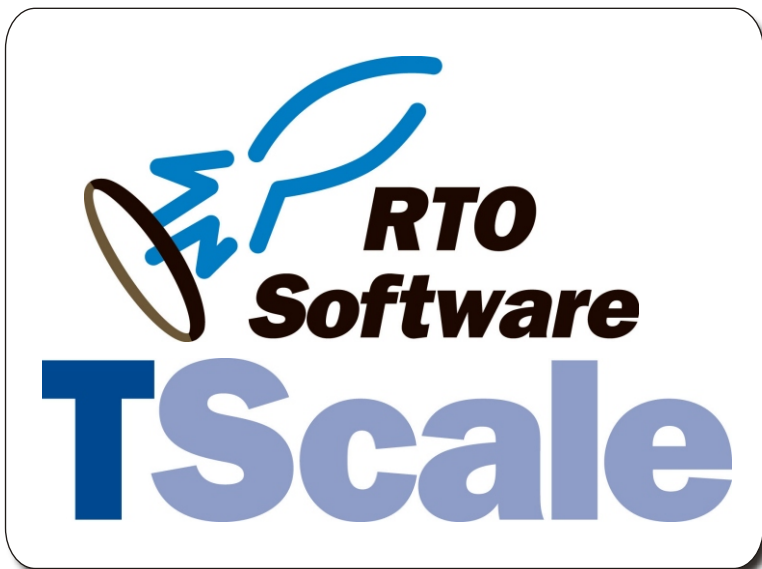


Terminal Server Performance Tuning

A real-world, step-by-step approach to getting the most out of your Citrix MetaFrame, New Moon Canaverall, or Microsoft Terminal Server systems.

Sponsored by:



The Purpose of this Paper

Whether you're experiencing slow logons, random pauses, or you'd just like accommodate more users on your server, this paper is your real-world field guide to techniques that will identify, troubleshoot, and resolve performance issues of Terminal Servers in your environment. It assumes that you've made the decision to use Terminal Services and that you would like to increase the performance of live servers that are currently up and running.

This paper is written for Windows 2000 and Windows Server 2003 Terminal Servers. These techniques apply whether you're running Citrix MetaFrame, Tarantella / New Moon Canaverall iQ, or just plain old Terminal Services.

The intent of this paper is not to discuss server sizing, nor does it cover the details of whether you should build many small servers or a few large ones. It is not a list of registry hacks that you can implement to increase the performance of your servers, and it cannot fix bad designs.

There is no magic bullet for all Terminal Server performance issues. You're not going to find a web page with a mystical registry hack or Performance counter to track. However, by rolling up your sleeves and getting under the hood of your servers, you can unleash the true potential of your existing Terminal Server systems. This paper will be your guide.

Written by:

Brian Madden



brianmadden.com

thin client industry gossip and news
downloadable "how-to" videos
product reviews
white papers
books

Author's Note

I'm a consultant and writer. I've been consulted by hundreds of companies to help them with performance issues relating to their Terminal Server environments. This practice has given me a great amount of experience in this field. This document represents the exact approach that I take whenever someone asks me to address Terminal Server performance issues at their company.

I welcome your suggestions and pointers, and will keep this document up-to-date.

Brian Madden
brian@brianmadden.com
August 2003

About the Author

Washington DC-based freelance author and consultant Brian Madden has written numerous best-selling books about thin client computing. His book *Citrix MetaFrame XP: Advanced Technical Design Guide* is now in its sixth printing. His newest book (coauthored by Ron Oglesby), *Terminal Services for Microsoft Windows Server 2003* will be published January 2004. In addition, Brian speaks at tradeshow and consults with clients worldwide. More information and a portfolio of his work are available at www.brianmadden.com.

About the Sponsor

RTO Software, founded in 2000, is the market leader of a new category of performance management tools that automatically, continuously and autonomously improve the performance, scalability and manageability of Windows-based applications, servers and desktops.

Our first product, TScale, provides a cost-effective way to improve performance, capacity and manageability for servers like Citrix MetaFrame or Microsoft Terminal Services running Windows NT and/or Windows 2000/2003. RTO Software also offers consulting services to enterprises and software firms that want to improve the performance and scalability of their applications in the terminal services environment. Based in Suwanee, Georgia, RTO's products are used in a wide variety of industries, including financial management, manufacturing, healthcare, telecommunications, and government.

For more information, or to obtain a free evaluation version of TScale, please visit RTO Software at www.rtosoft.com, or call at +1-678-455-5506.

Version Information

This document is Version 1.0, published August 2003.

Terminal Server Performance Tuning, Copyright © 2003 by Brian Madden.

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

All rights reserved. No part of this document shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

All terms mentioned in this document that are known to be trademarks or service marks have been appropriately capitalized. Brian Madden cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

| | |
|--|----|
| What is Performance?..... | 3 |
| Approaching your Performance Problem..... | 3 |
| Troubleshooting Slow Logons..... | 4 |
| Understanding the Terminal Server Logon Process..... | 4 |
| Step 1. Isolate the Problem..... | 5 |
| Step 2. Check the Roaming Profile..... | 5 |
| Step 3. Identify Anything that Runs when a User Logs On..... | 6 |
| Check the Logon Script..... | 6 |
| Check the Registry..... | 6 |
| Check the Startup Folders..... | 6 |
| Dealing with Programs that Run at Logon Time..... | 6 |
| Step 4. Identify Other Activities that Take Place at Logon Time..... | 7 |
| Step 5. Trace the Debug Logs from the User Logon Process..... | 8 |
| Getting more Users on your Server..... | 9 |
| Choosing your Software Versions..... | 10 |
| Windows Version..... | 10 |
| Citrix MetaFrame Version..... | 10 |
| Memory..... | 10 |
| Real-World Memory Estimation..... | 11 |
| How Memory Usage works in Terminal Server environments..... | 11 |
| Determining whether you have Enough Physical Memory..... | 13 |
| Identifying Memory Leaks..... | 15 |
| Page File Usage..... | 16 |
| Changing the way Windows uses the Page File..... | 17 |
| Making the Page File Faster..... | 17 |
| Page File Sizing..... | 18 |
| Processor Usage..... | 18 |
| Tracking Processor Usage..... | 18 |
| Minimizing the Processor Impact of Applications..... | 19 |
| Hyperthreading with Intel Xeon Processors..... | 19 |
| Disk Usage..... | 20 |
| Addressing Disk Usage Bottlenecks..... | 20 |
| Server Network Usage..... | 21 |
| Addressing Network Bottlenecks..... | 21 |
| Kernel Memory Usage..... | 22 |
| Understanding how the Kernel Uses Memory..... | 22 |
| Evaluating your Windows 2000 Terminal Server for Kernel Memory Usage Problems..... | 24 |
| Implementing Kernel Memory Usage Changes on Windows 2000..... | 25 |
| Evaluating your Windows 2003 Terminal Server for Kernel Memory Usage Problems..... | 26 |
| Understanding BOOT.INI Kernel Memory Usage Switches..... | 29 |
| Registry Usage..... | 30 |
| Troubleshooting Erratic Spikes, Pauses and Hangs..... | 30 |
| Step 1. Search the Web for your Problem..... | 30 |
| Step 2. Update Service Packs, Hotfixes, and Drivers..... | 31 |
| Step 3. Launch the Performance Monitor MMC Snap-In..... | 31 |
| Look for applications that are taking up 100% of the processor..... | 32 |
| Dealing with Overzealous Applications..... | 32 |
| Look for Periods when Everything Goes to Zero..... | 32 |
| Overall Sluggishness and Lack of Responsiveness..... | 33 |
| Understanding Factors that Affect Network Performance..... | 33 |
| Resolving Network Bandwidth Issues..... | 35 |
| Hardware Network Bandwidth Shapers..... | 35 |
| Removing Traffic..... | 35 |
| Squeezing ICA and RDP..... | 35 |
| Resolving Network Latency Issues..... | 36 |
| Conclusion and Final Thoughts..... | 36 |

What is Performance?

Although an odd question to ask right off the bat, this is an important one to help frame the context of what you can reasonably expect to achieve by reading this paper. Simply stated, performance (in this case) is getting expected results based on a fixed set of inputs.

Terminal Servers perform well when they meet your expectations. Whether you have 10, 100, or 1,000 users per server is not as important as *knowing* that you'll reliably have 10, 100, or 1,000 users per server.

All Terminal Server performance problems really fall within one of two categories:

- You want something to happen faster.
- You want more of something.

There are an infinite amount of things that people want to happen faster in Terminal Server environments. They want faster logons, faster application response times, and faster screen updates.

Think about the speed of different activities on your Terminal Servers. Do you have dialog boxes that you want to pop up in one-tenth of a second instead of one-half of a second? Do you want to cut the logon time from thirty seconds down to ten seconds?

Instead of wanting things to happen faster, perhaps you want more of something. In most cases, people want to accommodate more users per server.

Approaching your Performance Problem

Before you even begin to troubleshoot a performance issue, it's important to understand that every Terminal Server has a limit. This limit varies from company to company, but it's based on applications, user profiles, hardware, the network, and countless other factors. There are plenty of "finely tuned" environments in which only 25 users can be accommodated on a server. Then again, there are plenty of environments with 350 users per server.

Once you accept the fact that you'll never fit 750 users on a dual-processor server, the next step is to define your problem. This is an important step that can be fairly complex. After all, what's the *real* problem in your environment? If you have a server that is slow with 100 users, does that mean that your server is not tuned properly? Or does it mean that you have too many users on it? From a performance standpoint, those are just two different ways to look at the same problem.

Given that all Terminal Server performance issues are about the desire for "more" or "faster," your particular performance issue is likely to be one of the following:

- Logons are too slow.
- The overall environment is too slow.
- You want to get more users on your server.
- The server erratically hangs, spikes, pauses, freezes, and/or gets slow.

You might experience multiple (or all) of these problems in your environment, and quite often they are related. For this reason, it's recommended that you read through this entire paper before customizing your attack plan for your specific performance issues. Let's begin by troubleshooting slow logons.

Troubleshooting Slow Logons

All of the problems outlined in this paper can be annoying, but slow logons seem to be the curse of so many Terminal Server environments. There are plenty of real-world situations in which people are completely happy with their servers except for the fact that logons are too slow.

What exactly is a “slow logon?” It depends on your environment. Logging on to a Terminal Server and establishing a new session will never be as fast as connecting to a previously disconnected session. Some companies have logon processes that complete in a few seconds, while others take a few minutes. Unfortunately, there are some environments in which the logon process takes several minutes, even 20-30 minutes is not unheard of. This is the situation that we will troubleshoot here.

Understanding the Terminal Server Logon Process

The logon process in Terminal Server environments is interesting, especially since it doesn't necessarily relate to the usability and overall feel of the server in general. A series of events happen on a Terminal Server from the time a user clicks the “connect” button until the time his application or desktop is presented to him. In order to successfully troubleshoot slow logons, you need to fully understand what happens during the logon process. Only then can you begin to investigate which aspect of the logon process is holding things up.

From a high level, this sequence of events occurs when a user logs on to a Terminal Server:

1. The user clicks the connect button.
2. In session directory-enabled Terminal Server 2003 environments, the server routes the user to the server that is hosting the user's disconnected session.
3. The server negotiates with the requesting client for its encryption level and virtual channel capabilities.
4. The user is authenticated to the domain and his rights are checked for access to the connection.
5. The licenses are verified. The server client access license is verified first, and then the Terminal Server client access license is verified.
6. The Terminal Server loads the user profile.
 - First, it contacts the logon server (domain controller) to see if a roaming profile is configured for the user.
 - If so, the server checks to see if a local profile exists.
 - If so, it checks to see which is newer.
 - If the remote copy of the roaming profile is newer, the server copies the roaming profile from the remote server.
7. The Terminal Server applies any GPOs that have been configured.
 - The server connects back to the logon server to check for any GPOs for the user.
 - If so, the server downloads those GPOs.
 - Then, it applies the GPOs.
 - The Terminal Server checks GPO filtering, recursion, etc.
 - It then processes each GPO extension, such as folder redirection, security policy, disk quota, etc
8. The Terminal Server launches any applications as specified in the policies.
9. The server executes the contents of the “run” registry values.
10. The server runs the user's logon script(s).
11. The server runs any programs in the Startup folder of the user's Start Menu.

The above steps take place each time a user logs on. If you're using Citrix MetaFrame, then you can add more steps to address local time zone estimation and load calculations.

Now that you've seen what happens (or could potentially happen) each time a user logs on, you can start to trace this process in your environment to see where the delay could be. The most common-sense way to do this is to begin with the easy steps and move to the more difficult ones. Here's the approach I like:

1. Isolate the problem
2. Check the roaming profile.
3. Check for anything that runs, executes, or is loaded when users log on.
4. Check for any other actions that take place when users log on.
5. Trace the logon process with server debug logging options.

Step 1. Isolate the Problem

Before you can really start troubleshooting the logon process, you should get a feel for the situations in which slow logons occur. To do this, collect as much information about the symptoms of the problem as you can. Some potential questions to ask include:

- Are logons slow for some users or all users?
- If you're using Citrix MetaFrame, does a user with a slow logon via ICA also experience a slow logon via RDP?
- What happens if a user with a slow logon logs in via the server console?
- Are logons slow at all times of the day, or just sometimes?
- Do users experience slow logons every time, or is it sporadic?
- Is there anything else that the slow logon users have in common? (Are they all in the same domain group or OU? Are they all in the same building? Do they all have the same type of client device or desktop image? Are they all accountants?)

Answering these questions will help you frame your investigation. For example, if you determine that slow logons only occur via ICA and not via RDP sessions, then you'll be able to focus your efforts on the ICA session startup process.

Once you've isolated the symptoms of the problem, you can move to Step 2 with the information you need to make an intelligent diagnosis.

Step 2. Check the Roaming Profile

Although checking roaming profiles doesn't logically seem like the best place to start, roaming profiles are probably responsible for 95% of all slow logon issues in Terminal Server environments. The proper use and design of profiles in Terminal Server environments is outside the scope of this document. (See the "Resources" section at the end of this paper for more information on this topic.) If you're not using roaming profiles, skip directly to Step 2.

If you are using roaming profiles, check to see how big your users' profiles are. In theory, Windows 2000 and 2003 should not allow the master copies of your users' profiles to include space-wasting items such as temporary Internet files. In practice, however, roaming profiles can contain all sorts of garbage, including temp files, Internet cache, crash logs, and hundreds of files that start with "~."

Remember that in environments with roaming profiles, the entire roaming profile needs to be copied across the network to the Terminal Server each time a user logs on. (The only exception to this is if the user logs onto the same server they last logged out of and a cached copy of the roaming profile is locally stored on that server.)

If you do have huge roaming profiles, then you'll need to make them smaller. By using today's techniques of folder redirection, home drives, and profile folder exclusion, it's possible to design an environment in which roaming profiles never grow to more than a few mega-

bytes in size. A type of profile called the “hybrid profile” is also growing in popularity. Hybrid profiles give you the benefits of roaming profiles at a fraction of the size.

Step 3. Identify Anything that Runs when a User Logs On

If you determine that huge profiles are not causing slow logons, you should next identify everything that runs when a user logs on. In Terminal Server environments, you must check several places.

Check the Logon Script

Remember that users can get logon scripts from several locations. In addition to a script applied as part of the user account settings, you can also have scripts applied as part of GPOs. Finally, don't forget that the `usrlogon.cmd` file still exists (even in Windows Server 2003), and it's possible that some processes are being launched from there.

Check the Registry

There are several registry locations that can launch processes when a user logs on. On your Terminal Servers, check the following registry keys:

Key: `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\`

Value: `AppSetup`

Data: This is a comma-separated list of executables that run at session startup.

For most servers, this list will contain `UsrLogon.Cmd`. (Don't forget to check `UsrLogon.cmd` to see what it also might be adding to the logon process.) For Citrix MetaFrame servers, this registry value will also include “`cmstart.exe`.” This is the program that configures the post-logon Citrix environment, including drive mapping and printer redirection.

Key: `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`

Value: The name of a program to run.

Data: The path of the program to run.

`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

Value: The name of a program to run.

Data: The path of the program to run.

These two “run” keys are often used by software vendors who want to launch something at startup without giving the user the option of canceling it. This is how most of those annoying icons appear in your system tray. Each of these keys might list several programs in your environment, and that's probably fine. Just be sure you know what each one is.

Check the Startup Folders

Many people don't think to check the standard Windows Start Menu's Startup folders (both the *user's* folder and the *all users* folder). However, you'd be surprised at how many users manage to download garbage that installs itself and automatically launches via these folders.

Technically speaking, the contents of the Startup folders are launched after logon, since the Windows shell must be up and running first. However, these applications can still significantly slow the perceived logon time for your users.

Dealing with Programs that Run at Logon Time

Now that you have a list of what runs when a user logs on, what should you do? First, see if you can figure out what everything is and whether any one item is taking a long time to run. Since this is a Terminal Server, this is easy to do. Log in as an administrator and launch Task

Manager or the Performance Monitor MMC snap-in. Then, log in as a user and observe the process names that are running for that user.

What should you do if you identify something that's taking too long? Delete it. If the program is required, however, there are still some tricks you can use. The best is to let the program run in its own window that doesn't affect the window. In essence, you can turn any program into a background process.

For example, the `cmstart.exe` application from Citrix has been known to take a long time to start, especially if users have a lot of printers and printer mapping is enabled. Since the `cmstart.exe` program is invoked each time a user logs on via the "AppSetup" registry key we discussed previously, the user's session cannot start until `cmstart.exe` successfully completes. To combat this, delete `cmstart.exe` from the "AppSetup" registry location and add it to a user's logon script. If you add it to the `usrlogon.cmd` script (which ironically is also invoked via the "AppSetup" registry key), it will run for every user that logs onto the server.

Instead of adding a line to the logon script that simply reads "cmstart," add the following line:

```
start /c cmstart.exe
```

The "start" command will launch this process in its own command Window, allowing the system to move on without waiting for it to finish, and the "/c" option instructs that new window to close as soon as the command is carried out.

You can use "start /c" to launch any application from a logon script with the confidence that the application won't slow down your logon process. If you decide to do this, you should run a "start /?" and "cmd /?" from a command window, since these both have several additional options that are useful when being used in logon scripts.

The key here is that the "start /c" method of launching programs won't make them run any faster, but it will at least let them run in the background so that your users can start working sooner.

Step 4. Identify Other Activities that Take Place at Logon Time

If you can't find a program that's slowing things down by running at logon time, then you'll need to make a list of everything else that happens at logon time.

To do this, check the configuration of all your programs. Look at what's happening with the thin client (RDP and ICA) virtual channels. Do you have drive mapping? How about printer mapping or port redirection? All of these options must initialize themselves when a new session starts. While some of these options are part of the `cmstart.exe` application discussed in Step 2, other options are more native to the system and can't be tricked into running in a different way (other than disabling them).

You're probably wondering how much this can really affect the performance of the system. Consider these facts: disabling Citrix MetaFrame's client time zone estimation at startup allowed one company to increase their server loads from 65 to 110 users per server. Another company was able to increase from 35 to 62 users per server simply by disabling the automatic printer mapping. (Printer redirection takes significant time at logon as new printers are detected and installed by the spooler service. Disable that feature or check the box that allows users to logon without waiting for their printers and watch your logon times drop.)

This is not meant to be an exhaustive list of everything you should try to disable. Rather, these examples should give you some ideas of what to look for and examples of the dramatic affects they can have on logon times.

When considering the activities that take place at logon time, don't forget about non-Terminal Server and non-Citrix applications. For example, antivirus software is notorious for slowing down the logon process of a Citrix server as it scans every file of the user's profile as it's loaded. Running virus software on your master profile file server instead of your Terminal Servers has great potential to substantially decrease logon times.

Step 5. Trace the Debug Logs from the User Logon Process

If after these three steps you haven't determined why you're experiencing slow logons, it's time to get dirty. In Windows 2000 and Windows Server 2003, an application called "userenv.dll" is responsible for creating the entire user environment at logon time. This includes loading user profiles and applying GPOs.

Fortunately, you can enable diagnostic trace logging on all actions that the userenv.dll file conducts. These trace logs are *extremely* detailed, describing down to the millisecond exactly what the server was doing. For example, in addition to being able to trace the high-level logon process outlined previously, you can also see the userenv.dll verifying the profile file list build, checking for disk space, verifying ntuser.dat, loading ntuser.dat into HKCU, and replacing system variables in the path with actual variables.

By viewing this log you will see if a particular file in the roaming profile is getting stuck, if the file copy process is taking too long, or if DNS or WINS name resolution is holding the server up. It also allows you to track the application of GPOs to see if one is taking an inordinate amount of time (or if you simply have too many GPOs that are slowing things down).

This log file can help you pinpoint slow logon issues related to things you might not have thought of otherwise. For example, one environment was experiencing slow logons due to the domain controller. The IT staff had been focusing their attention on the path between the Terminal Server and the file server hosting the master copies of the user profiles. But by looking at the userenv.dll logs, they discovered that the domain controller was so busy that it was taking 30-45 seconds to respond when the Terminal Server queried it to see where the user's roaming profile was stored. A \$3,000 hardware upgrade to the domain controller saved this company 45 seconds on every user logon across 20 Terminal Servers.

You can enable userenv.dll logging by adding the following registry entry to a Terminal Server:

```
Key: HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon  
Value: UserEnvDebugLevel  
Type: REG_DWORD  
Data: 10002 (Hex)
```

The data value of 10002 will enable verbose logging to a file on the server. Once you set this value, reboot your server and check for a "userenv.log" file in the %System-Root%\Debug\UserMode\ folder. Remember to turn this off when you're done troubleshooting it, since each user logon can easily add 100KB to the size of this log file.

Let's look at a small example from a userenv.log file. This example has been severely trimmed, and shows only a few random lines to give you a feel for the type of information that is available in this log file. (The complete log for this single user's logon process would have filled 15 printed pages.)

Notice that the exact time down to the millisecond is listed for every line in this log. This allows you to see exactly what's happening and where the hold-up could be.

```

09:25:30:606 UnloadUserProfile: Entering, hProfile = <0x850>
09:25:30:606 UnloadUserProfile: In console winlogon process
09:25:30:616 UnloadUserProfileP: Entering, hProfile = <0x850>
09:25:30:616 CSyncManager::EnterLock <S-1-5-21-2364083253-1420309831-852573094-500>
09:25:30:616 CSyncManager::EnterLock: No existing entry found
09:25:30:616 CSyncManager::EnterLock: New entry created
09:25:30:616 CHashTable::HashAdd: S-1-5-21-2364083253-1420309831-852573094-500 added
in bucket 19
09:25:30:616 UnloadUserProfileP: Wait succeeded. In critical section.
09:25:30:872 MyRegUnLoadKey: Returning 1.
09:25:30:872 UnloadUserProfileP: Successfully unloaded profile
is local, not setting preference key
09:27:39:975 CreateLocalProfileImage: One way or another we haven't got an existing
local profile, try and create one
09:27:39:975 CreateSecureDirectory: Entering with <C:\Documents and Settings\brian>
09:27:40:052 CreateSecureDirectory: Created the directory <C:\Documents and Set-
tings\brian>
09:27:40:052 ComputeLocalProfileName: generated the profile directory <C:\Documents
and
09:27:42:068 CopyProfileDirectoryEx: Leaving with a return value of 1
09:27:42:106 MyRegLoadKey: Returning 00000000
09:27:42:106 IssueDefaultProfile: Leaving successfully
09:27:42:115 RestoreUserProfile: Successfully setup the local default.
09:27:42:115 SetupNewHive: Entering
09:27:42:115 SetDefaultUserHiveSecurity: Entering
09:27:42:335 SecureUserKey: Entering
09:27:42:335 SecureUserKey: Leaving with a return value of 1
09:27:42:335 SecureUserKey: Entering
09:27:42:345 SecureUserKey: Leaving with a return value of 1
09:27:49:719 ProcessGPOs: -----
09:27:49:719 ProcessGPOs: Processing extension Microsoft Disk Quota
09:27:49:719 CompareGPOLists: The lists are the same.
09:27:49:719 ProcessGPOs: Extension Microsoft Disk Quota skipped with flags 0x6.
09:27:49:719 ProcessGPOs: -----
09:27:49:719 ProcessGPOs: Processing extension QoS Packet Scheduler
09:27:49:719 CompareGPOLists: The lists are the same.
09:27:49:719 ProcessGPOs: Extension QoS Packet Scheduler skipped with flags 0x6.
09:27:49:719 ProcessGPOs: -----
09:27:49:729 ProcessGPOs: Processing extension Scripts
09:27:49:729 CompareGPOLists: The lists are the same.
09:27:49:729 ProcessGPOs: Extension Scripts skipped because both deleted and changed
GPO lists are empty.
09:27:49:862 ProcessGPOs: User Group Policy has been applied.
09:27:49:862 ProcessGPOs: Leaving with 1.
09:27:49:872 ApplyGroupPolicy: Leaving successfully.
09:27:51:763 IsSyncForegroundPolicyRefresh: Synchronous, Reason: policy set to SYNC
09:27:52:700 LibMain: Process Name: C:\WINDOWS\system32\userinit.exe
09:27:53:139 GPOThread: Next refresh will happen in 109 minutes
09:27:57:926 LibMain: Process Name: C:\WINDOWS\system32\ie4unit.exe
09:28:40:507 LibMain: Process Name: C:\WINDOWS\system32\msiexec.exe
09:28:47:967 LibMain: Process Name: C:\WINDOWS\system32\userinit.exe

```

With these tools and techniques, you should be equipped to troubleshoot slow logons. Keep in mind that the logon process always takes time, and in reality 15 or 20 seconds is probably as fast as you'll ever get it. However, if your current users are waiting a minute or two, there are steps you can take to speed up the process.

Getting more Users on your Server

Just about everyone wants to figure out how to support more users on their Terminal Servers. While this is a noble endeavor, it's crucial to remember the opening paragraphs of this paper where we discussed the limitations of server hardware.

In order to fit more users on your system, you need to look for bottlenecks. A bottleneck will always occur. You just want to see if you can find it and if it's easily fixable. If it is, you can then look for the next bottleneck. Eventually you'll encounter a bottleneck that you can't work around, but hopefully by this point you'll have a lot more users on your server than you started with.

Even the largest datacenter class servers have practical limits to the number of users they can support, especially if they are running 32-bit versions of Windows. Today's biggest 32-bit systems can support up to about 500 simultaneous users, although most servers are configured to support somewhere between 50-150 users.

In order to determine whether your servers can support more users, we'll break the server down into its technical components and analyze the techniques required to assess and optimize each one. Let's start by looking at the version of Windows software that runs on your Terminal Server.

Choosing your Software Versions

Before you begin to troubleshoot and track down the performance issues of your Terminal Servers, you should understand that the version of the software that you use on your servers can have a significant impact on its performance.

Windows Version

All things being equal, Microsoft Windows Server 2003 will support 25-80% more users than Windows 2000 Server. (This is based on studies by Gartner, HP, Unisys, Microsoft, and what I've seen with my own eyes.) This is hard to believe, and many people are skeptical until they see it for themselves, but it's proven that you can fit a lot more users on a Windows 2003 Server than a Windows 2000 server with identical hardware.

The only footnote to this rule is that Windows 2003 supports more users than Windows 2000 only when the servers are not constrained by a hardware limitation. In other words, if a server is low on memory or underpowered, then Windows 2003 and Windows 2000 will perform in identical ways. However, if you have a large server running Windows 2000 (let's say four processors and 4GB of memory), upgrading to Windows Server 2003 will enable you to run more users on that server. This is a fact.

The reasons for this are twofold. First is the fact that Windows 2003 performs much better than Windows 2000 across the board, not just with regards to Terminal Services. Secondly (and much less relevant although still interesting) is the fact that you don't need to tweak and tune Windows 2003 as much as Windows 2000. Windows 2003 includes all the Terminal Server-related tweaks that have been "discovered" in the past three years with Windows 2000.

Citrix MetaFrame Version

Citrix MetaFrame XP running Feature Release 3 is substantially faster than previous versions. In turn, this allows you to host more users on a single server. A recently published HP/Compaq ActiveAnswers paper claims that some tasks with Feature Release 3 can be accomplished in as little as 12% of the time required of Feature Release 2. Feature Release 3 introduced dozens of performance enhancements to MetaFrame XP, and all of these can add up to big gains in user load. Why spend the effort trying to manually cram more users on a server when a simple software update can do it for you?

Memory

Memory is easily the most important hardware component of a Terminal Server. The amount of memory in a server completely affects the performance of other hardware components. For example, in addition to not being able to support as many users as you'd like to, not having enough memory can add an extra burden to the processor and disk since the page file would be more heavily used.

If you really want to fit more users onto your Terminal Servers, you'll need to investigate the physical memory. There are several steps to take to be successful at this:

- You need to understand how memory works in Terminal Server environments.
- You need to figure out whether you have enough memory to do what you want to do.
- If the system seems to get slower over time, you need to check for memory leaks.

Real-World Memory Estimation

Before you even begin to think about whether a server has enough memory to support the desired number of users, it's not a bad idea to do a quick mental calculation to determine whether the server's memory is anywhere near appropriate.

The amount of memory required per user depends on the applications they use and the operating system of your server. It is typically appropriate to estimate about 128MB for the base system.

For heavy workers who use Outlook, IE, Word, and Excel and who often switch back and forth between them, a good estimation is about 15MB of memory for each user on Windows 2000 servers and 10MB per user on Windows 2003 servers. For light, task-oriented data entry users who only use a single application without the Windows shell, you can count on about 7MB for Windows 2000 and 5MB for Windows 2003. If your Terminal Servers support complex client / server line-of-business applications, then there's no way to guess how much memory you need. It could easily be 20, 30, or even 50MB per user, depending on the application.

Of course these numbers are just starting points, and your mileage may vary. Now, let's see how to validate these numbers and determine whether your server has enough memory. We'll start by reviewing how a Terminal Server uses memory.

How Memory Usage works in Terminal Server environments

Every user that runs an application on a Terminal Server will use the memory for that application just as if it were running it on a normal workstation. A quick check of the task manager shows that Microsoft Word requires about 10MB of memory to run. Each user of Word will need 10MB, meaning that 20 simultaneous users will require 200MB of memory.

Of course, this is on top of the overhead required for each user to run a session, which is about 4MB, so that 20 users running Word require a collective 280MB of memory on the Terminal Server.

To this you must add the memory required by the base operating system, which is usually around 128MB on a Terminal Server. If you add all of these together, you'll find that 20 users will theoretically require that a server have 408MB of memory.

Before you run out and start checking the memory usage of your applications, you should know the two reasons that any calculations you make based on these parameters will be totally useless:

- Applications require varying amounts of memory. Even though task manager showed Microsoft Word to only consume 10MB of memory, memory consumption will vary greatly depending on the documents that are open. Download and open a 300-page graphics-laden Windows 2003 white paper, and you'll see that Word can consume much more than 10MB. Another thing to watch out for is that the supporting files, such as DLLs, sometime consume the largest amount of memory.
- Windows treats multiple instances of the same executable in a special way. If 20 users are all using Word at 10MB each, then you would assume that 200MB of memory is being consumed, right? In actuality, Windows is a bit smarter than that. Because all 20 users are using the same copy of winword.exe, the system figures that it doesn't need to physically load the same binary executable image into memory 20 times. Instead, it loads the executable only once and "points" the other sessions to that first instance. This is done discreetly. The components controlling each user's session think that

they have a full copy of the executable loaded locally in their own memory space, when in fact all they have is a pointer to another memory space. If one session should need to modify the copy of the executable in memory, the server seamlessly (and quickly) makes a unique copy of the executable for that session.

What is particularly tricky here is the fact that if you look at the task manager, each user's session will report the full amount of memory being used. Only in the total memory usage statistics will you see that the numbers don't add up.

Even after understanding all this, there's still a bit more that we need to cover before you'll be able to get an accurate understanding of your server's real memory utilization. Most people use Task Manager to provide information as to whether a server has enough physical memory (Performance Tab | Physical Memory (K) | Available). The problem here is that the number reported by Task Manager is a bit misleading. While it does correctly show the amount of physical memory that's free, it doesn't really show *why* that memory is free. For example, if Task Manager shows that you're almost out of memory, you might think that you need more, when in fact, adding more memory won't help at all. To appreciate why, you need to understand how Windows manages memory allocation.

In the Windows operating system, individual processes request memory from the system in chunks. Each chunk is called a "page" and is 4K in size. Any pages of memory that Windows grants a process are called its "committed memory." A process's committed memory represents the total amount of memory that the system has given it, and this committed memory can be in physical memory or paged to disk (or some of both).

Windows watches how each process uses its committed memory. The pages of memory that a process uses a lot are stored in physical memory. These are called a process's "working set." This means that of all of a process's committed memory, the working set portion is stored in physical memory and the rest is stored in the page file on the hard disk.

When physical memory is plentiful (i.e. when overall system memory utilization is low), Windows doesn't pay much attention to how each process uses the memory it's been granted. In these cases, all of a process's committed bytes are stored in physical memory (meaning that each process's working set is the same size as its committed memory—even if it doesn't actively use all of it.)

However, when overall physical memory utilization gets a bit higher (80% overall by default), Windows starts to get nervous. It begins checking with all the processes to see how much of their working sets they are actively using. If a process isn't using all of its working set, the system will page some of it out to disk. This will let the system reclaim some physical memory for other processes. It's important to note that this is natural, and does *not* really affect performance, since only unused portions of a process's working set are paged to disk.

Remember that the system doesn't start reclaiming unused working set memory until overall memory utilization hits 80%. This means that in systems with plenty of memory, Windows doesn't bother paging out unused working set memory. (Why bother if you have plenty of memory, right?)

A "side effect" of this is that when memory is plentiful, looking at how much memory the system is using at any given point in time (by looking at the working set memory) will show a number that's much higher than it needs to be. This can lead you down the path of thinking that your server needs more memory than it actually does.

Let's summarize that concept again, since it's very important. The committed memory for a process is the amount of memory that Windows allocated it, and the working set is the subset of committed memory that's in the physical RAM. If a process doesn't actively use all of its working set, then the system might take away some of the working set in order to better

use the physical RAM somewhere else. This does not affect overall system performance since the process wasn't using that memory anyway.

In environments that really start to run out of memory, the system will get desperate and start to page out portions of the working set that a process is currently using. At that point, operations will start to slow down for your users, and you won't be able to add any more users. This is the circumstance we'll look for with Performance Monitor.

Determining whether you have Enough Physical Memory

Using Performance Monitor to track actual memory usage is difficult. Now that you understand (or have read, anyway) how Windows processes use memory, let's use a real-world analogy so that you can start to appreciate the complexity of the issues that you'd need to track. Refer to the following chart as you read through this short analogy:

| Windows Servers | The Analogy |
|-------------------------------------|--------------------------|
| Physical RAM | Your office building |
| Page file | Offsite Document Storage |
| A Process or Program | Employee |
| Section of memory used by a process | A Paper Document |
| The System | The boss |

If there are only a few people in your office, you'll probably have plenty of room to store all your papers. Everything—even papers you haven't touched for the past twenty years—are stored in your office since space is plentiful. Even though you're storing a lot of unused documents, it doesn't matter since there's so much space in your office.

If you get some new coworkers, they will take up some of your office space. To mitigate this, your boss might say, "Sort through your files and put anything you haven't touched in five years in offsite storage." That would free up space to allow more people to work in the office.

As your company continues to add employees, you need to provide space for them. Your boss might ask you to move documents that are only a year old to offsite storage. This is not a big problem, since you can always request your documents back from offsite storage. (Of course this takes a few days and you'd have to send something else there in its place to make room for the newly-retrieved documents.)

If your company continues to add employees, you might be forced to store documents that are only six-months old, and then one-month old. Eventually there will be so many people in such a small office that the majority of your documents will be in offsite storage, and no one will be very productive. The only solution is to get a bigger office or lay off some employees (or just convince your customers that this is how all companies work).

Supposing you are a business analyst called in to alleviate this company's document problem, how would you measure it? Of course it's obvious that they need more space or fewer people, but what is reasonable? How can you track productivity as compared to the ratio of on-site to offsite document storage? It's difficult to create hard numbers for this. After all, who can decide what's acceptable and what's not?

This difficulty is why it's hard to use Performance Monitor to track *real* memory usage on a Terminal Server. This is also why people have not traditionally been good at estimating the actual memory requirements in the past.

The best way to use Performance Monitor in this case is to track trends. You'll need to watch several counters at once to see how they relate to each other when your system starts behaving poorly. Here are the counters that you should track:

Process | Working Set | _Total

The Working Set counter is actually a property of a process, not the system memory. Therefore, you'll need to select the process object. You'll see every running process listed, but selecting the “_Total” will give you the grand total, in bytes, of all the working sets of all processes on the system. When you check this, remember that this will not include the base memory, so you'll need to add another 128MB or so.

Memory | Pages Input/Sec

This counter tells you the number of times per second that a process needed to access a piece of memory that was not in its working set, meaning that the system had to retrieve it from the page file. From our analogy, this is comparable to when an employee needed to use a document that was at the offsite storage facility.

Memory | Pages Output/Sec

This counter is the opposite of the “Pages Input/Sec” counter. It tells you how many times per second the system decided to trim a process's working set by writing some memory to disk in order to free up physical memory for another process.

Memory | Available Bytes

This counter tracks the amount of bytes that are free in the physical memory. Free bytes are ready to be used by another process. When memory is plentiful, these free bytes are used with reckless abandon, which is why you can't track this counter alone to determine memory requirements.

Terminal Services | Active Sessions

You should also track the number of user sessions you have on the system in order to have a reference point for what was going on.

So what exactly are you looking for? Start adding some users to the system. One of the first things you'll notice is that the working set will grow at a very high rate. (Remember that the system doesn't bother managing it until physical memory starts running low.) You'll also notice that the Available Bytes counter will initially take a nosedive since the system freely lets processes' working sets stay in memory. At this point you'll see some activity in both the Memory Pages counters as new users logon, but nothing to be concerned about. (Spikes may be in the 200 range, but overall fairly low.)

As you continue to add users, you'll eventually notice that the working set counter starts to drop. This occurs when the physical memory starts to run low and the system has started trimming the unused portions of processes' working sets. The Pages Output/Sec counter will start to spike high (several thousand per second even) as the system begins writing those pages to disk. As you continue adding users, the Pages Output/Sec *spikes* will decrease, although the general trend will be that Pages Output/Sec will increase. This is to be expected, and the performance of your system will not suffer because of it. (Don't buy more memory just yet!)

As you continue to add more users, the working set will continue to drop, and both memory pages counters will continue to steadily rise. Available Bytes is down around zero at this point.

What does all this mean? How do you know how many users you can add, and whether more memory will help you? If your server doesn't show the trends as described, then you have plenty of memory. The critical counter to watch is the Pages Output/Sec. Remember that it remains low for a while and then starts spiking dramatically. The spikes slowly become less and less pronounced until the counter begins rising overall. The point between spikes

dying down and the counter's slow rise is the sweet spot for a Terminal Server. If your counter never starts to rise significantly after it's done spiking, then you have enough memory, and your user base is limited by something else. If your server's Pages Output/Sec counter starts to steadily climb after it's done spiking, then you could probably benefit from more memory (or other tuning techniques outlined later in this paper).

Identifying Memory Leaks

Another problem relating to memory that can negatively impact the performance of a Terminal Server is a memory leak. A memory leak takes perfectly good memory away from the system. Most memory leaks are progressive and take up more and more memory as time goes on. Depending on the leak, this could be as slow as a few KB per hour and as fast as several MB per minute. In all cases, memory leaks are due to a problem with an application or driver, and most can be fixed with a patch (assuming the application vendor is competent enough to create one).

A memory leak is like a cancer, slowly eating away at the system. Left unchecked, it will cause the system to slow down until it becomes completely unresponsive and hangs. Memory leaks can also cause client connection and disconnection problems, excessive CPU utilization, and disk thrashing. Sometimes you can identify the offending application and kill it manually. Other times a system reboot is the only fix.

The good news about memory leaks (if there is any) is that they are very rare these days, occurring most often in "homegrown" or "really crappy" applications. It seems like they happened all the time in Windows NT, but not so much anymore. If you ever run into a consultant who's quick to suggest that all Terminal Server performance problems are due to memory leaks, you will know that this is an "old school" person who hasn't updated his troubleshooting skills in five years.

Identifying a memory leak is usually pretty easy. (Figuring out what's causing it is the hard part.) To do this, you will use Performance Monitor. In technical terms, a memory leak occurs when the system allocates more memory to a process than the process gives back to the pool. Any type of process can cause a memory leak. You can see that you're having a memory leak by monitoring Paged Pool Bytes (Memory | Pool Paged Bytes) and Page File Usage (Paging File | %Usage | _Total) with Performance Monitor. If you see either (or both) of these counters steadily increasing even when you're not adding more users to the system, then you probably have a memory leak. You might also have a memory leak if you see a more intermittent increase (still without adding new users), since memory leaks can occur in processes that aren't always running.

As we said previously, identifying that you have a memory leak is easy. Figuring out which process is causing it is harder. You can use Performance Monitor to track the amount of memory that each process is using (Process | Private Bytes | pick a process). Of course on Terminal Servers, you'll have hundreds of processes running, so it's not like you'll want to track each one. Unfortunately, there isn't an easier way to find it. (This is why IT departments need interns.) You can chart all processes at once, by selecting the "all instances" radio button on the "Add Counters" dialog box. Sometimes this works well, especially on idle servers. The hundreds of lines paint horizontal stripes across the chart, and any increase is immediately visible. When you see it, enable highlighting (Ctrl+H) and scroll through your list of processes until you find the one that's steadily increasing.

What should you do if you weren't able to isolate the memory leak with Performance Monitor? In this case the memory leak was most likely caused by something operating in kernel mode. Kernel mode memory leaks usually require the assistance of Microsoft Product Support to identify. They'll have you run a utility (poolmon.exe, located on the Windows CD in the "support" folder) that monitors the kernel mode memory pool and outputs contents to a command window.

If you do manage to figure out the cause, there's nothing you can really do about it other than to contact the vendor for a fix or to discontinue using whatever's causing it.

Page File Usage

The Windows page file is an interesting creature. A common misconception is that it's "merely" an extension of physical memory used on servers that don't have enough memory. Most people think that if they buy enough physical memory, they'll never have to worry about the page file. In Terminal Server environments, nothing could be further from the truth. While it's true that the page file is used more when physical memory is scarce, Windows also uses the page file in other ways.

Remember from the previous section that Windows is smart enough to only load a single copy of a binary executable into memory when multiple processes (or users) utilize an application. That's technically called "copy-on-write" optimization, since Windows will make an additional copy of a portion of the application in memory only when a process attempts to write to it.

In Windows environments, every executable and DLL is written to as it's used. (This doesn't mean that the EXE or DLL files on the disk are written to. It simply means that once they're loaded into memory, the versions in memory change as they are used.)

Therefore, a single DLL is loaded into memory and the system lets multiple processes share it. However, as soon as a process tries to write to a portion of that DLL, the system makes a quick copy of it (via the "copy-on-write" functionality) and lets the process write to the copy instead. Additionally, the system also backs up that section of that DLL to the page file for safekeeping. This means that there are effectively three copies of that portion of the DLL in memory—the original, the copy for the other process to write to, and the backup in the page file. This same phenomena occurs for every program that is shared by multiple process (or users), including EXE and DLL files.

In regular Windows environments, backing up the copy-on-written section of an executable to the page file is no big deal. However, imagine how inefficient this is in Terminal Server environments!

Think about a Terminal Server hosting 30 users who are all using a fat client application such as JD Edwards One World. This application is a standard client / server application, and launching the JD Edwards client software loads an executable and several DLLs into the memory space of each user's session. However, Windows only initially loads a single copy of the executables into physical memory.

As all 30 users utilize the application, Windows' copy-on-write optimization will create 29 "copies" in memory of large portions of each JD Edwards executable. (One for the first user and 29 copies for the 29 other users.) This means that Windows will have also placed an *additional* 29 copies of the original executables in the page file before the copies were made. This means that 30 JDE users will effectively cause 59 copies of the single executable to be loaded in memory. Now, imagine this multiplied by each of the many EXEs and DLLs that JD Edwards loads.

Unfortunately, this is all too common. These fat client / server applications were never really designed for Terminal Server environments. Applications like JDE OneWorld, Cerner, Lotus Notes, Siebel, PeopleSoft, SAP, and others all load massive client environments when they're launched. (This usually includes the core EXE plus several DLLs.)

Understanding this behavior starts to give you an idea of just how important the page file is in Terminal Server environments, regardless of how much physical memory you have. To help mitigate this, there are really only two things that you can do for a page file:

- You can change the way Windows uses the page file.
- You can make the page file faster.

Changing the way Windows uses the Page File

Windows' copy-on-write "optimization" is part of the core Windows memory management components, and you can't just turn it off. "Unfortunately," as Kevin Goodman puts it, "it's not like there's a 'NoCopyOnWrite' registry flag that you can use to disable it."

However, you can use third party software products to change the way that Windows uses this copy-on-write functionality. This can be done with RTO Software's TScale product, which is also sold by Wyse under the Expedian brand.

TScale watches how applications use their working sets and how multiple instances of an application are affected by the Windows copy-on-write optimizations. It logs potential optimizations to an optimization map file on the server's hard drive. Then, the next time a user launches the application, the server reads the optimization map.

This optimization allows multiple instances of an application to share the backup copies in the page file. This dramatically cuts down on page file usage, which in turn frees up the processor to support more users. TScale also decreases the working set of each instance of the application, freeing up memory that can allow you to support more users. Each application on a Terminal Server is analyzed separately, and each has its own optimization map.

TScale really shines with the big client / server applications. In fact (and quite ironically), the only applications that TScale doesn't affect too much (maybe 10% more users instead of 30% more) are applications from Microsoft, such as Office, Visio, and Project. (It's almost as if the folks writing these applications in Redmond know something about the way Windows works that no one else does.)

The cool thing about TScale is that RTO Software offers a 30-day evaluation copy that you can download from www.rtosoft.com. Therefore, you can see for yourself how much of a difference it would make in your environment.

Making the Page File Faster

Even after applying TScale or Expedian page file optimizations, your page file will still be used in a Terminal Server environment. Because of this, you need to ensure that your page file is as accessible as possible.

A heavily-used page file will overly tax the disk I/O. Therefore, refer to the "Disk Usage" section of this document for information about how to determine whether your hard disk I/O capacities are causing bottlenecks in your environment. If you determine that your page file is your bottleneck and you'd like to make it faster, there are a few things that you can do:

- Put the page file on its own drive on its own SCSI channel.
- Make sure the partition supporting the page file is FAT, since it's faster than NTFS.
- Buy one of those Flash RAM hard drives. (These look like regular hard drives except that they are solid state. They have Flash RAM instead of disks and spindles. They're very fast, except they're also very expensive, costing several thousand dollars for a few gigabytes.)

None of these solutions will make a dramatic difference, and you shouldn't even attempt them until after you've implemented a software page file optimization solution like TScale or Expedian.

Page File Sizing

The last aspect of the page file that has the ability to affect how many users you can fit on your server is the page file size. If your page file runs out of space, then you won't be able to fit any more users on your server. The "official" page file size recommendation for Terminal Server environments is 1.5 times the amount of physical memory. However, this does not need to be strictly followed. When determining your page file size, look at the types and numbers of applications that users will be using. Also consider the amount of total system memory. If you have a server with 512MB, then 1.5x page file is adequate. However, if you have 8GB of memory, you can probably get away with a smaller page file. Try a 4GB page file first and then increase from there if necessary.

You can check the percentage of page file usage via the following Performance Monitor counter:

Paging File | % Usage | _Total

Once you figure out the size that you want your page file to be, go ahead and configure your server so that the page file starts out at full size. To conserve disk space, Windows allows you to specify a minimum and maximum page file size. The system starts with the minimum and then grows from there. Unfortunately, this means that your system would need to spend resources extending the page file right when the resources are needed most. (After all, that's why the page file is being extended anyway.) Configuring the page file to start out at the maximum size (by entering the same values for the maximum and minimum sizes) will let you avoid this situation. Besides, disk space is usually plentiful on Terminal Servers.

Processor Usage

Fortunately, understanding the processor usage of a Terminal Server is much easier than understanding memory usage. There are a few simple steps that you can take to evaluate the processor and address any issues you might find.

- Understand how Terminal Servers make use of processors and how you can track there usage.
- Take steps to minimize the impact that applications have to the processor.
- If your server is running on Intel Xeon processors, understand how enabling or disabling Hyperthreading will affect performance.

Tracking Processor Usage

Tracking processor utilization is easy with Performance Monitor. Add the following two counters to your chart:

Processor | % Processor Time | _Total

This counter shows how busy the processors are. If it pegs at 100% then you need more of something. However, if the processor is too busy, don't automatically think that you need more processing power. The processor might be busy because you're running out of memory and it is spending unnecessary time writing to and reading from the page file.

System | Processor Queue Length

If you notice that the processor utilization is fairly high, you might want to track the Processor Queue Length counter as well. This counter shows how many requests are backed up while they wait for the processor to get freed up to service them. By tracking this, you can see if the processor is very busy or too busy. (Yes, there is a difference.) A processor that is very busy might show 100% utilization, but it will back down as soon as another request comes through. You can see this because the Processor Queue Length will be almost zero. A processor that is too busy might also show 100% utilization, except that because it's too busy it cannot service additional requests, resulting in the Processor Queue Length beginning to fill up.

You should never use the Task Manager to get a good understanding of the processor utilization of your system. It is meant to be used as a general estimation only, and it can be off by 5% or more at times.

Minimizing the Processor Impact of Applications

If you determine that the processing power of your server is limiting the number of users you can host, there are several things that you can do. Your overall approach will be to identify unneeded activities that are using processor resources and eliminate them.

Many applications have “features” that are enabled by default and wreak havoc on the performance of Terminal Servers. Disabling Microsoft Word 2000’s background grammar checking will allow you to *double* the number of users on a server. Even though grammar checking might not be too taxing on a single system, one hundred users with constant background checking can severely impact a Terminal Server.

The good news is that you now know what can cause unneeded processor utilization. The bad news is that these issues are impossible to detect with Performance Monitor. (With 120 users on a server, would you really know that disabling background grammar checking could take each user’s average utilization from 0.82% to 0.46%?)

Another example of this is that disabling Internet Explorer’s personalization settings will dramatically increase its loading speed and allow you to run more users.

The bottom line here is that you’ll need to manually weed through each of your applications and disable all the neat “features” that could potentially consume resources. Some people feel that they shouldn’t be forced to do this. Whether you do or don’t depends on what you want out of your servers. Do you want to fit the most users you can on a server or do you want to have shadows under your mouse cursors? You decide.

If your users can live with 256 colors instead of 24-bit color, you might be able to fit 10% more users on your servers. Remember that each option you disable won’t have enough impact its own, but multiplying its effect by several hundred users can easily produce some dramatic results.

How your users actually use their applications also affects processor utilization. If you build a Terminal Server that hosts only Microsoft Word, you’ll fit a lot more 20 WPM typists than you will 65 WPM typists.

As you analyze your application usage, don’t forget to consider the applications that run in the background on the server. A good example of this is antivirus software. Whether you should run antivirus software on your Terminal Servers is outside the scope of this document. However, understand that running antivirus software that offers “live” file system protection will severely limit the number of users you can fit on a server. Again, this is not something that you’ll be able to track with Performance Monitor. Alternately, if you can devise an alternate antivirus plan (such as antivirus protection at the perimeter and file server level), you may be able to add 20-50% more users to your Terminal Servers.

Hyperthreading with Intel Xeon Processors

If your Terminal Server systems are running Intel Xeon processors, then you have the added option of enabling Hyperthreading (via the BIOS). Hyperthreading is an Intel-proprietary technology that will make one processor look like two to the operating system (and two processors look like four, etc). Xeon processors have two data pipelines going in and out of the core processing unit on the chip. The processor generally alternates between the two pipelines. The advantage of Hyperthreading is that by having two inputs, the CPU will always have something to execute. If one pipeline is not quite ready, then the CPU can pull code from the other pipeline. This happens billions of times per second.

Hyperthreading does have some general disadvantages. One major disadvantage (which doesn't affect performance) is that many applications that are licensed based on the number of processors are tricked into thinking the system has doubled the amount of processors it actually has.

Another potential problem with Hyperthreading (which does affect performance) is that in terms of instruction execution, Windows isn't smart enough to know that you have Hyperthreading. It can't tell the difference between a two-processor system with Hyperthreading enabled and a regular four-processor system. This can lead to performance problems since the system might split up complex process across two of the four processors for faster execution. In a Hyperthreaded system, this might mean that those two threads are both going to the same physical processor, while the other processor sits idle.

Whether you should enable Hyperthreading in a Terminal Server environment depends on which operating system you're using. At the time of this writing, enabling Hyperthreading on Windows 2000 servers causes stability problems and actually *decreases* the overall performance of the system.

However, according to research conducted by Tim Mangan of TMurgent, enabling Hyperthreading on Windows 2003 Terminal Servers seems to give a 10-20% performance boost, meaning that the CPU can support 10-20% more users.

Disk Usage

In terms of performance, the hard drives of a Terminal Server are almost never the bottleneck. However, they certainly have the potential to be a bottleneck and you should certainly check them with due diligence.

Be sure to check memory and page file usage before you investigate your hard drives, since not having enough memory can cause excessive paging and your hard drives to work harder than they have to.

In most environments, your Terminal Servers only need to contain the Windows operating system, the page file, and your software application files. User and application data is usually stored on non-MetaFrame servers or a storage area network (SAN), so the local drives don't slow things down based on user file access.

In order to evaluate whether your hard drives are slowing down your server, check out the following Performance Monitor counters:

Physical Disk | % Disk Time

This counter shows you how busy your server's hard drives are. A value of 100% would indicate that the disks are 100% busy, meaning that you might need faster disks, more memory, or fewer users.

Physical Disk | Current Disk Queue Length

As with the processor counters, if your % Disk Time counter is at or near 100, you might also want to monitor this counter. It will tell you how many disk requests are waiting because the disk is too busy. If you have multiple physical disks in your server (that are not mirrored), you should record a separate instance of this counter for each disk instead of one counter for all disks. This will allow you to determine whether your disks are being used evenly, or if one disk is overworked while the another sits idle.

Addressing Disk Usage Bottlenecks

If you do determine that your server's hard drives are the bottleneck, there are a few approaches you can take. The first is to investigate your server's disk configuration. You can also opt to replace the current disks with faster ones, or perhaps even change your disk architecture altogether.

Server vendors have all sorts of tricks you can implement to increase the performance of your disks. A now infamous example is that Compaq's RAID cards and disks came with 64MB of cache that was all configured as read-cache. Changing the cache configuration (via a software utility) to 50% read and 50% write cache allowed people to almost double the number of users they could put on a system.

As a quick side note, you'll notice that many of these solutions allowed companies to "almost double" the number of users they can support. Keep in mind that this entire performance analysis is all about finding bottlenecks. In your case, implementing one change might only yield a 5% increase since it would then reveal a new bottleneck. You might have to work your way through several bottlenecks before you see substantial performance gains.

If software configuration alone won't alleviate your disk-related bottleneck, replacing your current disks with faster ones should allow you to (at least partially) release some of the pressure from the disks. Remember that when dealing with servers, you can buy faster disks (15k verses 10k RPMs), a faster interface, or both.

Finally, you might ultimately decide that changing the architecture of your disks is the best way to fix your disk problem. If you have a server with two mirrored drives, you might decide that breaking the mirror and placing the operating system on one disk and the paging file on the other is the best way to make use of your hardware.

Server Network Usage

Limitations of the physical network interfaces on Terminal Servers certainly have the ability to cause a bottleneck. What's interesting about this is that it's usually not the RDP or ICA user sessions that clog the network card. Rather, it's the interface between the Terminal Server and the network file servers that usually cause the blockage. These connections are responsible for users' roaming profiles, home drives, the Citrix IMA data store, and all other back end data that's transferred in and out of a server. Hundreds of users on a single server can easily saturate this link. As with all hardware components, if your network link is your bottleneck, you'll be limited as to how many users you can fit on your server regardless of how much memory or processing power you have.

The easiest way to check your network utilization is via Performance Monitor. Of course you can also use Task Manager (Networking Tab) to do a quick check, but you can't save anything or get any quick details from there. Within the Performance MMC snap-in, load the following counters:

Network Interface | Bytes Total/sec | Select your card

Be sure to select a different instance of this counter for each network card in your server. When you look at the results, keep in mind that a 100 Mb/second network interface is 100 megabits, and the performance counter tracks bytes. Since there are 8 bits in a byte, the performance counter would max out at 12.5M bytes. If you factor in physical network overhead, the actual maximum of a 100Mb network is about ten megabytes per second.

Network Interface | Output Queue Length | Select you card

Just like the other counters, you can determine whether you have a network bottleneck by looking at the output queue lengths of your cards. If you see a sustained value of more than two, then you need a take action if you want to get more users on your server.

Addressing Network Bottlenecks

Identifying network bottlenecks at your server level is easy. Fixing them can be as simple as putting a faster NIC or implementing NIC teaming or full duplexing to double the interface to your server.

However, you might be able to relieve some pressure from your server's network interface by adjusting the overall architecture of your Terminal Server environment. For example, if

you're using Citrix MetaFrame XP, you could install an additional network card in your server that is dedicated to the background Citrix IMA communication via a private network. Or, you might choose to build a private network connection between your Terminal Servers and your user's home drives. This would allow users' ICA or RDP sessions to flow over one interface while their data would flow over another.

If your environment involves long-haul networks or WAN links, then it's possible that a delay in that area could cause adverse performance issues. That issue is not tied specifically to the user capacity of a server though, and is discussed later in this document in the "Overall Sluggishness" section.

Kernel Memory Usage

In some cases, the performance of your system might slow down (or you might hit user limits) when it appears that plenty of hardware resources are available. What should you do when you're experiencing a major performance limit in spite of testing the memory, processor, disks, and network interface, and determining that none of them shows evidence of being the bottleneck?

By running hundreds or even thousands of processes, you're pushing the architectural limits of Windows 2000 or Windows 2003, especially on larger servers. To understand why (and how to fix it), you need to understand how Windows works.

Understanding how the Kernel Uses Memory

Have you ever wondered what the "32" means in 32-bit Windows? If you thought it has to do with 32-bit processors from Intel, you're half-right. In fact, the 32-bit Windows name is due to the fact that Windows has a 32-bit memory address space. This means that Windows can only address 2^{32} bytes (or 4GB) of memory, regardless of the amount of physical RAM installed in a system. Thinking back to your Windows training, you'll remember that this 4GB memory space is split in two, with 2GB for user-mode processes and 2GB for kernel-mode processes.

Every user-mode process has its own personal 2GB address space. This is called the process's "virtual memory," since it is always available to the process regardless of the amount of physical memory. (A common misconception is that "virtual memory" means memory that's been paged out to disk. This is simply not true when discussing the kernel.)

The kernel and other important system functions all share the same "other half" of the 4GB total memory space. This means that all kernel functions must share the same 2GB memory area.

As you can imagine, this 2GB "kernel-mode" memory space can get quite crowded since it must house all kernel-related information (memory, drivers, data structures, etc.). This means that there is effectively a limit on the amount of data-structures and kernel-related information a system can use, regardless of the amount of physical memory.

On particularly busy Terminal Servers, certain components running in this 2GB kernel address space can run out of room, causing your server to slow to a crawl and preventing additional users from logging on. This can happen in environments with plenty of memory, processors, disk space, and network bandwidth.

The good news is that you can tweak the kernel memory usage of some of the kernel's key components, allowing you to fit more users on your server. The bad news is that since all of these components share the same 2GB memory area, giving more memory to one component means that you have to take it away from another.

It's like a game, and you're looking for the right balance. Increase one area 5% and you might get ten more users on a server. Increase it 6% and you might get twenty fewer users on the server.

Let the fun begin.

Since adjusting one component affects another, we're going to start out by looking at several components together. The kernel memory area is divided into several parts, with the two major parts (called "pools") being a nonpaged pool and a paged pool. The nonpaged pool is a section of memory that cannot, under any circumstances, be paged to disk. The paged pool is a section of memory that can be paged to disk. (Just being stored in the paged pool doesn't necessarily mean that something has been paged to disk. It just means that it has either been paged to disk or it could be paged to disk.)

Sandwiched directly in between the nonpaged and paged pools (although technically part of the nonpaged pool) is a section of memory called the "System Page Table Entries," or "System PTEs."

To understand what a PTE is, (and to understand how it's relevant to Terminal Server sizing), you have to think back to what you just read about how each process can use up to 2GB of memory.

In reality, most processes don't actually use anywhere near their 2GB of memory. However, since they each think they have a full 2GB, they reference all their memory as if they were the only thing running. This means that the system must track the *actual* memory usage of every single process, and it must "translate" each process's actual memory utilization to physical memory or page file locations. To do this, the system creates a memory page table for each process.

This page table is simply an index that keeps track of the actual locations of a process's memory pages. Each entry in this table tracks a different memory page, and is called a "Page Table Entry." The 2GB of memory that the kernel uses also has a page table. Entries in that table are called "System PTEs."

Why does all this matter when you're troubleshooting the performance of a Terminal Server? Simply, the maximum number of System PTEs that a server can have is set when the server boots. In heavily-used Terminal Server environments, you can run out of system PTEs. You can use the registry to increase the number of system PTEs, but that encroaches into the paged pool area, and you could run out of paged pool memory. Running out of either one is bad, and your goal is to tune your server so that you run out of both at the exact same time. This will indicate that you've tuned the kernel's memory usage as much as possible.

How can you determine whether you're running out of PTEs or paged pool memory? That depends on whether you're using Windows 2000 or Windows 2003. It's worth repeating that you can host many more user sessions on Windows 2003 than you can on Windows 2000. The memory manager has been completely updated in Windows 2003 (technically, this update happened with Windows XP). Windows 2003 systems use kernel memory much more efficiently, and chances are that any kernel limitations you're experiencing on Windows 2000 Terminal Servers won't exist on Windows 2003 Terminal Servers.

That being said, let's take a look at how you can determine if your Windows 2000 Terminal Server is experiencing kernel memory usage problems. Remember that these symptoms occur on systems that are heavily loaded and that do *not* show *any* other signs of hardware limitations. If your processor is pegged at 100%, don't you dare try to make any of the changes outlined in this section.

Evaluating your Windows 2000 Terminal Server for Kernel Memory Usage Problems

If you're in the unfortunate position of having to cram more users onto a Windows 2000 server without the possibility of an upgrade to Windows Server 2003, there are several points to understand.

You must first see if your system is actually running low on paged pool memory or System PTEs. Windows 2000 can only support about 600MB of PTEs. In large Terminal Server environments it almost always runs out of PTEs before running out of paged pool space.

To really tune your PTE and paged pool usage, you'll need to use a kernel debugger. However, before going down that path, use Performance Monitor to see if any further steps are necessary. Fire up a Performance Monitor session and add the following counters:

Memory | Free System Page Table Entries

Quite simply, this counter indicates the number of free system PTEs. It does not show bits or bytes or anything like that, rather, it's the literal number of PTEs. When you first boot your server, this number should be high, maybe 40,000 or 80,000 or 160,000. When this number drops down below about 3,000 or so, you're in trouble.

Memory | Paged Pool Bytes

This counter indicates the total size in bytes of the Paged Pool. You could also add a counter called "Paged Pool Resident Bytes" that shows how much of the paged pool is actually in use. However, we don't really care about that because we know that the Paged Pool's size increases when it gets full. For this test we don't care how full it is. We care how big it is.

You'll also want to track one more counter that displays a component we haven't discussed yet: The System File Cache. The system file cache is the part of memory where files that are currently open are stored. Like PTEs and the Paged Pool, the System File Cache needs space in the 2GB kernel memory area.

If the Paged Pool starts to run out of space (when it's 80% full by default), the system will automatically take some memory away from the System File Cache and give it to the Paged Pool. This makes the System File Cache smaller. However, the system file cache is critical, and so it will never reach zero (which means yes, the Paged Pool can still run out of space). The system will make a trade-off and try to extend the Paged Pool as much as possible. You can view the usage of the system file cache with the following counter:

Cache | Copy Read Hits %

This counter tracks the percentage of requests to the system file cache that are actually found in the cache. For decent performance, this number needs to consistently stay at 99 or 100%. Remember that if your server starts to run out of Paged Pool space, it will "borrow" some from the System File Cache. When this happens, you'll see this Copy Read Hits % counter start to drop.

Now that you have all of these counters active, you can start adding users to your system. One of two things will happen:

- You'll run out of PTEs.
- Your Cache Hit % will consistently drop below 99%

If you run out of PTEs, then you need to increase the number that the system starts with. This will decrease the amount of Paged Pool that's available, but as your test just showed, the PTEs were your bottleneck.

On the other hand, if your Cache Hit % ratio drops consistently below 99%, then you need to increase the size of your paged pool (which will decrease the number of PTEs the system starts with).

If you continue to add users and you never run out of PTEs and your Cache Hit % never goes below 99%, then go back and reread this paper, because kernel memory tuning is not your problem.

If your server runs out of PTEs and the Cache Hit % drops below 99% at the exact same time, then congratulations! Your server's kernel memory usage is perfectly tuned, and you've effectively hit the limit as to the number of users you can support. Look at how many users were on the server when these two events occurred, subtract two or three from that number, and tell your boss you've found the architectural limit of Windows 2000 Terminal Server in your environment. If you want to support more users, then upgrade to Windows Server 2003 or implement one of the third party memory optimization tools.

Implementing Kernel Memory Usage Changes on Windows 2000

The default paged pool and system PTE levels are configured via the registry. Let's look at the system PTE entries first:

```
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\SystemPages
```

This registry value allows you to (somewhat) control the number of system PTEs that Windows creates at boot time. A value of zero lets the system create however many it needs to, and a value of FFFFFFFF (Hex) tells the system that you want it to create the absolute maximum number of PTEs that it can. (However, with Terminal Services enabled in application mode, the system will always create as many as it can anyway.) Values anywhere between these two will cause the system to create the specified number of PTEs, although the system does reserve the right to do whatever it wants if the numbers you specify are too extreme.

```
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\PagedPoolSize
```

The PagedPoolSize value specifies the size of the paged pool (duh). A value of zero will allow Windows to automatically choose the optimum size, and a value of FFFFFFFF (Hex) will tell Windows to maximize the size of the paged pool (at the expense of the ability to expand other areas, including system PTEs). A hex value anywhere in between gives Windows an idea of what size you'd like the paged pool to be, although (as with the PTEs) Windows reserves the right to ignore your setting. If you limit the paged pool to 192MB or smaller (0C000000 Hex), Windows will be able to use the extra space for other things (such as system PTEs).

Most systems will never let the paged pool get bigger than about 500MB.

There's one more registry value that you should understand here:

```
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\PagedPoolMax
```

The PagedPoolMax value specifies the maximum percentage of the paged pool that you want to be full before the system starts stealing space from the system file cache. The default value of zero will cause the system to reclaim space from the system file cache only when the paged pool is 80% full. If you want the system to wait until the paged pool is 90% full, then set this registry value to 90 (decimal). If you want the system to steal file system cache space when the paged pool is only 40% full, set this value to 40 (decimal).

The first thing you should do when tuning your kernel is to look at the values for these three registry keys. If any of these three values is not set to “0,” then you should reset them all to zero, reboot the system, and run your test again.

With later service packs (SP3 and later), a Windows 2000 Server running Terminal Server in application mode will automatically try to max out the number of PTEs that it creates. If Performance monitor shows you that your system is running out of system PTEs, you can try setting the SystemPages value to FFFFFFFF (Hex), but it probably won't do you any good.

On the other hand, if Performance Monitor indicates that you're running out of paged pool space, then you can set your PagedPoolSize value to allow for a larger paged pool. To do this, look at the values of the Pool Paged Bytes and Free System PTEs counters as soon as the Copy Read Hits % counter consistently falls below 99.

If you have fewer than 3000 free system PTEs at this point, then forget it, there's really nothing more that you can do (other than upgrading to Windows Server 2003). If you have more than 3000 free system PTEs, you could try maxing out the paged pool size by changing the PagedPoolSize registry value to FFFFFFFF (Hex). Reboot your server and run your test again to see if your changes made a difference for the better.

The risk you run here is that your optimal system PTE and paged pool sizes might be somewhere in the middle of maxing out one or the other. The only way to avoid this is to use a kernel debugger to view the actual size of the paged pool. Instructions for kernel debugging follow in the next section of this document. Using the kernel debugger in Windows 2000 environments is similar to using it in Windows 2003 environments. The only major difference is that Windows 2000 does not allow local debugging. Therefore, you'll have to find a laptop with a serial port and hook it up to the back of your Terminal Server. (Full instructions for doing so are outlined in the debugger's documentation.)

Evaluating your Windows 2003 Terminal Server for Kernel Memory Usage Problems

The general concepts outlined in the previous section about Windows 2000 Terminal Servers also apply to Windows 2003 Terminal Servers. (You should quickly read through that section even if you only have Windows 2003 servers.) However, there are several architectural changes with regard to memory management that will make your life much easier when using Windows Server 2003.

First, the number of System PTEs in Windows Server 2003 has been increased to a maximum of about 1.3GB. This is double the maximum available with Windows 2000.

Your Windows 2003 server will automatically use the maximum number of PTEs, as long as you:

- Boot the server without the /3GB option. (More on this later.)
- Do not have the registry key set that allows for this RAM to be used as system cache (since it will be used for System PTEs instead)
- Do not have registry keys set that make session space or system mapped views larger than the default size (48MB).

Furthermore, the Windows Server 2003 memory management enhancements also cause substantially less paged pool to be used.

If you feel that you might be running out of PTEs or paged pool on a Windows Server 2003 Terminal Server, there's an easy test to do. Instead of using the Performance Monitor, you can get a much more accurate snapshot of kernel memory usage with a kernel debugger.

Contemplating a kernel debugger probably conjures up nightmares about host machines and serial cables and all sorts of things that developer-type people handle that you probably never ever thought you would have to. Fortunately, times have changed.

In Windows Server 2003, using the kernel debugger is easy. It's windows-based, and you can even use it to debug the computer that it's running on. The kernel debugger can tell you amazing things about the state of the Windows kernel, which is exactly why we're going to use it here.

The first thing you have to do is to download the Debugging Tools for Windows. They're available from:

<http://www.microsoft.com/whdc/ddk/debugging/>

Go ahead and install them on the Terminal Server that you're testing. Choosing the default options should be sufficient in this case. Then, fire up the Windows Debugger (Start | All Programs | Debugging Tools for Windows | WinDbg).

Next, you'll need to establish a kernel debugging session with the local computer.

1. Choose File | Kernel Debug...
2. Click the "Local" tab.
3. Click "OK."

Now, before you can effectively use the debugger, you need to tell it where your "symbol" files are. Symbol files are files with the .SDB extension that tell the debugger how to interpret all the information that it's getting from the system. Without symbol files, the debugger is useless.

Information about obtaining symbol files is available from the Microsoft Debugging Tools for Windows webpage referenced earlier in this document. At the time of this writing, there are two ways to use symbol files with the Windows Debugger.

- You can download the symbol files to your hard drive (or copy them from the Windows Server 2003 CD). Then, you configure the Debugger so that it knows where to look for them.
- You can use Microsoft's new "Symbol Server," which allows the Debugger to automatically and dynamically download the required symbol files from Microsoft's web site.

Using the symbol server is by far the easiest option and is the one you should definitely choose unless your Terminal Server does not have direct Internet access. Configuring your Windows Debugger to use Microsoft's symbol server is easy:

1. Choose File | Symbol Path File in the Debugger.
2. Enter the following statement into the box:
`SRV*c:\websymbols*http://msdl.microsoft.com/download/symbols`
("c:\websymbols" can be changed to an appropriate local storage path for your environment.)
3. Check the "Reload" box.
4. Click "OK."

Now you're ready to begin debugging. Have your users log into the system. Then, when your system begins to slow down, enter the following command into the "lkd>" box at the bottom of the debugger screen:

```
!vm 1
```

This will display a snapshot of the kernel's virtual memory (its 2GB memory area) usage, which should look something like this:

```
lkd> !vm 1

*** Virtual Memory Usage ***
Physical Memory:    130908    ( 523632 Kb)
Page File: \??\C:\pagefile.sys
Current:    786432Kb Free Space:    767788Kb
Minimum:    786432Kb Maximum:    1572864Kb
Available Pages:    45979    ( 183916 Kb)
ResAvail Pages:    93692    ( 374768 Kb)
Locked IO Pages:    95    ( 380 Kb)
Free System PTEs:    245121    ( 980484 Kb)
Free NP PTEs:    28495    ( 113980 Kb)
Free Special NP:    0    ( 0 Kb)
Modified Pages:    135    ( 540 Kb)
Modified PF Pages:    134    ( 536 Kb)
NonPagedPool Usage:    2309    ( 9236 Kb)
NonPagedPool Max:    32768    ( 131072 Kb)
PagedPool 0 Usage:    4172    ( 16688 Kb)
PagedPool 1 Usage:    1663    ( 6652 Kb)
PagedPool 2 Usage:    1609    ( 6436 Kb)
PagedPool Usage:    7444    ( 29776 Kb)
PagedPool Maximum:    43008    ( 172032 Kb)
Shared Commit:    1150    ( 4600 Kb)
Special Pool:    0    ( 0 Kb)
Shared Process:    2939    ( 11756 Kb)
PagedPool Commit:    7444    ( 29776 Kb)
Driver Commit:    2219    ( 8876 Kb)
Committed pages:    63588    ( 254352 Kb)
Commit limit:    320147    ( 1280588 Kb)
```

Your output may be a bit different from this sample. (For example, multi-processor systems will have five paged pools instead of the three shown here.) Out of all this stuff on the screen, you only need to care about three lines:

- Free System PTEs
- PagedPool Usage
- PagedPool Maximum

If your PTEs are really low, then you'll need to try to increase them. If your paged pool usage is almost at the paged pool maximum, then you'll need to increase it. If neither of these is true, then cram a few more users on your system and run the "!vm 1" debugger command again.

To increase your PTEs, change the value of the SystemPages registry location (referenced in the previous section) to FFFFFFFF (Hex). Reboot the server and run the test again. Quite honestly, however, this probably won't change anything, since a Windows 2003 Terminal Server tries to maximize the number of PTEs. However, manually setting this value is still worth a shot since it will make maximizing the PTEs a priority for the system.

If your paged pool usage is within a few percentage points of the paged pool maximum, then you'll want to increase the maximum size. You can only do this, however, if you have plenty of system PTEs available. (Let's say at least 3000.)

Each PTE is 4K in size. For every PTE that you can afford to lose, you can add 4K to the size of your paged pool. If you have 10,000 free system PTEs, you can probably afford to lose 7000 (leaving you with 3000). If you have 14,000 free, you can afford to give up 11,000.

Take however many PTEs you can afford to give up and multiply that number by 4. Then, add that number to the size of your PagedPool Maximum as shown in the debugger. (Be sure to add it to the Kb value to the far right.) That number will be the size that you should set your paged pool to be. Open up your registry editor and set your value in the following location:

```
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\PagedPoolSize
```

This registry location stores the value in bytes, not kilobytes. Therefore, multiply your calculated value by 1024 to get the number that you should enter here. When you enter the value, be sure to enter it in decimal format. The registry editor will automatically convert it to Hex format.

At this point you can reboot your server and rerun your test to see if it made a difference. Keep in mind that Windows will override your manual setting if it doesn't like it, so it's possible that your registry editing won't change anything at all.

Understanding BOOT.INI Kernel Memory Usage Switches

While we're still on the topic of kernel memory usage, we should take a second to address the various boot.ini file switches that can be used in Terminal Server environments. If you do an Internet search on performance of Terminal Servers, you'll come across different theories about how these switches should be used. Let's debunk the theories and look at how these switches really work.

There are two boot.ini switches with which you should be familiar: /3GB and /PAE.

The /3GB Switch

As you recall, 32-bit Windows systems can address 4GB of memory, and that memory is split into two 2GB chunks, one for the kernel and one for each process. Quite simply, adding the "/3GB" switch to a boot.ini entry changes the way the server allocates the 4GB memory space. Instead of two, 2GB sections, using the "/3GB" switch changes the partition so that the kernel gets 1GB and each process gets 3GB.

This is useful for memory-hungry applications such as Microsoft Exchange or SQL Server. As you know, however, Terminal Servers have enough trouble with the Windows kernel due to the fact that it's by default limited to 2GB, and as you can imagine, limiting the kernel to only 1GB of virtual memory would have disastrous consequences in a Terminal Server environment. (Besides, in order to use the full 3GB, an application has to be compiled in a special way, so it's not like adding the "/3GB" switch would affect "regular" applications anyway.)

If your Terminal Server is booting up via a boot.ini entry with the /3GB switch, remove it immediately.

The /PAE Switch

The "Physical Address Extensions" (PAE) boot.ini switch is used when Terminal Servers have more than 4GB of physical memory. Since the 32-bit Windows operating system can only address 4GB of virtual memory, systems with more than 4GB have to perform some fancy tricks to be able to use the physical memory above 4GB. These "fancy tricks" are enabled by adding the "/PAE" switch to the entry in the boot.ini file. If you're using a server with more than 4GB of RAM, then be sure that you have the "/PAE" switch in your boot.ini file. (In order to use more than 4GB of physical memory, you'll have to use Windows 2000 Advanced Server or higher or Windows 2003 Enterprise Edition or higher.)

Registry Usage

If your Terminal Server is running on Windows 2000, it's possible that you can't host any more users because you ran out of space in the registry. The registry must load an HKU hive for every single user who's running a session on the server. Fortunately, checking for the size of your registry is easy with the Performance Monitor. All you have to do is load the following counter:

System | % Registry Quota in Use

This counter gives you the percentage of your server registry's maximum size that's currently in use. If this counter indicates that the registry is nearing its maximum size, then you might be able to add more users to your server by configuring it to allow for a larger registry.

However, there are potential consequences to this action. On Windows 2000 Terminal Servers, all of the registry's files are stored in the kernel's paged pool. This is a problem for two reasons:

- It means the registry has a maximum size of about 160MB.
- Increasing the size of the registry decreases the amount of paged pool that's available for other kernel processes, potentially causing the paged pool to become your sizing bottleneck.

If you do need to adjust the size of the registry on Windows 2000, you can do so via the Control Panel. (Control Panel | System | Advanced Tab | "Performance Options" Button | "Change..." Button | "Maximum Registry Size" box)

If your Terminal Servers are running on Windows Server 2003, then you have nothing to worry about. Windows 2003 does not store the registry in the paged pool, meaning that there is effectively no size limit of the registry. Because of this, there is no registry size limit setting in Windows 2003. The registry only consumes 4MB of the paged pool space regardless of how large it actually is.

Troubleshooting Erratic Spikes, Pauses and Hangs

The erratic freezes and pauses that sometimes occur on Terminal Servers are probably the most annoying and difficult performance problems to troubleshoot. The good news (and the bad news) about these kinds of problems is that they are almost never your fault. The resolution usually points back to a device driver, service pack, or hotfix of some sort.

Erratic issues usually fall into two categories:

- An application or process pegs the processor at 100% utilization for a short time and then returns to normal. During this time, users' sessions are usually unresponsive.
- The server just freaks out for a few seconds. Every so often everything freezes, including the Performance Monitor MMC snap-in. Then after a few (or even 20 or 30) seconds, the performance chart jumps ahead in time to the current position. However, the "blackout" period causes a blank, with all performance counters showing zeros or no data.

The best attack plan for these types of problems is as follows:

1. Check the web for your specific problem.
2. Update service packs, apply hotfixes, and/or update device drivers.
3. Use Performance Monitor to check for anomalies.

Step 1. Search the Web for your Problem

Solutions to the erratic problems are almost never intuitive, so it's worth it to spend ten or fifteen minutes on the web to gain an understanding of your problem.

For example, Windows Server 2003 was released in April 2003. In July, Microsoft KB article 821467 was published with a title “Windows Server 2003 Terminal Server Stops Responding.” This article indicated that the problem only happens with Windows 2003 Servers, and that a fix is available from Microsoft. Why bother troubleshooting on your own when someone else might have done it already?

These are the websites that I search in the order that I search them:

1. Google groups (groups.google.com). You’re best luck usually comes from the Microsoft news groups. Often times the Microsoft MVPs keep these lists up to date with hotfixes, and they’re usually faster than Microsoft KB articles.
2. Microsoft Knowledge Base (www.microsoft.com/support).
3. The THIN list archives (www.thethin.net). The archive searching tool on the THIN’s main website is sometimes awkward to use, so you might try searching the archives via their listserv provider. (www.freelists.org/archives/thin)
4. Citrix Support Knowledge Base (support.citrix.com/kb). The search engine never seems to find what I’m looking for, but this site is a requirement for locating Citrix hot fixes.

Step 2. Update Service Packs, Hotfixes, and Drivers

Most of the erratic problems have already been fixed at some point. Do a quick search on the Microsoft Knowledge Base for “server stops responding” and you’ll see that 90% of the solutions say “obtain the latest service pack or hotfix.”

This is serious. For example, Service Pack 2 for Windows 2000 fixes a problem with the registry cache locking. This problem usually occurs on busy Terminal Servers, and causes the whole system to pause if any registry writes need to be made while the registry is being backed up (which Windows does periodically). Applying Service Pack 2 or newer completely eradicates the problem.

However, you also need to be careful about updating production servers. When Service Pack 4 for Windows 2000 first came out in August 2003, it broke a lot of people’s Terminal Server environments. Check the web resources listed in Step 1 to make sure that whatever patch you’re applying is safe. Also, apply the patch to a test server before putting it on a production server.

While you’re at it, you should also update the hardware device drivers and firmware. There have been countless cases in which hard drive firmware or driver updates have magically fixed the occasional hiccup. Keep in mind that the users in a Terminal Server environment really push a server to its limits, and your hardware (and the drivers) is definitely getting its exercise.

Step 3. Launch the Performance Monitor MMC Snap-In

If web searching and server patching didn’t fix your erratic problems, you’ll have to continue the investigation yourself. Chances are that you’ve already fired up Performance Monitor. If you’re still having the problem, you’ll need to have it active during one of the glitches.

Add counters for your processors (Processor | % Processor Time). It’s best to add one counter for each processor instead of the “_Total,” since that will allow you to more easily see whether a single process pegs the CPU.

If your problem is extremely intermittent, don’t forget that you can configure an alert to watch for it. Then, you can configure that alert to automatically start a performance data log. The only problem with this is that you’ll need to configure your alert to check at a frequent interval—maybe every few seconds. Be careful that you don’t create a catch-22 situation where your complex monitoring, logging, and alerting schemes actually tax the system more.

The most ideal situation is for you to be able to view the problem live. (You could also configure Performance Monitor on a remote computer to track your Terminal Server.)

Look for applications that are taking up 100% of the processor

This is probably fairly obvious, but you need to look at what exactly is happening when your system slows down. Does the CPU spike? If so, there are a few different approaches you can take. First of all, try to determine what's causing the spike. Is someone's roaming profile loading? Did a bunch of users just logon? In some cases, you may encounter an application or process that takes up too much CPU utilization.

Dealing with Overzealous Applications

The easiest solution in these cases (especially in Terminal Server environments) is to add CPU throttling software to your server. This software monitors all running processes and clamps down on anything that hits a predefined limit. This is helpful for applications that aren't really Terminal Server-friendly but that your users insist on using.

There are many vendors that make products to help ensure that CPU resources are available when they need to be. Here are some of the more popular tools:

- Appsense Optimizer (www.appsense.com)
- RES PowerFuse CPUShield (www.respowerfuse.com)
- TAME (www.tamedos.com)
- TMuLimit (<http://www.tmurgent.com/TMuLimit.htm>)
- ThreadMaster (<http://threadmaster.tripod.com>)

Each of these tools approaches CPU over-utilization in a different way, so you should definitely investigate all of them. There are situations in which one of these tools will fail to control a process while another tool works. If you don't get the results you need with one tool, it's definitely worth trying another.

Look for Periods when Everything Goes to Zero

Hopefully searching the web, patching your server, and updating your drivers and firmware will alleviate any sporadic problems that you were having. However, if you're still experiencing performance issues, there are a few more things left to try.

In some cases, you'll notice that all Performance Monitor counters will just disappear for a few seconds and everything pauses. When the system comes back, Performance Monitor has jumped ahead, with nothing but zeros left in the twilight zone.

In other less extreme cases, you might not notice anything strange in Performance Monitor. In fact, everything might look normal even as your system takes a performance hit. In these cases, you'll need to continue stepping through this document.

As with the previous problems, you'll also need to consider everything that's happening on your server. One company's Terminal Servers would max out at 25 users, even though Performance Monitor showed no problems. It was later discovered that this was due to the fact that they were redirecting the users' "Application Data" folder to a remote home drive location. By redirecting this critical profile folder, the server would slow to a crawl each time a user needed to access data from that folder (a very frequent event). Stepping through the performance troubleshooting steps caused them to consider everything that was happening on their server, and ultimately led them to experiment by turning off folder redirection. Thus they were able to isolate their problem.

Overall Sluggishness and Lack of Responsiveness

This paper has previously focused on the fact that overall sluggishness of your Terminal Servers could be related to having too many users on a server. However, it's often the case that only some of your users will experience sluggish sessions, while others will not. In these cases the issue can usually be traced back to poor network performance. Before we explore the details of how you can tune your network, it's important to review some basics of network performance.

Understanding Factors that Affect Network Performance

When considering network performance, you need to understand the difference between "latency" and "bandwidth." Both are used to describe the speed of a network. Bandwidth describes how much data can pass through the network in a given period of time, such as 10 megabits per second, or 256 kilobits per second. Latency describes the length of time, usually expressed in milliseconds (there are 1000 milliseconds in one second), that it takes for data to get from point A to point B. Bandwidth and latency are independent of each other.

The fact that bandwidth and latency are different from each other is an important concept to understand in Terminal Server environments. (This is so important that it calls for another analogy.)

Imagine that each packet of data is an automobile, and the network is a highway. In order for the data to get from point A to point B, an automobile would have to travel from one end of the highway to the other. In high-bandwidth environments, the highway has many lanes, and hundreds of automobiles can be on it at the same time. In low bandwidth environments, the highway is a narrow country road, and only a few automobiles can fit on it at the same time. The width of the highway is like the bandwidth of the network. Since latency affects how long it takes data to traverse the network, the speed of the automobiles on the highway represents the latency. Even on narrow highways (low bandwidth), there might be people who drive really fast and travel the road quickly (low latency). Conversely, even if you have a large number of automobiles on a wide highway (high bandwidth), the drivers may choose to drive slowly (high latency).

Bandwidth and latency are independent of each other because the width of the highway does not directly affect how fast you drive on it. However, as you've probably guessed by now, it's possible that bandwidth *can* affect latency.

Now imagine a low-bandwidth environment that also has low latency (a narrow highway where the people drive really fast). If there are only a few automobiles on the road, they can go fast without problems. However, imagine that the highway begins to fill up with more and more autos. Even though the people want to drive fast, they can't because the highway is too crowded. The effect is that it will take longer for automobiles to get from one end of the highway to the other. In a sense, the latency has increased from low latency to high latency simply because there are too many vehicles on the highway.

There are several solutions to the overcrowded highway problem. You could:

- Widen the highway.
- Remove some of the vehicles.
- Force people to drive smaller vehicles.
- Install traffic signals and lane control devices to manage the traffic.

As you'll see, the four potential solutions to the overcrowded highway problem are also the four potential solutions to overcrowded networks in Terminal Server environments.

Back in the real world, a network connection's bandwidth and latency each affect Microsoft RDP or Citrix ICA session traffic in different ways:

- Bandwidth affects how much data the session can contain. Higher resolution sessions require more bandwidth than lower resolution sessions. Sessions with sound, printing, and client drive mapping all require more bandwidth than sessions without. If a particular session only has 15Kbps of bandwidth available, that session can still have decent performance so long as the resolution, color depth, and other virtual channel options are tuned appropriately.
- Latency is usually more critical in Terminal Server environments. Since latency affects the amount of time it takes for communication to pass between the client and the server, environments with high latency can seem like they have a “delay” from the user’s perception.

For example, imagine an environment in which a user was using Microsoft Word via a remote RDP session. When they press a key on their client device, the key code is sent across the network to the Terminal Server. The server processes the keystroke and prepares to display the proper character on the screen.

Because this is a Terminal Server, the screen information is redirected back across the network where it is displayed on the local client device. In order for a character to appear on the screen, data must travel from the client to the server and then from the server back to the client again.

In this situation, if the latency of the network is 10ms, the network delay will only add 20ms (because the data crosses the network twice) to the time between the key press and the character appearing on the user’s screen. Since 20ms is only 0.02 seconds, the delay will not be noticeable. However, if the latency was 200ms, the total delay to the user would be 400ms, or almost one-half of a second. This length of delay would be noticeable to the user, and would probably be unacceptable.

An easy way to get an approximation of the latency in your environment is to perform a TCP/IP ping. You can ping the server from the client or the client from the server, it doesn’t matter which way. For example, if your server is called “server01,” you would execute the following command from a client workstation:

```
ping server01
```

(Be sure that you execute the ping command locally on the workstation, not via an RDP or ICA session on the server.) The results will look something like this:

```
Pinging server01 [10.1.1.42] with 32 bytes of data:
Reply from 10.1.1.42: bytes=32 time=378ms TTL=118
Reply from 10.1.1.42: bytes=32 time=370ms TTL=118
Reply from 10.1.1.42: bytes=32 time=360ms TTL=118
Reply from 10.1.1.42: bytes=32 time=351ms TTL=118
Ping statistics for 10.1.1.42:
    Packets: Sent = 4, Received = 4, Lost = 0
    Approximate round trip times in milli-seconds:
    Minimum = 351ms, Maximum = 378ms, Average = 364ms
```

Notice that the “time=” section of each line shows you the approximate latency. This time is the time that the “pinger” waited for a response from the “pingee,” meaning that the time shown represents the entire round-trip. The above scenario with the 364ms latency could have occurred in a dial-up environment with a bandwidth of 28kbps or a frame-relay environment with 512kbps. In either situation, the performance would not be as good as in an environment with less latency.

Resolving Network Bandwidth Issues

Once you've determined whether your network performance issues are bandwidth-related or latency-related, you can begin to address them. If your network suffers from a lack of bandwidth:

- Install a hardware device to monitor and control applications and bandwidth. This is like adding a traffic cop and traffic signals in our highway example.
- See what type of traffic you can remove from the network. This is like removing extra automobiles in our highway example.
- Make the ICA or RDP sessions as "small" as possible. This is like convincing everyone to drive smaller cars.

Let's take a look at how you could implement each one of these three solutions.

Hardware Network Bandwidth Shapers

The two most popular types of bandwidth management devices in Terminal Server environments are Packeteer's PacketShaper (www.packeteer.com) and Sitara's QoSWorks (www.sitaranetworks.com). Both are physical hardware devices that sit between your network and the WAN router.

These devices allow you to analyze and capture current traffic usage (which is when you'll discover that 75% of your WAN traffic is web surfing). You can then give Citrix ICA or Microsoft RDP traffic priority over other types of traffic. You can even configure these devices to give different priorities to different IP addresses. You can also guarantee certain amounts of bandwidth to ICA or PRD (or any protocol).

These third-party devices are similar to Cisco Quality of Service (QoS) devices, except that the Sitara and Packeteer devices are "Layer 7" routers and can differentiate ICA and RDP traffic from other types of traffic.

Removing Traffic

One of the easiest things you can do to free up bandwidth for your ICA or RDP sessions is to remove as much non-Terminal Server traffic as possible. Are you backing up data across the network? Are your users downloading MP3s? All of this takes bandwidth away from your ICA and RDP protocols. (Be sure to read the next section about hardware bandwidth shapers for tips on how to find out what's happening on your network.)

Squeezing ICA and RDP

Once you've removed any unnecessary traffic from your network, you can start to think about squeezing the ICA and RDP protocols down as small as possible. There are several steps that you can take to do this:

1. First, turn off as much desktop "glitz" as possible. This means disabling wallpapers, menu animations, and desktop themes.
2. Next, disable any RDP or ICA virtual channels that you're not using, such as printing or local drive access.
3. You should also configure the users' sessions so that they're using the minimally required settings, such as the lowest resolution and color depth needed.
4. If you haven't done so yet, be sure that your Terminal Servers are running the most recent versions of service packs and hotfixes. For example, SP3/FR3 for MetaFrame XP used in conjunction with the ICA version 7 clients produce substantial speed increases.

Each of these steps will help to minimize the size and network utilization of individual user sessions.

Resolving Network Latency Issues

If you determine that your network performance issues are due to high latency, there are also some steps that you can take to help improve performance:

- If you're using Citrix MetaFrame, enable "SpeedScreen" technology. This technology uses different kinds of caching and character generation methods to make user sessions appear faster in highly-latent environments.

If you have high latency, don't forget (as shown in our highway example) that freeing up some bandwidth might also have the positive effect of lowering your overall latency.

Conclusion and Final Thoughts

Hopefully after reading this paper you were able to increase the performance of your Terminal Servers. Keep in mind that performance tuning is always an art, and that no server is perfect. You simply play the game by moving from one bottleneck to the next, and eventually you get to the point where you feel that they're "good enough."

I plan on continually revising and updating this document and you'll always be able to find the most recent version at www.brianmadden.com. Feel free to email any comments to me at brian@brianmadden.com.

If you like what you've read, you might consider buying one of my books about designing server-based computing solutions. They're self-published and written in the same style as this document. My two current books are:

Citrix MetaFrame XP, Advanced Technical Design Guide, by Brian Madden, published November 2002. BrianMadden.com Publishing Group, ISBN 0971151040

Terminal Services for Microsoft Windows Server 2003, Advanced Technical Design Guide, by Brian Madden and Ron Oglesby, due out in January 2004. BrianMadden.com Publishing Group, ISBN 0971151040.

These books are available at most bookstores or online at Amazon.com.

References

Most of the techniques, processes, and approach detailed in this article were based on my own consulting experience and methodologies. However, I did quite a bit of research on some of the more obscure areas of Terminal Server performance. When writing this paper, I used the following materials from the Internet:

Citrix MetaFrame XP, Advanced Technical Design Guide, Brian Madden, BrianMadden.com Publishing Group, November 2002.

Citrix MetaFrame XP Presentation Server with Feature Release 3 on HP ProLiant Servers, HP ActiveAnswers White Paper, August 6, 2003. activeanswers.compaq.com.

How to Configure the Paged Address Pool and System Page Table Entry Memory Areas, Microsoft Knowledge Base Article 247904.

HOWTO: Map Adapter RAM into Process Address Space, Microsoft Knowledge Base Article 189327.

Large Memory Support Is Available in Windows 2000 and Windows Server 2003, Microsoft Knowledge Base Article 283037.

Microsoft Windows Server 2003 Terminal Server Capacity and Scaling, HP ActiveAnswers White Paper, April 24, 2003. activeanswers.compaq.com.

Should you enable Hyper-Threading on Server 2003?, June 25, 2003, Tim Mangan, TMurgent Technologies, http://www.tmurgent.com/images/WP_HyperThread.pdf.

Windows Server 2003 Driver Development Kit Documentation, Microsoft Corporation.

Windows 2000 Internals, Dr Robert Capener, Weber State University. icarus.weber.edu/home/bob/cs3100/lectures/win2k_lectures/Win2KMemoryManagement.pdf.

Windows 2000 Terminal Services Performance Guide, UNISYS e-@ction Enterprise Application Delivery Solutions, November 2000.

Windows NT Virtual Memory (Part II), The NT Insider, Vol 6, Issue 1, Jan-Feb 1999. <http://www.osronline.com/article.cfm?id=60>.

Resources

In addition to this paper, there are several great Internet resources that you should be familiar with if you're involved in the tuning and optimization of Terminal Servers.

BrianMadden.com

This is my personal website, and home of my white papers (such as this one), books, product reviews, downloadable training videos, industry news and interviews, and a thin client blog.

dabcc.com

Doug Brown's (Citrix Systems Engineer) site is the home of MetaFrame XP Methodology-in-a-Box and Project Compatibility. Since Doug works for Citrix, this is the first place to go for (unofficial) news about Citrix products and hotfixes.

ServerGeek.net

This site is "a blog dedicated to server-based computing and technology." This site has a broad array of information and news tidbits.

TMUrgent.com

This is the site of Tim Mangan's company. In addition to some interesting tools and utilities, Tim's site is one of the best sites for really technical, in-depth, and detailed analyses of the guts of Windows Terminal Services. His site was the source of all the HyperThreading information in this paper.

Tokeshi.com

This fairly comprehensive support site that is geared towards server-based computing. Tokeshi.com has a ton of information about service packs and hotfixes (both from Citrix and Microsoft) and how they affect Terminal Server environments.

theThin.net

The THIN Net is easily the oldest thin client site on the web, and it's home to the infamous THIN list listserv. Run by Jim Kenzig, this site has a ton of information for Citrix and Terminal Server administrators, including useful tools, utilities, and templates.

TweakCitrix.com

This site is run by Rick Dehlinger, a systems engineer for Citrix. It is most widely known for a document called "MetaFrame XP Tuning Tips & Tricks." This document is filled with various setting and tricks that you can apply to your MetaFrame XP servers.