Performance Testing Guide for Windows

August 18, 2009

Abstract

This paper provides information about testing performance on Windows® 7. It provides guidelines for hardware and software industry professionals, information technology (IT) professionals, and technical users who run performance tests to ensure accurate, repeatable results.

This information applies to the following operating systems:

Windows 7
Windows Vista®

References and resources discussed here are listed at the end of this paper.

The current version of this paper is maintained on the Web at:
http://www.microsoft.com/whdc/system/sysperf/Win7Perf.mspx

Disclaimer: *This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.*

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

Document History

Date	Change
August 18, 2009	Updated for Windows 7. Includes changes for new Windows features, more detailed methodology, and reorganization.
July 2007	First publication as “Measuring Performance in Windows Vista”

Contents

Introduction

A key goal for Windows® 7 is to improve performance in common customer scenarios. To increase system responsiveness and performance, Windows 7 adapts over time to user behavior and takes advantage of the machine's idle cycles to perform background tasks. Although Microsoft focused on making "in-box" background tasks efficient, these tasks can nevertheless create challenges for those who want to test performance in a controlled environment.

When you evaluate performance, ensure that you understand whether the tests measure raw hardware performance or operating system performance. Tests that measure raw hardware performance often do not represent the types of end-user scenarios for which Windows is optimized. To measure performance in end-user scenarios, use tests that are designed to perform the same kinds of tasks that end users frequently perform.

This paper provides general guidance and points to consider when you conduct performance testing. In particular, it explains how the system adjusts its behavior and how the system services and settings affect performance measurements. By understanding these details, you can ensure consistent results and address issues during testing.

Except as noted, the topics in this paper apply to both Windows 7 and Windows Vista® and the best practices apply to both operating systems.

This paper is intended for hardware and software industry professionals, information technology (IT) professionals, and enthusiasts who want to understand how to measure performance on Windows 7 and Windows Vista.

Windows Performance Testing Challenges

Evaluating Windows performance can be difficult because the operating system attempts to improve its performance over time based on observed usage patterns. The following Windows features present challenges that affect performance evaluation:

- Background tasks.

  Some tasks run in the background when the system is idle. Consequently, system behavior changes based on whether the system is idle or a user is interacting with the PC.

- Scheduled tasks.

  Scheduled tasks can be triggered by user presence, time of day, or user logon.

- SuperFetch™ and memory management.

  The memory manager and SuperFetch monitor system activity and optimize for common scenarios over time.

- Performance versus power.

  Windows can scale performance depending on the current load while it enables lower power consumption.

Background Tasks

When the system is otherwise idle, Windows performs background tasks. These tasks are typically related to maintenance, such as running the disk defragmenter to help improve disk layout or using Windows Defender to scan for malware. The Windows Task Scheduler monitors user activity and CPU and disk utilization to determine when the system is idle.

If a performance test includes idle periods within its runtime, background tasks can run during the test and can cause the results to vary significantly from other runs of the same test. Furthermore, some background tasks, such as disk defragmenter, can dramatically improve performance results. Ensure that you know whether these tasks have previously run on the platform, because some tasks run in the next idle period if they could not complete during a previous run. Many background tasks are scheduled to run after the system is idle for a specified duration. By using Task Scheduler, users can configure a delay for each individual task, so that the task does not start until the system has been idle for the additional delay period. The total delay can be up to 15 minutes to ensure that the system is idle plus the delay that the user configured for the individual task.

We recommend that you call the **ProcessIdleTasks** function to help ensure that background tasks run before performance testing. This function notifies the Task Scheduler that the system is idle. However, it does not prevent background tasks from running during idle periods that occur during the test. For information about how to run this function, see “Test Methodology” later in this paper.

Scheduled Tasks

In addition to background tasks that run when the system is idle, Windows 7 includes scheduled tasks that run based on a timer or a predefined system trigger. For example, Windows Update runs at a particular time each day or week and System Restore runs at a predefined time each day and at system startup. The first time Windows boots after installation, Windows Defender immediately downloads definition files from Windows Update and scans for malware.

All these tasks pose significant challenges when evaluating performance in a lab environment. You must be aware of the scheduled tasks on the test systems and ensure that any such tasks are appropriate for the testing scenario.

For example, when you assess hard disk drive (HDD) performance, you should disable System Restore. HDD-intensive performance tests create additional activity for System Restore that does not represent typical end-user interactions.

The Appendix lists some of the scheduled tasks that might run during idle periods in performance tests. If the test system has timer-based tasks, do not run the performance tests when the tasks are scheduled to run. For more information about triggered tasks, see “Task Scheduler” on the MSDN® Web site.

Except during hardware-specific testing such as HDD tests, we do not recommend disabling services or scheduled tasks because this does not represent actual end-user experience. However, many scheduled tasks, such as Windows Update, can create problems if they run during a particular test pass. Ensure that you are aware of the scheduled tasks on the system and how they interact with the particular performance test.

Memory Management and SuperFetch

Both Windows Vista and Windows 7 use the SuperFetch service to observe system usage over time and to prefetch useful pages into memory. Windows XP uses prefetching to speed up boot and application start, but Windows Vista and Windows 7 use this mechanism much more often to bring programs, documents, and other data that users frequently access into memory before their actual use.

In Windows 7, SuperFetch is automatically enabled for disks that have a low Windows Experience Disk Score and disabled for disks that have a high score. During performance testing, you should use the default SuperFetch setting because it represents actual end-user experience. SuperFetch adds the prefetched pages to the system's standby page list, which has been reorganized and redesigned to retain useful data in memory over longer periods of time. Both Windows 7 and Windows Vista set priorities for pages on the standby list so that historically important pages remain in memory and less frequently used pages do not. For example, the prefetched pages of a frequently used program have a higher priority than those of a recently copied file that might not be used again.

To accurately measure performance when SuperFetch is enabled, you must test a broad range of end-user scenarios. Microsoft classifies these scenarios as either "cold" or "warm." The following describes both types of scenarios:

- Cold scenario

  In a cold scenario, the test applications are not already in memory when the test begins. Cold scenarios measure performance either after a state transition, such as boot or resume from hibernation, or after another application claims most of the available memory, such as after launching and quitting a game. To measure Windows performance in a cold scenario, reboot the system, process all the idle tasks, wait at least 5 minutes, and then measure an initial test run. If you start the test sooner, the system considers the test activities to be part of the boot process, which causes inconsistent measurements of boot performance.

- Warm scenario.

  In a warm scenario, some or all the scenario contents are in memory before measurement. This usually means that the test has run at least once during this logon session. To assess performance in a warm scenario, repeat the same test several times. Do not reboot the system or use other applications between test runs.

If Windows 7 is properly trained by previous runs, SuperFetch can prefetch to the standby list all pages that correspond to the most likely user scenario based on historical usage. This improves performance, but not as a direct result of user actions. Thus, a system in a cold-memory state is considered as remaining in a cold-memory state even after SuperFetch updates memory.

Although SuperFetch provides benefits for end users, it introduces challenges for consistent performance testing. For example, consider a test of file copy performance. The first file copy after the system boots is considered a cold file copy operation and subsequent file copies are considered warm operations. However, to duplicate the cold file copy operation, you must rename the source directory and reboot the system to prevent SuperFetch from populating memory with the source files before the test.

For more information about SuperFetch and memory performance analysis, see "Memory Sizing Guidance for Windows 7" on the WHDC Web site.

We recommend that you evaluate performance in both cold and warm end-user scenarios by measuring an initial cold test run after you boot the system and later performing warm test runs.

If you plan to test boot performance in addition to end-user scenarios, you should perform a similar type of training. Similar to the way that SuperFetch works, Windows creates a boot plan that is based on the previous five boots. Therefore, you should complete at least five training runs before you measure boot performance. The tests that are available in the current Windows Logo Kit automatically train the system before taking timing runs. For more information on testing boot performance, see "Windows On/Off Transitions Solutions Guide" and "Windows On/Off Transition Performance Analysis" on the WHDC Web site.

Performance versus Power

Starting with Windows 2000, the kernel works with the hardware to cooperatively manage processor performance and power consumption through a range of performance states. Windows chooses the optimal processor performance state and scales the performance of the system for the current workload. This approach enables lower power consumption than on previous versions of the system.

Windows 7 incorporates changes to the accounting of processor activity. These changes enable the system to more accurately determine the appropriate processor performance state and therefore result in greater energy efficiency across a broad range of scenarios.

The default power policy settings are based on extensive evaluation of end-user usage patterns. We recommend that you use the default settings when you evaluate performance. However, when you run performance tests that measure raw hardware throughput, you should use the High Performance Power Plan to ensure that the processor remains in the highest performance state at all times.

Performance Test Design Considerations

Performance tests require careful design to ensure consistent results. Tests that measure hardware performance are fundamentally different from tests that measure end-user scenarios. To design a test that provides useful information, you must understand the differences between hardware and software performance tests and be familiar with the specific situations that the test measures.

Tests that measure performance from an end-user perspective typically require some automation of user activity. In addition, it can sometimes be difficult to run these tests slowly enough to take advantage of the performance-enhancing features that are built into Windows. The following sections provide more information on designing such tests.

Automating User Presence

As described in the previous section, Windows schedules some tasks only when no user is present so that background activity does not interfere with user-initiated activity. Some performance tests send messages directly to a specific window, which bypasses the mechanism that Windows uses to detect keyboard and mouse movement. This can cause Windows to determine that the system is idle during an active portion of the test and, if so, the test will not accurately represent the actual user input scenarios. Although sending such messages can help to ensure that a test is robust, it can also introduce variability and inconsistencies in performance measurement.

Tests that evaluate the performance of end-user scenarios—instead of background tasks such as backups—should simulate user input periodically to ensure that Windows does not start background tasks.

To simulate input, call the **SendInput** function every 30 seconds. A test can use this function to move the mouse without affecting the focus window. The following example shows how a test program can use **SendInput** to simulate user input:

```
//--------------------------------------------------------//
Simulates User input
//
// This function simulates user input by producing a mouse
// move event that doesn't move the mouse. This is a benign
// event.
//
// This function should be called every 30 seconds in code
// to simulate user presence.
//
// This function returns the value from the Win32 API
// SendInput() function.
//
// Upon success, the returned value indicates the
// number of events that were successfully inserted into the keyboard
// or mouse input stream.
//
// If the function returns zero, the caller should call
// GetLastError() to collect the detailed error values
// that indicate why SendInput failed.
//
```

```
// For more information, see the MSDN documentation for
// SendInput().

DWORD SimulateUserInput()
{
    INPUT DummyInput[1];

    ZeroMemory( DummyInput, sizeof(DummyInput) );
    DummyInput[0].type       = INPUT_MOUSE;
    DummyInput[0].mi.dwFlags = MOUSEEVENTF_MOVE;

    return (DWORD)SendInput( _countof(DummyInput), DummyInput,
                             sizeof(INPUT) );
}
```

We also recommend that scripting applications run at the highest user interface privilege level.

For more information, see the following resources on the MSDN Web site:

- “Windows Vista Integrity Mechanism Technical Reference”
- “Vishal’s Windows Application Compatibility Awareness: What Is User Interface Privilege Isolation (UIPI) on Vista”
- “SendMessage, PostMessage, and Related Functions”

Testing End-User Scenarios

Microsoft designed Windows 7 with a strong focus on user responsiveness and strong performance in key user scenarios. However, many performance tests are designed to assess hardware performance capabilities instead of software performance and thus might not accurately represent end-user scenarios.

The differences between hardware and software performance tests can be subtle, but the concept is critically important. For example, consider a software performance test that runs various end-user scenarios much more quickly than an actual end user could. Such a quick test might not benefit from SuperFetch and other Windows optimizations and therefore would inadequately reflect actual end-user performance. When possible, you should run such tests at human speed.

Because Windows is optimized for end-user scenarios, the speed of a test can dramatically affect the results. When you design and run a test, ensure that you understand what the test measures so that you can use the results in appropriate comparisons.

Effects of Hardware on Performance

Several classes of hardware can affect performance:

- CPU
- RAM
- Type of storage device: hard disk drives (HDD) or solid state drives (SSD)
- Graphics subsystem

This section details how the hardware can affect performance and, where appropriate, includes any hardware-specific considerations for testing performance.

CPU

The speed of the CPU is an important factor in measuring system performance. Windows 7 can run on a wide range of CPU speeds from low-power processors in the low-gigahertz range to high-speed workstation processors. Both Intel and AMD processors are supported.

When you evaluate performance, include a range of hardware that includes multicore and 64-bit processors. Also, be aware that mobile processors behave differently when on wall power instead of battery power.

RAM

The amount of available system memory has an important effect on performance. Paging on a memory-constrained system causes disk I/O that can greatly slow responsiveness, throughput, and other measurable performance.

Performance tests often behave quite differently on systems that have minimum memory configurations and on systems that have large amounts of memory. For some performance tests, you can achieve more consistent results by running a test, adding RAM to the system, and running the test again than by running the test on a different but similarly configured system that has more RAM. SuperFetch and other Windows optimizations reduce paging and provide improved experiences on larger memory systems.

For more information on the effects of RAM on performance, see “Memory Sizing Guidance for Windows 7” on the WHDC Web site.

Type of Storage Device

The type of storage device—HDD or SSD—and its individual design also influence the results of performance tests.

Hard disks differ in speed and bandwidth and in the policies that their firmware applies for internal cache write-back and flushing. The firmware in the storage device can also affect system response times because some disks quickly begin writing data from their cache to the rotating media, whereas others retain data for much longer times or until a read operation is performed.

SSDs typically perform random reads faster than HDDs because SSDs do not have a read/write head to move. However, the typical SSD performs small random writes—and in some cases sequential writes—more slowly than an HDD. For more information about SSDs, see “Support and Q&A for Solid State Drives” on the Engineering Windows 7 blog on the MSDN Web site.

Graphics Subsystem

The capabilities of the graphics processing unit (GPU) can have an important effect on system performance, although the extent of the effect depends on the particular performance test. The amount of dedicated graphics memory, the video memory bandwidth for the graphics hardware, and the bandwidth between the GPU and system memory can also affect performance. Graphics performance tests themselves

are more sensitive than general performance tests to specific hardware features and capabilities.

Graphics hardware can use either its own dedicated physical memory or system memory. However, access to system memory is slower than access to graphics memory. Any system memory that the graphics hardware claims is no longer available to the rest of the system.

The number of monitors, the resolution of the monitors, and the number of surfaces that a performance test uses can affect the results of a performance test. Dual-monitor configurations and large, high-resolution displays can consume the available dedicated graphics memory and can place high demands on the internal video memory bandwidth of the graphics hardware. If a test creates many windows, the many surfaces from which the display is composited can place a severe strain on the operating system, because of the processing that is required in compositing and the memory that the surfaces require.

Recommended Test Methodology

The following steps constitute the recommended methodology for testing Windows performance. By following these steps, you can generate accurate, useful, and consistent performance measurements.

Step 1. Set Up the Operating System

- Install the operating system.
- Ensure that all device drivers are installed on the platform.
- Activate the version of Windows that is installed on the test hardware. Failure to do so can interfere with your results because additional activity occurs on the system when Windows is not activated.

Before you install any third-party applications, evaluate a baseline configuration with only the operating system components and the applications that are required for performance testing.

On systems that have a single disk drive, install only one instance of Windows at any given time. The location of the system files and the number of volumes that are mounted on the physical disk can change performance results.

Step 2. Set Up the Test Software

Install the test software on the system and allow time for Windows Search to index the newly installed files. To determine when the initial indexing is complete, see Step 11.

Step 3. Run Windows Update

Run Windows Update to ensure that you have installed the latest drivers and all critical and recommended updates on the test system.

After Windows Update completes, save a system image that includes the updated drivers, which represent a baseline system image against which you can compare later test results.

Drivers can have a significant effect on performance results. If you plan to compare current performance data against data from a previous version of Windows, it is a good practice to establish baseline data with the same set of drivers that you used during the earlier tests. Using the same drivers for baseline comparisons is particularly important for core components such as storage, graphics, and networking.

Step 4. Reboot the System

Reboot the system after Windows Update completes. The installation of updated drivers and applications can trigger additional system activity. The second reboot helps reduce the number of files that remain in memory from the test installation procedure.

Step 5. Download Windows Defender Definition Files

Download the most recent definition files for Windows Defender. Failure to do so can create additional system activity if Windows Defender tries to download definition files during testing.

After this step is complete, the system should be at a “known good” state with all current updates installed.

Step 6. Disable Windows Update

Disable Windows Update until all testing is complete.

Windows Update might automatically install updated drivers during a test or between tests that can cause inconsistencies among test runs.

Step 7. Calculate the Windows Experience Index

Calculate the Windows Experience Index by running WinSAT.

Do not run other programs while WinSAT is active because foreground tasks can interfere with the assessment.

To run WinSAT

- Open an elevated command prompt and issue the following command:

```
Winsat formal
```

WinSAT assesses the performance capabilities of the system and generates the Windows Experience Index. The index includes subscores for the system’s processor, memory, graphics, and hard disk. Windows uses the Windows Experience Index to determine which default theme to use and whether to enable SuperFetch.

Step 8. Reboot the System

Reboot the system and wait at least 5 minutes.

This reboot helps reduce the number of files that remain in memory from the test installation procedure. Furthermore, 5 minutes after boot is complete, SuperFetch checks the WinSAT score to determine whether to disable itself.

Step 9. Let Windows Defender Build a System File Cache

The first time that you boot the system after installing Windows and the Windows Defender definition files, Windows Defender performs an initial scan of the system. During this scan, it identifies system files that do not require later scanning and builds a cache of these files.

To check for the system file cache

Look for a file that is named MpSfc.bin in the following folder:

C:\ProgramData\Microsoft\Windows Defender\Scans\History\CacheManager

If MpSfc.bin is present, Windows Defender has already built the cache. If no such file is present, you must manually run Windows Defender to generate the system file cache.

To manually generate the system file cache

Run the following from an elevated command prompt:

```
"%programfiles%\Windows Defender\MpCmdRun.exe" –BuildSFC
```

Step 10. Disable User Account Control

User Account Control (UAC) can create challenges for testing because it prompts for user input, which can interfere with test scripts. If such problems occur, you should disable UAC prompts.

Caution: Do not disable UAC in a production or personal work environment.

To disable UAC prompts

1. In an elevated Command Prompt window, run the following command (all on one line):

   ```
   reg.exe ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
   \Policies\System /v EnableLUA /t REG_DWORD /d 0 /f
   ```

2. Reboot the system so that the changes take effect.

Step 11. Complete Indexing

Windows 7 includes an indexing service that enables Windows Desktop Search to quickly search for documents, photos, e-mail messages, and other data. The service runs by default and uses the file system's unique service name (USN) journaling feature to track changes in file system content. By default, only certain parts of the system volume are indexed. Some of the indexing service I/O occurs at low priority, so that it is delayed when normal-priority work is accomplished. If Windows 7 detects user activity such as mouse movement or keyboard input, the indexing service can reduce or delay its work.

By default, Windows 7 indexes files that are stored in user profiles, but excludes some per-user application data areas. Tests that modify files in unindexed areas of the file system produce small amounts of activity. When you install the test suite, the indexing service might initiate significant activity if many files are added to an indexed portion of the file space.

Ensure that you know which folders are indexed on the system and that the initial indexing completes before you begin testing.

To determine whether indexing is complete

1. Click **Start.**
2. Type **Indexing Options** in the **Search** box.
3. Windows displays an **Indexing Options** dialog box similar to the one shown in Figure 1. Near the top of the box, look for the **Indexing complete** message.

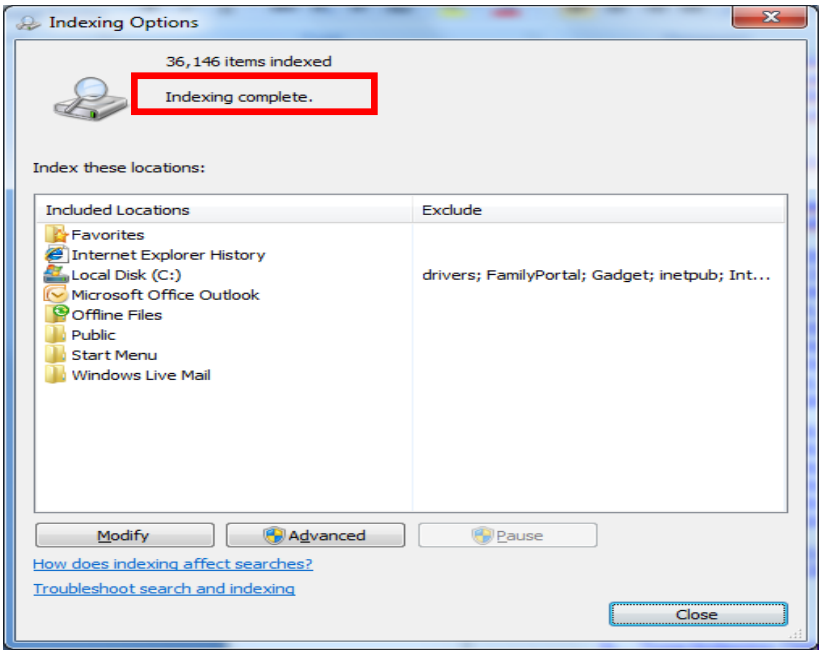

Figure 1. Indexing Options dialog box

Step 12. Use SuperFetch to Train the System

After you have installed the test software and associated data, train the system so that SuperFetch prefetches the appropriate pages.

To train the system, run the performance test several times before you collect measurements. For example, before you measure boot performance, reboot the system at least six times. Windows looks at the previous six boots to create a boot plan for the system.

If SuperFetch determines that a user is present, it observes system activity during the training runs to optimize the user experience and it ignores idle-time tasks such as disk defragmentation.

Operationally, this means that SuperFetch observes the system only if keyboard or mouse input occurs at least once every minute. Therefore, if you want to run a test when no user is present, your script must include a program that generates input events every 30 seconds to simulate true user activity. For information about creating such a program, see “Automating User Presence” earlier in this paper.

Follow these guidelines to train the system:

- To allow proper prefetching for boot and logon, reboot the system at least six times. Ensure that a user is logged on for at least 10 minutes between reboots.
- If you make significant system configuration changes, such as changes to drivers, services, or applications that start automatically, remove the \Windows\Prefetch directory and all its contents from the system drive. This removal eliminates any stale prefetched data. You must then retrain the system.
- On Windows Vista SP1 or SP2, do not run performance tests until at least 10 minutes after boot, unless the tests measure post-boot responsiveness. This delay reduces the variability in results that post-boot activities can cause. On all versions of Windows Vista, SuperFetch waits 5 minutes after boot before it populates memory with prefetched pages. In Windows 7, this delay is not required because **ProcessIdleTasks** ensures that SuperFetch runs immediately.

Step 13. Complete Idle-Time Tasks

Ensure that idle-time tasks complete by calling **ProcessIdleTasks** to trigger the idle condition on the system. To call **ProcessIdleTasks** from the command line, use the following syntax:

```
Cmd.exe /c start /wait Rundll32.exe advapi32.dll,ProcessIdleTasks
```

Do not interact with the system until the command exits. **ProcessIdleTasks** simply notifies Task Scheduler that the system is idle. Any user input during this time causes the system to exit the idle condition.

When **ProcessIdleTasks** is called from the command line, its work occurs asynchronously in the background and can take 10 to 15 minutes to complete. After it completes, you should wait an additional 5 minutes to allow the system to reach a steady state.

ProcessIdleTasks does not ensure that the scheduled idle tasks do not run during an idle period in the test. It simply notifies Task Scheduler that the system is idle.

Step 14. Disable System Restore

Before you assess hardware performance—particularly disk performance—you should disable the System Restore service. System Restore takes snapshots of disk volumes before application installations, before system updates, and at regular intervals. Most sectors in a volume are marked as copy-on-write, and therefore the system must copy them before it can update them. The copies are of the following two forms:

- Sectors that are frequently updated are copied by a low-priority background process (background “precopy”).
- All other sectors are copied on write as a synchronous process.

Both the background precopy and the synchronous copy-on-write processes can affect performance. Although the background precopy runs at low priority, it causes random I/O that can increase seek times for sequential I/O streams in foreground tasks. On disks that have relatively ineffective dynamic RAM (DRAM) cache

management, foreground application reads might be forced to wait for lower priority background precopy operations. To avoid decreasing user performance, Windows 7 slows the precopy process when a user is present by pausing 100 milliseconds (ms) between copies.

Step 15. Review Scheduled Tasks

Review the scheduled tasks that are listed in the Appendix so that you are aware of any tasks that might run during the performance evaluation.

As mentioned previously, we do not recommend that you disable these scheduled tasks, but to ensure consistent test results, you should be aware of any scheduled tasks that might run during testing. These tasks do not interfere with end users because they run only when the system is idle at specific times during the day.

Step 16. Run the Test

Run the test several times to determine the average scoring for that particular configuration.

To assess network performance, test over both wired and wireless networks. The advantage of wired networks over wireless networks is that wireless networks can introduce significant latency. If other machines, such as remote file servers, are part of the performance evaluation, ensure that you are familiar with their capabilities and configuration. For example, applications that are running on the other machine might consume resources and consequently slow its performance, which in turn affects the performance of the test system.

If the test involves a server, be aware of the network protocols that the server uses, which can affect performance. For example, if you run a test on a server that uses server message block (SMB) 1.0, the results from the same test run on a server that uses SMB 2.0 are likely to vary because of protocol differences.

When test results vary significantly from one run to another or from one configuration to another, you can use the Windows Performance Toolkit to help analyze and understand the causes of the variation. For more information or to obtain this kit, see” Windows Performance Analysis” on the MSDN Web site.

Best Practices

This section summarizes best practices for designing and running performance tests.

Best Practices for Designing Performance Tests

The following are general best practices for designing performance testing:

- ✓ In tests that evaluate performance of end-user scenarios, use the **SendInput** function to automate user presence.
- ✓ Run scripted applications at the highest user interface privilege level.
- ✓ Be aware that the speed at which your performance tests run can dramatically affect SuperFetch and other optimizations. This, in turn, can mean that your results do not accurately represent the performance that an end user would experience.

- ✓ Understand clearly what the performance test measures so that you can use the results in appropriate comparisons.

Best Practices for Running Performance Tests

The following are general best practices for performance testing:

- ✓ Call the **ProcessIdleTasks** function to help ensure that background tasks run before performance testing.
- ✓ Ensure that you are aware of the scheduled tasks on the test system and how they interact with your performance tests.
- ✓ Evaluate performance in both cold and warm scenarios. Measure an initial cold test run after booting the system and perform additional warm test runs after the system has observed the operations and behavior of the system.
- ✓ To assess boot performance, complete at least five training runs before you measure performance. Windows creates a boot plan based on the previous five boots.
- ✓ Use the default power plan settings during performance assessment unless you are measuring raw hardware throughput. Switch to the High Performance power plan to measure raw hardware throughput as opposed to typical usage patterns.
- ✓ Before you install any third-party applications, evaluate a baseline configuration with only the operating system components and the applications that are required for performance testing.
- ✓ On systems that have a single disk drive, install only one instance of Windows at any given time. The location of the system files on the physical disk can change performance results.
- ✓ Disable UAC prompts to simplify test scripts. However, do not disable UAC in a production or personal work environment.
- ✓ Be sure that initial indexing completes before starting the test.
- ✓ Test on a range of hardware including multicore and 64-bit systems. Also, be aware that mobile processors behave differently when on battery power and wall power.

Resources

White Papers on the WHDC Web Site

Memory Sizing Guidance for Windows 7
http://www.microsoft.com/whdc/system/hwdesign/MemSizingWin7.mspx

Windows On/Off Transition Performance Analysis
http://www.microsoft.com/whdc/system/sysperf/on-off_transition.mspx

Windows On/Off Transitions Solutions Guide
http://www.microsoft.com/whdc/system/pnppwr/powermgmt/OnOffTrans.mspx

MSDN

Windows Vista Integrity Mechanism Technical Reference
http://msdn.microsoft.com/en-us/library/bb625964.aspx

SendMessage, PostMessage, and Related Functions
http://msdn.microsoft.com/en-us/library/dd317330(VS.85).aspx

Task Scheduler
http://msdn.microsoft.com/en-us/library/aa383614(VS.85).aspx

Windows Performance Analysis
http://msdn.microsoft.com/en-us/performance/default.aspx

Blogs

Engineering Windows 7: Support and Q&A for Solid-State Drives
http://blogs.msdn.com/e7/archive/2009/05/05/support-and-q-a-for-solid-state-drives-and.aspx

Vishal's Windows Application Compatibility Awareness: What Is User Interface Privilege Isolation (UIPI) on Vista
http://blogs.msdn.com/vishalsi/archive/2006/11/30/what-is-user-interface-privilege-isolation-uipi-on-vista.aspx

Appendix. Scheduled Tasks

Table 1 lists some of the scheduled tasks that run at specific times during the day only when the system is idle. You should ensure that these tasks do not interfere with results for a particular test pass.

Several of these tasks run at the times that are set in Task Scheduler if the system is idle. However, if a task does not complete at the specified time, Task Scheduler schedules it to run the next time that the system is idle. These tasks do not interfere with end-user activities because they run only when the system is idle.

These tasks should have no effect on the results of performance tests that simulate user presence with actual keyboard or mouse movement and do not have idle periods.

Table 1. Scheduled Tasks

Description	Trigger Condition	When	Conditions
Scheduled Defrag	Idle	Weekly at a set time	Start only on AC power
System Restore	Idle	At system startup and daily at a set time	Start only on AC power
DefenderScan	Idle	Daily at a set time	Start only on AC power
RegistryBackup	Idle	Daily at a set time	Stop if the computer ceases to be idle

You can run most performance tests without disabling any scheduled tasks. However, you can disable a task if your particular testing setup requires it.

To disable a particular task

- Open an elevated command prompt and issue a command in the following form:

```
Schtasks.exe /Change /tn "taskFullPath" /disable
```

  Replace *taskFullPath* with the full path to the task, as in the following example:

```
Schtasks.exe /Change /tn "\Microsoft\Windows\Defrag" /disable
```