create a new email message. Clear any messages that display before opening the new message. Verify the e-mail message opens in Word, and position the message window in the middle third of your screen.
12. Click the “Insert File” button. The “Insert File” dialog displays.
13. Type the string “c:\whitepaper2\sc2\content\digitalhome.jpg” into the file-name box. (Omit the apostrophes.)
14. Click “Insert”.
15. Exit Access, Word, and Outlook.

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

1. Reboot the system.
2. 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 wait three seconds.
3. Double-click on the Outlook shortcut that you created in Step 3 above and wait three seconds.
4. To start the background task select “OK” in the Microsoft Access “Start Queries” dialog.
5. In Outlook, click on the “New” button to create a new message. Wait two seconds.
6. Click the “Insert File” button. The “Insert File” dialog displays.
7. From the pull-down menu choose the string that you created in setup step 12 above. Click “Insert” and wait a second.
8. Select File/Exit to exit the message. A Microsoft Office Word dialog displays asking, “Do you want to save the changes to ‘Untitled Message’?”.
9. Wait 10 seconds, a delay a user might have while re-reading the message before saving it.
10. To start the timed part of the foreground task, select “Yes” to the Microsoft Word “Do you want to save the changes to ‘Untitled Message?’” dialog.

Start timing the foreground task (saving, and in the process of saving, encrypting the Outlook message) as soon as you click the Yes button on the “Do you want to save the changes to ‘Untitled Message?’” dialog. Stop timing when the Microsoft Word window that displays the message closes.

Scenario 3

This scenario uses two files:

- Excel2Minlarge.xls – a 7.44MB spreadsheet the foreground task uses. This file contains three worksheets that provide a very simple model of a call-management system.
- Test3.doc – a 3.7MB document the background task uses. This four-page file focuses on media center PCs and, like many modern business documents, contains multiple graphical images on its first page.

The scenario requires three applications:

- Norton AntiVirus 2004 (foreground)
- Adobe Distiller 6.0 (background) (We used a 30-day trial version in our tests, but a purchased version will, of course, work as well.)
- Microsoft Word 2003 (background)

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

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

1. Create a desktop shortcut on the left side of the screen to the file excel2minlarge.xls.
2. Right click on the excel2minlarge.xls desktop shortcut and select Properties.
3. Copy the “Target” string. (It should look something like “C:\Whitepaper2\SC3\Content\excel2minlarge.xls”.)
4. Click the Start button and open the Run command prompt.
5. Paste the target string into that prompt.
6. Click “OK” to ensure that the command works. Excel should open the excel2minlarge.xls file.
7. Size the Excel window so it displays on the left half of the screen and does not obscure the icons on the left.
8. Open Norton AntiVirus. Locate the AutoProtect item in the Security Scanning Features section. If AutoProtect is not already set to on, set it on.
9. Click on the Norton AntiVirus Options button. Locate the AutoProtect item in the System box on the left side of the Norton AntiVirus Options window. Click on this item to expand it. Click on the Bloodhound item in the expanded list. If ‘Enable Bloodhound heuristics’ isn’t checked, check it. If the ‘Highest level of protection’ radio button isn’t selected, select it.
10. Locate the Miscellaneous item in the Other box on the left side of the Norton AntiVirus Options window. Click on this item. If the ‘Enable Office Plugin’ item isn’t checked, check it. With this item checked, Norton AntiVirus will scan Excel and Word documents when you open them.
11. Close Norton AntiVirus.
12. Create a desktop shortcut on the left side of the screen to the file test3.doc.
13. Double-click the test3.doc desktop icon to open that Word document.
14. Size the Word window so it displays on the right half of the screen.
15. Exit Word.

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

1. Delete the file C:\whitepaper2\SC3\content\test3.pdf if it exists.
2. Reboot the system.
3. 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 5 above.
4. As soon as the Windows hourglass disappears and Windows XP has completed its startup sequence right-click on the test3.doc file and select “Convert to Adobe PDF”.
5. When the “Save Adobe PDF File As” dialog displays, click “Save” to save it to the default file name This step launches the background task.
6. Wait 10 seconds, as a user might while deciding what to do next.
7. Click “OK” on the Run dialog to launch the Excel file.

Start timing the foreground task (opening the Excel file and, in the process, virus-scanning that file) as soon as you click “OK” on the Run dialog to launch the Excel file. Stop timing when Excel displays the excel2minlarge filename in the Excel window title.

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 various versions of Microsoft Windows. We have used this product (and previous versions of it) for many years in to build performance tests. The tool does, however, have 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 a tool for automating application testing, 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 in our tests.

The presence of any scripting tool 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 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 have to wait 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, to avoid these limitations and problems we sometimes had to use scripting techniques that would achieve the same results as typical user behavior but not exactly mirror that behavior. 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 type of HT Technology benefits 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 operating systems have running at any given time.

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 we provided. 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 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 for wide distribution and use. 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 benefit 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.
4813 Emperor Blvd., Suite 100
Durham, NC 27703
info@principledtechnologies.com