May 2001

eTESTING LABS

1001 Aviation Parkway, Suite 400 • Morrisville, NC 27560 • 919-380-2800 • Fax 919-380-2899
320 B Lakeside Drive • Foster City, CA 94404 • 650-513-8000• Fax 650-513-8099
www.etestinglabs.com • info@etestinglabs.com • 877-619-9259 (toll free)

Microsoft Services For UNIX Performance Testing

Test report prepared under contract from Microsoft Corporation

Executive summary

Microsoft Corporation commissioned eTesting Labs to conduct a series of performance tests against their Services For Unix 3.0 Release Candidate 1(SFU 3.0) product. The tests specified by Microsoft included the following:

- Comparing the NFS file serving performance of SFU 3.0 to SFU 2.0 and Red Hat Linux 7.2 using the Bonnie NFS performance benchmarking tools
- Comparing the NFS performance of SFU 3.0 to SFU 2.0 and Red Hat Linux 7.2 while performing a series of file operations representing “end user” operations against an NFS mounted volume on a server.

Key findings

- ❑ In the tests we conducted, we saw NFS file serving performance improvements when using SFU 3.0 compared to SFU 2.0. This was particularly true when running the end user operations testing.
- ❑ In the tests we conducted, we found the NFS file serving performance of SFU 3.0 to be competitive with that of Red Hat Linux 7.2

To conduct these tests, we assembled an isolated network testbed consisting of one Dell PowerEdge 350 system configured with a single 850MHz Pentium III processor, 512MB of RAM, a single -10GB disk drive, dual Intel 10/100 Mbps NICs and Red Hat Linux 7.2. We used the Red Hat Linux client system to generate the test load against a single NFS server system configured with either Windows 2000 Advanced Server and Service Pack 2 or Red Hat Linux 7.2. For the NFS server we used a Dell PowerEdge 4400 server configured with two 1.0 GHz Pentium III processors, 2GB of RAM, two Intel 10/100 Mbps NICs, a single Dell PERC 3/Di RAID controller and two Dell PERC 3/DC dual channel RAID controllers.

Each of the Dell PERC 3/DC dual channel RAID controllers was connected to a separate Dell PowerVault 220S RAID chassis. Each Power Vault 220S chassis contained fourteen 73GB SCSI drives configured into a single RAID 0 volume of approximately 955GB of storage for a total of nearly 2 terabytes of total disk storage. For all testing, we configured the RAID controllers to use RAID 0, 64K block size and Write Back caching. Please refer to the Testing Methodology section of this report for complete details of the test configurations.

In the testing we conducted, we found that the NFS performance generated using Microsoft's SFU 3.0 product was better than the NFS performance generated using Microsoft's SFU 2.0. This was particularly true when running the end-user performance testing. Additionally, we found that the NFS performance generated using SFU 3.0 was competitive with the NFS performance we obtained when using Red Hat Linux 7.2 on the NFS server.

NFS Performance Results Using Bonnie

We used the Bonnie performance measurement tools to compare the NFS file serving performance of SFU 3.0, SFU 2.0 and Red Hat Linux 7.2. Bonnie is a generic throughput benchmark product that allows users to measure the performance of file servers, including file servers running NFS. Bonnie is freely available from www.textuality.com/bonnie.

Bonnie performs a series of sequential and random read operations and sequential write operations on a file of known size. For our testing, we specified a 2GB file size. In this case, the test file resides on an NFS exported file system residing on the NFS server system. We ran the Bonnie software from the test client and used NFS version 3 and UDP. For each operation described above, Bonnie generates an output rate in KB/Sec and the CPU utilization recorded on the client during the specific test is reported as a percentage. For all tests, higher numbers are better.

Figure 1 below shows the sequential output results generated using Bonnie for the sequential output tests. For the per-character write tests, we found that the performance generated using SFU 3.0 was approximately 9 percent higher compared to Red Hat Linux 7.2 and performance generated using SFU 2.0 was approximately 11 percent higher compared to Red Hat Linux 7.2. For the efficient block write tests, we found that the performance generated using SFU 3.0 was approximately 3 percent higher compared to Red Hat Linux and approximately 63 percent higher compared to SFU 2.0. For the rewrite tests, we found that the performance generated using SFU 3.0 was approximately 28 percent better compared to Red Hat Linux 7.2 and approximately 53 percent better than SFU 2.0.

	Per Character		Block		Rewrite	
	KB/sec	%CPU	KB/sec	%CPU	KB/sec	%CPU
SFU 3.0	10394	97.3	10485	7.9	5744	4.2
SFU 2.0	10621	98.9	6414	4.7	3754	2.6
Red Hat 7.2	9553	91.5	10218	7.5	4486	4.2

Figure 1. Sequential Output Results

Figure 2 below shows the Sequential Input test results generated using Bonnie on a per character and block basis. For the sequential input tests, Bonnie uses a series of getc() macro invocations and efficient block reads to measure the sequential read performance.

For the per-character read tests, we found that the performance generated using SFU 3.0 and SFU 2.0 was approximately 69 percent higher compared to Red Hat Linux 7.2. For the efficient block read tests, we found that the performance of both SFU 3.0 and Red Hat Linux 7.2 to be approximately the same and both better than SFU 2.0 by approximately 10 percent.

	Per Character		Block	
	KB/sec	%CPU	KB/sec	%CPU
SFU 3.0	9511	97.3	10806	4.2
SFU 2.0	9503	97.3	9932	4.1
Red Hat 7.2	5601	57.7	10919	3.7

Figure 2. Sequential Input Results

Figure 3 below shows the Random testing results obtained using Bonnie. For these random read tests, Bonnie performs a series of "seeks" to random locations in the test file and records the number of seeks per second. For these tests, SFU 3.0 generated the best overall seek rate of 591 seeks per second. This was approximately 11 percent better than Red Hat Linux 7.2. Please refer to the Test Results section for complete details.

	Seeks/Sec	%CPU
SFU 3.0	591	7.2
SFU 2.0	489	5.4
Red Hat 7.2	532.1	8

Figure 3. Random Results

NFS Performance Results With User Interaction Scenarios

In addition to testing NFS performance using the benchmarking software described above, we tested the NFS performance of SFU 3.0, SFU 2.0 and Red Hat Linux 7.2 NFS servers by manually issuing a number commands to manipulate files and directories residing on a shared NFS server volume. These tests included the following:

- Listing files in a directory residing on the NFS server using variations of the Unix "ls" command from a test client running Red Hat Linux 7.2.
- Copying a large, 480MB file from a test client running Red Hat Linux 7.2 to the NFS server and then from the NFS server back to the test client.
- Creating a large, 10GB file on the NFS server from a test client running Red Hat Linux 7.2

To record the time required to accomplish each of the tasks listed above, we used the time function at the same time we issued the specific commands required to accomplish each task. For example, the command below shows how we recorded the elapsed time required to view information related to all files in a directory that begin with "aaa" :

- time ls –a aaa*

Using the time command in combination with another command returns three time measurements related to the specific command as follows:

- Real – The real elapsed time required to complete the command
- User – The total number of CPU seconds spent is user mode
- System – The total number of CPU seconds spent in system mode

After issuing each command necessary to measure the items listed above, we recorded the real, user and system times for each item. For these tests, lower elapsed and system times are better.

We found that the time required to perform the general file listing, copying and creation functions described above was similar when comparing results generated using SFU 3.0 and Red Hat Linux 7.2. We also found that, in general, both SFU 3.0 and Red Hat Linux 7.2 were faster than SFU 2.0 when performing these operations. Please refer to the Test Results section for complete details.

Testing methodology

Microsoft Corporation commissioned eTesting Labs to conduct a series of performance tests against their Services For Unix 3.0 Release Candidate 1(SFU 3.0) product. These tests included the following:

- Comparing the NFS performance of SFU 3.0 to SFU 2.0 and Red Hat Linux 7.2 using the Bonnie NFS performance benchmarking tools

- Comparing the NFS performance of SFU 3.0 to SFU 2.0 and Red Hat Linux 7.2 while performing a series of file operations representing "end user" operations against an NFS mounted volume on a server.

To conduct these tests, we assembled an isolated network testbed consisting of one Dell PowerEdge 350 system configured with a single 850MHz Pentium III processor, 512MB of RAM, a single 10GB disk drive, dual Intel 10/100 Mbps NICs and RedHat Linux 7.2. We used the Red Hat Linux client system to generate the test load against a single NFS server system configured with either Windows 2000 Advanced Server and Service Pack 2 or Red Hat Linux 7.2. For the NFS server we used a Dell PowerEdge 4400 server configured with two 1.0 GHz Pentium III processors, 2GB of RAM, two Intel 10/100 Mbps NICs, a single Dell PERC 3/Di RAID controller and two Dell PERC 3/DC dual channel RAID controllers. We connected the Dell 4400 server system and the test client running Red Hat Linux 7.2 using an Extreme Summit48 switch using 100 Mbps full duplex connections. Figure 4 provides a diagram of the network used for these tests.

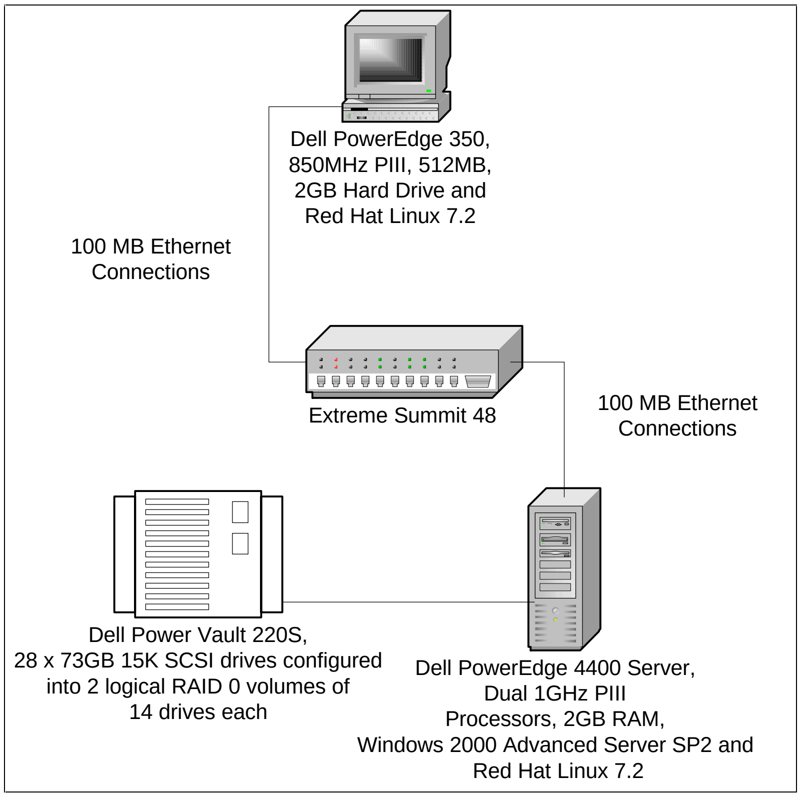

Figure 4. Test Network for NFS Performance Testing

The Dell PERC 3/Di RAID controller managed a total of seven 9GB SCSI drives configured into a single 9GB logical volume on which we installed the operating system under test and a second logical 50 GB RAID 0 volume spanning the remaining six drives.

Each of the Dell PERC 3/DC dual channel RAID controllers was connected to a separate Dell PowerVault 220S RAID chassis. Each Power Vault 220S chassis contained fourteen 73GB SCSI drives configured into a single RAID 0 volume of approximately 955GB of storage for a total of nearly 2 terabytes of total disk storage. For all testing, we configured the RAID controllers to use RAID 0, 64K block size and Write Back caching.

Installing and Configuring Services For Unix

For these tests, we installed Windows 2000 Advanced Server and Service Pack 2 on the Dell PowerEdge 4400 server. We configured each of the two logical 955GB RAID 0 disk volumes connected to the Dell PowerEdge 4400 server into eight logical NTFS volumes of approximately 120GB each using the Disk Management tools available under Windows 2000 Advanced Server. After creating each of the 16 separate volumes, we formatted each volume as an NTFS file system using the quick format feature and a 64K byte allocation unit size. We then increased the size of the NTFS log file on each of the 16 volumes by issuing the following command against each of the 16 NTFS volumes:

```
Chkdsk <drive-letter> /l 65536
```

After configuring the disk subsystem, we installed the SFU 3.0 software on the Dell PowerEdge 4400 server. The SFU 3.0 software provides NFS server software that allows NTFS volumes residing on a Windows based server to be shared to Unix based clients using the NFS protocol. After installing the SFU 3.0 software, we installed the User Name Mapping feature of SFU 3.0. This feature allows users to map UNIX user and group account names to Windows users and groups so that Unix users can access shared Windows volumes without specifying a user name and password. After installing the User Name Mapping software, we configured an advanced user and group mapping that allowed our test client to map shared volumes on the Windows 2000 Advanced Server system using NFS.

After creating the user and group mappings and verifying that the NFS server software was running, we created an NFS share on one of the 16 volumes described above. We then verified that the Red Hat Linux client could successfully mount the shared volumes using NFS by adding an entry in the clients' /etc/fstab file and performing a "mount –a" command to mount the volume.

When testing using SFU 3.0, we made the following registry modifications. In the list below HKLM indicates HKEY_LOCAL_MACHINE and CCS indicates CurrentControlSet:

- HKLM\CCS\Control\Executive\ AdditionalDelayedWorkerThreads = 16
- HKLM\CCS\Services\Nfssvr\Parameters\ OptimalReWrites = 1
- HKLM\CCS\Services\NfsSvr\Parameters\RdWrThreadSleepTime = 60.
- HKLM\CCS\Control\Session Manager\Memory Management\SystemPages = 0
- HKLM\CCS\Control\Session Manager\Memory Management\LargeSystemCache = 0
- HKLM\CCS\Control\Session Manager\Memory Management\PagedPoolSize = FFFFFFFF

To install SFU 2.0 under Windows 2000 Advanced Server and Service Pack 2, we followed the identical steps listed above to install and configure SFU 3.0. When testing using SFU 2.0, we made the following registry modifications:

- HKLM\CCS\Services\Nfssvr\Parameters\UseWriteCache = 0
- HKLM\Software\Microsoft\Server For NFS\Current Version\Mapping\ImplicitPermissions = 1

Installing and Configuring Red Hat Linux 7.2

We also tested with Red Hat Linux 7.2 on the Dell PowerEdge 4400 server for these tests. During installation, we chose the standard “server” install option. Additionally, we selected that the NFS server capabilities be installed.

We configured each of the Dell PowerVault disk units as a single RAID 0 stripe spanning all 14 physical drives in the unit. After installing the operating system, we created a single 120GB file system on one of the Dell PowerVault 220S RAID devices. We then enabled this file system to be mounted remotely from our test client by adding an entry to the /etc/fstab directory giving read and write access to the file system to our test client.

We then verified that our test client could mount the 120GB shared volume on the server using NFS version 3. We made no other modifications to the Red Hat Linux 7.2 server system.

Testing NFS Performance using Bonnie

Bonnie is a generic throughput benchmark product that allows users to measure the performance of file servers, including servers running NFS. Bonnie is freely available from www.textuality.com/bonnie.

Bonnie performs a series of tests on a file of known size. For our testing, we specified a 2GB file size. In this case, the file resides on an NFS exported file system residing on the NFS server system. We ran the Bonnie software from the test client and used NFS version 3 and UDP. The tests performed are described below and the information provided was taken directly from the Bonnie Web site:

Sequential Output

- Per-Character -The file is written using the *putc()* stdio macro. The loop that does the writing should be small enough to fit into any reasonable I-cache. The CPU overhead here is that required to do the stdio code plus the OS file space allocation.

- Block - The file is created using *write(2)*. The CPU overhead should be just the OS file space allocation.

- Rewrite - Each **Chunk** (currently, the size is 16384) of the file is read with *read(2)*, dirtied, and rewritten with *write(2)*, requiring an *lseek(2)*. Since no space allocation is done, and the I/O is well-localized, this should test the effectiveness of the filesystem cache and the speed of data transfer.

Sequential Input

- Per-Character - The file is read using the *getc()* stdio macro. Once again, the inner loop is small. This should exercise only stdio and sequential input.
- Block - The file is read using *read(2)*. This should be a very pure test of sequential input performance.

Random Seeks

This test runs **SeekProcCount** (currently 4) processes in parallel, doing a total of 4000 *lseek()*s to locations in the file computed using by *random()* in bsd systems, *drand48()* on sysV systems. In each case, the block is read with *read(2)*. In 10% of cases, it is dirtied and written back with *write(2)*.

After downloading the Bonnie software, we installed it on the test client running Red Hat Linux 7.2. We followed the installation instructions and simply issued a "make" command in the directory where the Bonnie software had been extracted. We then modified the "/etc/fstab" file on the test client to perform a mount of one of the sixteen 120GB volumes available on the Dell PowerEdge 4400 server (for the Windows test) as shown below:

```
        Nfs-server:/share0        /mnt/mnt1        nfs        nfsvers=3, rw, 0 0
```

Once the shared volume was properly mounted, we invoked the Bonnie software using the following command line:

```
        ./bonnie –d /mnt/mnt1 –s 2047 –html –m client1
```

We tested Bonnie on SFU 3.0, SFU 2.0 and Red Hat Linux servers.

Testing NFS Performance with User Interaction Scenarios

In addition to testing NFS performance using the benchmarking software described above, we tested the NFS performance of SFU 3.0, SFU 2.0 and Red Hat Linux 7.2 NFS servers by issuing a number commands to manipulate files and directories residing on a shared NFS server volume. These tests included the following:

- Listing files in a directory residing on the NFS server using variations of the Unix "ls" command from one a test client running Red Hat Linux 7.2.
- Copying a large, 480MB file from a test client running Red Hat Linux 7.2 to the NFS server and then from the NFS server back to the test client.
- Creating a large, 10GB file on the NFS server from a test client running Red Hat Linux 7.2.

Listing Files In a Directory

This test involved performing a directory listing of 60,000 files residing on the 120GB NTFS or ext3 partition on the Dell PowerEdge 4400 server. To create the directory layout for this test, we created and ran a script on the NFS server that created the files. The result of the script was to create 100 files named aaa1 – aaa100 and to create another 59,900 files named bb1 – bb59900. After we created the files on the server, we performed the following steps to run the tests:

- Reboot client and server
- Mount the shared server volume on the client using NFS version 3
- Run "time ls aaa*" on the mount point. Collect the time statistics
- Unmount the shared volume from the test client
- Reboot the client and the server machines.
- Mount the shared server volume on the client using NFS version 3
- Run "time ls –l aaa*" on the mount point. Collect the time statistics
- Unmount the shared volume from the test client
- Reboot the client and the server machines.
- Mount the shared server volume on the client using NFS version 3
- Run "time ls –l" on the mount point. Collect the time statistics

The time statistics described above are generated using the “time” command on the test client. This command runs a command, in this case the “ls” command, and writes a message to standard output giving timing statistics about the program including the elapsed time between invocation and termination of the specified command. Using this method allowed us to generate time statistics regarding how long it took to perform the operations described above.

Copying a Large File from Client to Server and from Server to Client

These tests include recording the time required to copy a 480MB file from a test client running Red Hat Linux 7.2 to the NFS server and then from the NFS server back to the test client. For this test we used the “mkfile” utility to create the test file. Because Red Hat Linux 7.2 did not include a mkfile utility, we downloaded a freely available version of mkfile from the following URL:

http://www2.cddc.vt.edu/linux/utils/scripts.

We then created the 480MB file using the command:

```
mkfile 480m /tmp/bigfile
```

This created a 480MB file in the /tmp directory on the test client. After creating the test file, we performed the following steps to run the test and record the elapsed time associated with each test:

- Reboot both client and server
- Mount the shared server volume on the client using NFS version 3
- From within the mount point, perform a “time cp /tmp/bigfile **.**” and collect the time statistics
- Unmount the shared volume from the test client and delete /tmp/bigfile from the client
- Reboot the client and the server machines.
- Mount the shared server volume on the client using NFS version 3
- From within the mount point, perform a “time cp bigfile /tmp” and collect the time statistics

Creating Large Files

These tests include recording the time required to create a 10GB file from a test client running Red Hat Linux 7.2 on the NFS server. For this test we used the “mkfile” utility to create the test file. Because Red Hat Linux 7.2 did not include a mkfile utility, we downloaded a freely available version of mkfile from the following URL:

http://www2.cddc.vt.edu/linux/utils/scripts.

We then used the mkfile utility to create the 10GB file using the command below executed on the client from within the shared server directory specified by the mount point:

```
mkfile 10240m really_big_file
```

This created a 10GB file on the NFS server. We performed the following steps to run the test and record the elapsed time associated with each test:

- Reboot both client and server
- Mount the shared server volume on the client using NFS version 3
- From within the mount point, perform a “time mkfile 10240m really_big_file” and collect the time statistics

Test results

Microsoft Corporation commissioned eTesting Labs to conduct a series of performance tests against their Services For Unix 3.0 Release Candidate 1(SFU 3.0) product. These tests included the following:

- Comparing the NFS performance of SFU 3.0 to SFU 2.0 and Red Hat Linux 7.2 using the Bonnie NFS performance benchmarking tools

- Comparing the NFS performance of SFU 3.0 to SFU 2.0 and Red Hat Linux 7.2 while performing a series of file operations representing “end user” operations against an NFS mounted volume on a server.

To conduct these tests, we assembled an isolated network testbed consisting of one Dell PowerEdge 350 system configured with a single 850MHz Pentium III processor, 512MB of RAM, a single 10GB disk drive, dual Intel 10/100 Mbps NICs and RedHat Linux 7.2. We used the Red Hat Linux client to generate the test load against a single NFS server system configured with either Windows 2000 Advanced Server and Service Pack 2 or Red Hat Linux 7.2. For the NFS server we used a Dell PowerEdge 4400 server configured with two 1.0 GHz Pentium III processors, 2GB of RAM, two Intel 10/100 Mbps NICs, a single Dell PERC 3/Di RAID controller and two Dell PERC 3/DC dual channel RAID controllers.

In the testing we conducted, we found that the NFS performance generated using Microsoft’s SFU 3.0 product was better than the NFS performance generated using Microsoft’s SFU 2.0. This was particularly true when running the end-user performance testing. Additionally, we found that the NFS performance generated using SFU 3.0 was competitive with the NFS performance we obtained when using Red Hat Linux 7.2 on the NFS server.

NFS Performance Results Using Bonnie

We used the Bonnie performance measurement tools to compare the NFS file serving performance of SFU 3.0, SFU 2.0 and Red Hat Linux 7.2. Bonnie is a generic throughput benchmark product that allows users to measure the performance of file servers, including servers running NFS. Bonnie is freely available from www.textuality.com/bonnie.

Bonnie performs a series of tests on a file of known size. For our testing, we specified a 2GB file size. In this case, the file resides on an NFS mounted file system residing on the NFS server system.

Figure 5 below shows the sequential output results generated using Bonnie for the sequential output tests. For the sequential output tests, Bonnie performs a series of single character write operations using putc() macro invocations, efficient block writes and re-writes of modified file blocks to measure the sequential write performance. For each operation described above, Bonnie generates an output rate in KB/Sec and the CPU utilization recorded on the client during the specific test is reported as a percentage. For all tests, higher numbers are better.

	Per Character		Block		Rewrite	
	KB/sec	%CPU	KB/sec	%CPU	KB/sec	%CPU
SFU 3.0	10394	97.3	10485	7.9	5744	4.2
SFU 2.0	10621	98.9	6414	4.7	3754	2.6
Red Hat 7.2	9553	91.5	10218	7.5	4486	4.2

Figure 5. Bonnie Sequential Output Results

For the per-character write tests, we found that the performance generated using SFU 3.0 was approximately 9 percent higher compared to Red Hat Linux 7.2 and performance generated using SFU 2.0 was approximately 11 percent higher compared to Red Hat Linux 7.2. For the efficient block write tests, we found that the performance generated using SFU 3.0 was approximately 3 percent higher compared to Red Hat Linux and approximately 63 percent higher compared to SFU 2.0. For the rewrite tests, we found that the performance generated using SFU 3.0 was approximately 28 percent better compared to Red Hat Linux 7.2 and approximately 53 percent better than SFU 2.0.

Figure 6 below shows the Sequential Input test results generated using Bonnie on a per character and block basis. For the sequential input tests, Bonnie uses a series of getc() macro invocations and efficient block reads to measure the sequential read performance. For each operation described above, Bonnie generates an output rate in KB/Sec and the CPU utilization recorded on the client during the specific test is reported as a percentage. For all tests, higher numbers are better.

For the per-character read tests, we found that the performance generated using SFU 3.0 and SFU 2.0 was approximately 69 percent higher compared to Red Hat Linux 7.2. For the efficient block read tests, we found that the performance of both SFU 3.0 and Red Hat Linux 7.2 to be approximately the same and both better than SFU 2.0 by approximately 10 percent.

	Per Character		Block	
	KB/sec	%CPU	KB/sec	%CPU
SFU 3.0	9511	97.3	10806	4.2
SFU 2.0	9503	97.3	9932	4.1
Red Hat 7.2	5601	57.7	10919	3.7

Figure 6. Bonnie Sequential Input Results

Figure 7 below shows the Random testing results obtained using Bonnie. For these random tests, Bonnie performs a series of “seeks” to random locations in the test file and records the number of seeks per second. For these tests, SFU 3.0 generated the best overall seek rate of 591 seeks per second. This was approximately 11 percent better than Red Hat Linux 7.2 and approximately 21 percent better than SFU 2.0.

	Seeks/Sec	%CPU
SFU 3.0	591	7.2
SFU 2.0	489	5.4
Red Hat 7.2	532.1	8

Figure 7. Bonnie Random Results

NFS Performance Results With User Interaction Scenarios

In addition to testing NFS performance using the benchmarking software described above, we tested the NFS performance of SFU 3.0, SFU 2.0 and Red Hat Linux 7.2 NFS servers by issuing a number commands to manipulate files and directories residing on a shared NFS server volume. These tests included the following:

- Listing files in a directory residing on the NFS server using variations of the Unix “ls” command from a test client running Red Hat Linux 7.2.
- Copying a large, 480MB file from a test client running Red Hat Linux 7.2 to the NFS server and then from the NFS server back to the test client.
- Creating a large, 10GB file on the NFS server from a test client running Red Hat Linux 7.2.

To record the time required to accomplish each of the tasks listed above, we used the time function at the same time we issued the specific commands required to accomplish each task. For example, the command below shows how we recorded the elapsed time required to view information related to all files in a directory that begin with "aaa" :

- time ls –l aaa*

Using the time command in combination with another command returns three time measurements related to the specific command as follows:

- Real – The real elapsed time in seconds required to complete the command
- User – The total number of CPU seconds spent is user mode
- System – The total number of CPU seconds spent in system mode

After issuing each command necessary to measure the items listed above, we recorded the real, user and system times for each item. For these tests, lower elapsed and system times are better.

Listing Files In a Directory

This test involved performing a directory listing of 60,000 files residing on a 120GB partition on the Dell PowerEdge 4400 server using variations of the "ls" command. Depending on the operating system running on the server, Windows 2000 Advanced Server or Red Hat Linux 7.2, the 120GB partition was either NTFS or ext3. We created the 60,000 files in a single shared directory on the server. The files consisted of 100 files named aaa1 – aaa100 and another 59,900 files named bb1 – bb59900. After installing the test files on the server, we mounted the shared volume from the test client running Red Hat Linux 7.2 and performed the following commands on the client from within the mounted directory:

- ls aaa* - simple filename listing of all 100 files starting with aaa
- ls –l aaa* - returns extended file information for all 100 files starting with aaa including permissions, file owner and group information, file size and last modified date.
- ls –l – returns extended file information of all 60,000 files in the shared directory including permissions, file owner and group information, file size and last modified date.

Figures 8 through 10 below show the time information recorded for each operation for each configuration tested. For the simple file listing of all 100 files starting with "aaa", we found that SFU required the least amount of elapsed time to perform the operation at 2.069 seconds. Red Hat Linux required approximately 1.5 elapsed seconds longer to perform this task compared to SFU 3.0 and that SFU 2.0 required nearly 2 more elapsed seconds to accomplish this task compared to SFU 3.

ls aaa*	SFU 3.0	SFU 2.0	RedHat Linux 7.2
Elapsed	2.069 sec	4.146 sec	3.623 sec
User	0.03 sec	0.01 sec	0.01 sec
Sys	0.89 sec	0.03 sec	3.54 sec

Figure 8. Time statistics for "ls aaa*" file list command

For the more complex file listing of all 100 files starting with "aaa", we found that SFU 3.0 required the least amount of elapsed time to perform the operation at 2.051 seconds. Red Hat Linux required approximately 2 additional elapsed seconds to perform this task compared to SFU 3.0 and SFU 2.0 required nearly 3.5 more seconds to accomplish this task compared to SFU 3.0

ls -l aaa*	SFU 3.0	SFU 2.0	RedHat Linux 7.2
Elapsed	2.051 sec	5.698 sec	4.022 sec
User	0.01 sec	0.01 sec	0.02 sec
Sys	0.96 sec	0.93 sec	3.2 sec

Figure 9. Time statistics for "ls –l aaa*" file list command

For the most complex file listing of all 60,000 files we found that Red Hat Linux 7.2 required the least amount of elapsed time to perform this operation at 73.608 seconds. SFU 3.0 required approximately 4 more seconds to complete this task compared to Red Hat Linux 7.2 and that SFU 2.0 required approximately 7 additional elapsed seconds to perform this task compared to Red Hat Linux 7.2.

ls -l	SFU 3.0	SFU 2.0	RedHat Linux 7.2
Elapsed	77.479 sec	80.983 sec	73.608 sec
User	2.69 sec	2.44 sec	2.66 sec
Sys	2.42 sec	2.39 sec	6.09 sec

Figure 10. Time statistics for "ls –l" file list command

Copying a Large File from Client to Server and from Server to Client

These tests include recording the time required to copy a 480MB file from a test client running Red Hat Linux 7.2 to the NFS server and then from the NFS server back to the test client. For this test we used the "mkfile" utility to create the 480MB test file in the /tmp directory of the client using the following command:

- mkfile 480m /tmp/bigfile

After creating the 480MB file, we issued the following commands from the test client to copy the file to the NFS server and then to copy the file back from the NFS server to the client. We issued both commands on the test client from within the shared server volume:

- time cp /tmp/bigfile . : copy the file /tmp/bigfile to the current directory on the shared server volume
- time cp bigfile /tmp : copy the file "bigfile" from the shared server volume to the clients local /tmp directory.

Figure 11 shows the times required to copy the file from the client to the NFS server running Red Hat Linux 7.2. We found that SFU 3.0 required the least amount of elapsed time to perform the operation at 44.057 seconds. Red Hat Linux 7.2 required approximately 1 additional elapsed second to complete the operation compared to SFU 3.0. Using SFU 2.0 required approximately 29 additional elapsed seconds to copy the file from the client to the server compared to both SFU 3.0 and Red Hat Linux 7.2.

Copy 480MB file to server	SFU 3.0	SFU 2.0	RedHat Linux 7.2
Elapsed	44.057sec.	73.045 sec.	45.192 sec.
User	0.06 sec.	0.03 sec.	0.09 sec.
Sys	6.26 sec.	5.35 sec.	6.07 sec.

Figure 11. Time statistics for copying 480MB file from client to server

Figure 12 shows the times required to copy the file from the NFS server to the client. In this test, we found that Red Hat Linux 7.2 required the least amount of elapsed time to perform this operation at 43.996 seconds. Using SFU 3.0 on the NFS server required approximately 0.2 additional elapsed seconds to complete the task compared to copying the same file when running Red Hat Linux 7.2 on the NFS server. Using SFU 2.0 required approximately 3 additional elapsed seconds to copy the file from the client to the server compared to both SFU 3.0 and Red Hat Linux 7.2.

Copy 480MB file from server	SFU 3.0	SFU 2.0	RedHat Linux 7.2
Elapsed	44.252 sec.	46.883 sec.	43.996 sec.
User	0.14 sec.	0.14 sec.	0.1 sec.
Sys	6.59 sec.	7.39 sec.	7.19 sec.

Figure 12. Time statistics for copying 480MB file from server to client

Creating Large Files

These tests include recording the time required to create a 10GB file on the NFS server from a test client running Red Hat Linux 7.2. For this test we used the “mkfile” utility to create the test file directly on the shared server volume. For this test, we issued the following command on the test client from within the shared server directory:

- time mkfile 10240m really_big_file

Figure 13 below shows the results of this test. We found that using SFU 3.0 required the least amount of elapsed time to complete this operation at 16 minutes and 26.415 seconds. Using Red Hat Linux 7.2 on the server required almost another 1.5 minutes of elapsed time to create the file on the NFS server compared to running SFU 3.0. Because of a known 4GB file size limitation using SFU 2.0, we were unable to complete this test using SFU 2.0.

Create 10GB file on server	SFU 3.0	SFU 2.0	RedHat Linux 7.2
Elapsed	16min 26.415sec	N/A	17min 47.429sec
User	7.09 sec	N/A	7.09sec
Sys	1min 34.660s	N/A	1min 38.780sec

Figure 13. Time statistics for creating a 10GB file on the server

Appendix A. System Disclosure Information

NFS Server	
Machine Type	Dell PowerEdge 4400
BIOS	AWARD
Processor(s)	Dual Pentium III 1GHz Xeon
Hard Drive	Seagate Barracuda
Memory	2GB
L2 Cache	512KB
Motherboard	Intel
Network Adapter(s)	4 x Intel Pro 100 Server Adapters
Video Card	ATI 3D Rage II PCI
OS	Microsoft Windows 2000 Server, Service Pack 2

Figure 14. NFS Server Configuration Information

Test Client	
Machine Type	Dell PowerEdge 350
BIOS	AWARD
Processor(s)	Pentium III 850MHz
Hard Drive	10GB IDE
Memory	512
L2 Cache	256KB
Motherboard	Intel
Network Adapter(s)	Intel Pro 100 Server Adapters
Video Card	ATI 3D Rage II PCI
OS	Red Hat Linux 7.2

Figure 15. NFS Server Configuration Information

eTesting Labs Inc. (www.etestinglabs.com), a Ziff Davis Media company, leads the industry in Internet and technology testing. In June 2000, ZD Labs changed its name to eTesting Labs to better reflect the breadth of its testing services. Building on Ziff Davis Media's history of leadership in product reviews and benchmark development, eTesting Labs brings independent testing, research, development, and analysis directly to publications, Web sites, vendors, and IT organizations everywhere.

For more information email us at info@etestinglabs.com or call us toll free at 877-619-9259.