CITI Technical Report 96–2

A Study of Network File System Performance and Capability on MVS

David H. Detlefs
dhd@umich.edu

Diana Noble
djk@umich.edu

ABSTRACT

This report contains results gathered in an NFS benchmarking study. These tests were conducted to compare the single and multiple session throughput, CPU utilization, and command response time of NFS for MVS on IBM and Interlink hardware and software.

April 1996

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

A Study of Network File System Performance and Capability on MVS

David H. Detlefs and Diana Noble

April 1996

This report contains results gathered in the NFS benchmarking study conducted by the University of Michigan's Center for Information Integration, under contract with the Storage Systems Division of IBM Corporation.[1] These tests were conducted to compare the single and multiple session throughput, CPU utilization, and command response time of NFS for MVS on IBM and Interlink hardware and software.

Benchmarking almost any modern software system is often more difficult than one would at first imagine. Networked systems increase the challenge, because each component adds to the overall complexity of the system. If the two systems are as diverse as MVS and UNIX, things can get completely out of hand very quickly. Thus, this report may, and probably does, contain some errors. We have done our best to treat both IBM and Interlink fairly and have worked to provide consistent results. However, due to the complexity of the task, the breadth of the workscope, and the limited time allocated for the project, perfection was probably not an attainable goal. Please keep this in mind when reviewing these results.

1. Configuration

All tests were conducted using an ES/9000 Model 580 IBM Mainframe that was one of three dedicated CPUs. It had 64MB dedicated real memory, 512MB dedicated extended storage, and 3390 disk storage (3990-2 with 256MB cache). Tests were conducted on MVS OpenEdition 5.1 (running under VM/ESA 2.1).

The IBM tests were conducted using an IBM 3172 Model 3 Controller (ICP mode) that had a Pentium 60 motherboard, IBM FDDI Base Adapter, and IBM Etherstreamer Adapter. The IBM tests also used an IBM Parallel Channel Adapter. The software used was TCP/IP 3.1 for MVS, DFSMS 1.2 NFS for MVS, and ICP 3.3 for 3172-3 (most recent level).

The Interlink tests were conducted using an Interlink 3762 Extended LAN Controller Model 3 that had an FDDI adapter. The software used was SNS/TCPaccess 2.0. and SNS/NFS 2.0.1.

Table 1 lists the programs used for testing and report generation during the NFS benchmark project. All of the programs were run under AIX. Some programs were provided by IBM, while

1. To the best of our knowledge, no confidential and/or proprietary information of the University of Michigan and/or any third party has been disclosed within this report, or during the course of NFS testing.

others were created at CITI. Many of the CITI programs are PERL scripts. PERL is a public domain, interpreted language that is available on most computing platforms.

TABLE 1. *Test Tools*

Program	Lang	Auth	Description
rpct5d	N/A	IBM	Used to measure throughput for NFS binary mode files.
nfstext	C	CITI	Used to measure throughput for NFS text mode files.
nfstest	N/A	IBM	Used to measure throughput for low-level NFS commands.
mvslogin	N/A	IBM	Used for NFS login to MVS NFS servers (IBM and Interlink).
mvslogout	N/A	IBM	Used for NFS logout from MVS NFS servers (IBM and Interlink).
nfstest.make.keys	PERL	CITI	Used to create a list of test keys.
nfs.setup.cmds	shell	CITI	Used to set the TCP/IP environment on AIX. The values we used were provided by IBM.
nfstest.prealloc	PERL	CITI	Used to test NFS throughput in the case where NFS test files have been pre-allocated on MVS. This program was used with MVS file allocation JCL that created corresponding pre-allocated files. All the performance results provided in this benchmark used pre-allocated MVS files.
nfs.ave.bytes	PERL	CITI	Used to collate results for NFS Single Session Throughput.
nfs.ave.bytes.ethernet	PERL	CITI	Used to collate results for NFS Multiple Session Throughput.
nfs.ave.bytes.and.cpu	PERL	CITI	Used to collate results for NFS MVS Server CPU Utilization. The mvsnfs.filter.cpu and mvsnfs.cpu scripts were run before this script to parse the MVS SMF/RMF report data.
nfs.filter.cpu	PERL	CITI	Used as a preliminary filter for SM/RMF reports downloaded from the mainframe.
nfs.mvsnfs.cpu	PERL	CITI	Used to prepare the SMF/RMF data for use in the nfs.ave.bytes.and.cpu script.
nfstest.list.file	PERL	CITI	Used to list a file in hex. This script was a debugging aid.
nfstest.	PERL	CITI	Used to print the number and length of all records in a record delimited file. This script was a debugging aid.

2. Test Procedures

This section details the step-by-step procedures that we followed for conducting each test.

2.1 NFS Single Session Throughput (FDDI)

1. Pre-allocated test files on MVS.
2. Started MVS TCP/IP and NFS servers.
3. Telneted from an RS/6000 workstation located at CITI to the FDDI attached NFS client RS/6000 workstation. All further commands were run from the client via the telnet session.
4. Ran nfs.setup.cmds to setup TCP/IP environment (only between reboots).
5. Ran nfstest.make.keys to generate a list of test keys or edit a previously generated list. For single session testing, the test key file was nfs/testkeys/test. As an example, consider the test key IBM#4#ps#sms#bin#fb#fastf#rd#20m. The key is a series of values separated by #'s. The first two values are vendor and test iteration. The last value is the file size. All other values describe the type of file being used in the test. Note that the network type (FDDI/Ethernet) is not part of the key, but must be set as a parm in the nfstest.prealloc script.
6. Edited nfstest.prealloc to adjust parameters. The parameters needed to change were documented in the script, and included items such as network type (FDDI/Ethernet) and the sleep time between tests.
7. Ran mvslogin to login to the MVS NFS server.

8. Ran nfstest.prealloc to perform the set of tests.
9. Ran mvslogout to log out from the MVS NFS server.
10. Ran nfs.ave.bytes to collate result logs into a report.

2.2 NFS Multiple Session Throughput (Ethernet)

1. Pre-allocated test files on MVS. Multiple session test files were suffixed with the name of the NFS client.
2. Started MVS TCP/IP and NFS servers.
3. Telneted from an AIX workstation located at CITI to each of the three Ethernet NFS clients, also located at CITI.
4. Ran nfs.setup.cmds to setup TCP/IP environment (only between reboots).
5. Ran nfstest.make.keys to generate a list of test cases or edit a previously generated list. Duplicated the list, creating a test.<hostname> file for each client. Edited each of the created files to adjust the branch of the results tree to which the output would be sent. For example, the file test.grumpy may have had a line IBM#4#ps#sms#bin#fb#fastf#rd#20m. The second value of the key, 4, designated the test iteration. The results of this example test would then be directed to the nfs/ethernet/IBM/4 directory. (Note: CITI uses the AFS files system; thus, all AIX workstations share a common file system. This assumption may not be true in other environments.)
6. Edited nfstest.prealloc to adjust parameters. This script was configured for multiple client testing.
7. Ran mvslogin to login to the MVS NFS server.
8. Ran nfstest.prealloc to perform the set of tests.
9. Ran mvslogout to log out from the MVS NFS server.
10. Ran nfs.ave.bytes.ethernet to collate result logs into a report.

2.3 NFS CPU Utilization

1. Pre-allocated test files on MVS.
2. Started MVS TCP/IP and NFS servers.
3. Started SMF and RMF on MVS and modified them to use PDS members designated for NFS testing.
4. Telneted from an RS/6000 workstation located at CITI to the FDDI attached NFS client RS/6000 workstation. All further commands were run from the client via the telnet session.
5. Ran nfs.setup.cmds to setup TCP/IP environment (only between reboots).
6. Ran nfstest.make.keys to generate a list of test keys or edit a previously generated list. Made sure the iteration value of each key was set so that it did not obliterate the results of previous tests.
7. Edited nfstest.prealloc to adjust parameters.
8. Ran mvslogin to login to the MVS NFS server.
9. Ran nfstest.prealloc to perform the set of tests.
10. Ran mvslogout to log out from the MVS NFS server.
11. On MVS, ran the RMF reporting program using the commands: DELTA, SUMMARY(INT,TOT), REPORTS(ARD).
12. Transfered the report to an MVS file then ftp'd it to a workstation for report generation.
13. Ran mvsnfs.filter.cpu, and then mvsnfs.cpu to extract pertinent SMF/RMF data.

14. Ran nfs.ave.bytes.and.cpu to produce an NFS CPU Utilization Report.

Figure 1 shows the configuration of the network on which testing was conducted. Single session NFS performance testing was isolated in the Test FDDI Ring, shown in the lower right corner of the diagram. Multiple session NFS performance testing was conducted using up to three RS/6000's clients at UM CITI (upper right), and thus traversed the UM Campus FDDI Backbone and two Ethernet's before reaching the MVS mainframe.

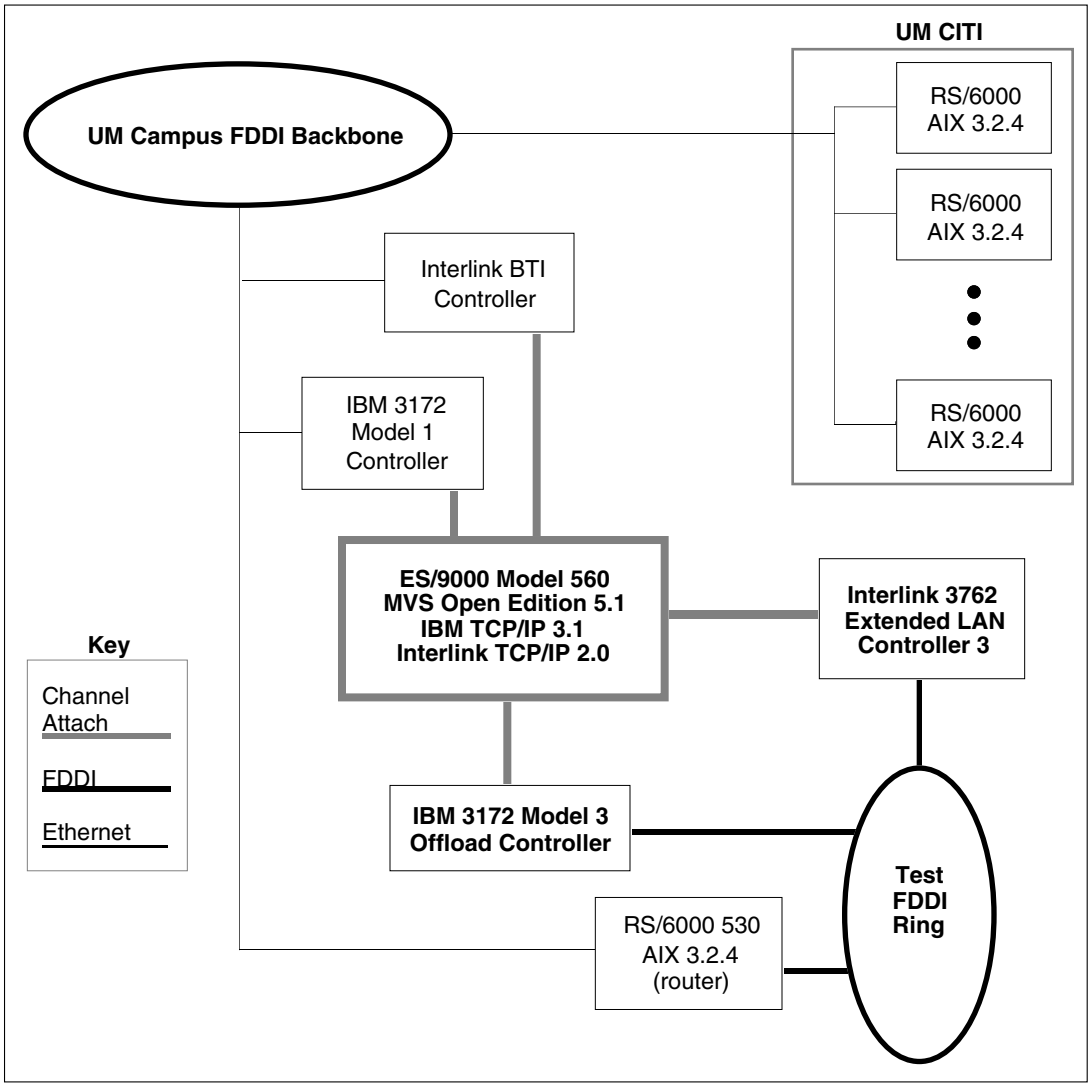

Figure 1: Network Topology

3. NFS and TCP/IP Parameters

The following parameter files define the NFS and TCP/IP configuration used for all performance testing. An effort was made to define like values for Interlink and IBM parameters. In some cases,

the format of the IBM and Interlink parameters varied, so exact duplication of values was not possible.

3.1 IBM Parameters

3.1.1 NFS NFSATTR (IBM)

```
# The following are known as data set creation attributes
space(100,10), trks
rlse
recfm(fb), blksize(24576), lrecl(8192)
dsorg(ps)
dsntype(pds)
dir(27)
volume(PH2027)
keys(64,0)
recordsize(512,4K)
nonspanned
shareoptions(1,3)
attrtimeout(5), readtimeout(5), writetimeout(5)
binary
LF
blankstrip
mapleaddot
maplower
retrieve
fastfilesize
setownerroot
executebitoff
# The following are known as site attributes.
security(NONE)
nopcnfsd
leadswitch
mintimeout(1)
maxtimeout(1800)
logout(1800)
nfstasks(8,16)
cachewindow(128)
logicalcache(4M)
bufhigh(32M)
percentsteal(20)
smf(NONE)
```

3.1.2 TCP/IP CONFIG (IBM)

```
ADDRESSTRANSLATIONPOOLSIZE 1500
ACBPOOLSIZE 3000
CCBPOOLSIZE 150
DATABUFFERPOOLSIZE 160 32768
SMALLDATABUFFERPOOLSIZE 1400
TINYDATABUFFERPOOLSIZE 1400
IPROUTEPOOLSIZE 300
ENVELOPEPOOLSIZE 750
LARGEENVELOPEPOOLSIZE 50 32768
RCBPOOLSIZE 50
SKCBPOOLSIZE 256
SCBPOOLSIZE 768
TCBPOOLSIZE 256
UCBPOOLSIZE 256
DATASETPREFIX TCPIP
NOCHECKSUM
TIMESTAMP PREFIX
NOTRACE ALL
DEVICE 3172-1 LCS 860
LINK CORWIN ETHERNET 0 3172-1
DEVICE BTIBOX LCS 870
LINK RANDOM ETHERNET 0 BTIBOX
DEVICE 3172-3 LCS 8E0
LINK JUSTIN ETHERNET 0 3172-3
LINK JUSTINF FDDI 0 3172-3
HOME
141.211.238.2 CORWIN
```

```
141.211.238.3 RANDOM
141.211.238.4 JUSTIN
141.211.132.4 JUSTINF
GATEWAY
141.211 = CORWIN 1500 0.0.255.0 0.0.238.0
141.211 = RANDOM 1500 0.0.255.0 0.0.238.0
141.211 = JUSTINF 4352 0.0.255.0 0.0.132.0
141.211 = JUSTIN 1500 0.0.255.0 0.0.238.0
DEFAULTNET 141.211.238.1 CORWIN 1500 0
ASSORTEDPARMS
NOFWD
NOUDPQUEUELIMIT
ENDASSORTEDPARMS
INTERNALCLIENTPARMS TIMEMARK 1800
ENDINTERNALCLIENTPARMS
BEGINVTAM
3278-3-E NSX32703
3279-3-E NSX32703
3278-4-E NSX32704
3279-4-E NSX32704
3278-5-E NSX32705
DEFAULTLUS
IFSPTP00..IFSPTP99
ECHO0000..ECHO0999
ENDDEFAULTLUS
LINEMODEAPPL IFSPTSO
ALLOWAPPL IFSPTS* DISCONNECTABLE
ALLOWAPPL *
ENDVTAM
PORT
21 TCP FTPSERV
20 TCP FTPSERV NOAUTOLOG
23 TCP INTCLIEN
25 TCP SMTPSRV
53 UDP NAMESRV
111 TCP PORTMAP
111 UDP PORTMAP
21 TCP LPDSERV
20 TCP LPDSERV NOAUTOLOG
750 TCP MVSKERB
750 UDP MVSKERB
751 TCP ADM@SERV
751 UDP ADM@SERV
AUTOLOG
FTPSERV
NAMESRV
SMTPSRV
PORTMAP
ENDAUTOLOG
START 3172-1
START BTIBOX
START 3172-3
```

3.2 Interlink Parameters

3.2.1 NFS NFSATTR (Interlink)

```
mintimeout(1)
maxtimeout ( 1800 )
attrtimeout(5), readtimeout(5), writetimeout(5)
space(100,10), trks
dir(27)
recfm(fb), blksize(24576), lrecl(8192)
dsorg(ps)
binary
blankstrip, lf
maplower
mapleaddot
retrieve
fastfilesize
executebitoff
setownerroot
model()
```

```
nonspanned
shareoptions(1,3)
keys(64,0)
volume(PH2027)
unit(sysda)
recordsize(512,4096)
logout(1800)
nfstasks(8)
cachewindow(128)
logicalcache(4194304)
bufhigh(33554432)
percentsteal(20)
readaheadmax(32768)
maxrdforszleft(32)
# parms below this line have no equivilant in IBM
hsm(NONE)
rootlogin
```

3.2.2 NFS RPCPARMS (Interlink)

```
memscale(4)
cmdsrc(WTOR,MODIFY)
task(NFSMAIN, NFSMAIN, NFSMAIN)
NFS APICNFG (Interlink)
SUBSYSID(ACSS)
APIOPTN(0)
APISENDREQ (12)
APISENDBYTE (108000)
APIRECVREQ (12)
APIRECVBYTE (108000)
```

3.2.3 TCP/IP ACPCFG00 (Interlink)

```
HOST MVS(GUEST)
DNR(* LOCAL)
JES(2 JES2 UMICHIF2 $)
LOGSTAMP
LUPARM(ACPLUP00)
TIMEZ(EASTERN)
XSEC(GLBLACT)
ACBNAME(A03ACCA)
NETWORK LNID(FCA3762)
NAME('FDDI-subnet')
HOST(141.211.132.5)
SUBNET(255.255.255.0)
MTU(4352)
MSSOPT(ALWAYS)
MSSDEF(576)
NOFWD
LOOP
LNILCS CUTYPE(3762)
NAME(DEV3762)
DEVADDR(8F0)
LNILINK FDDI ADAPTER(0)
NAME(FCA3762)
DEVNAME(DEV3762)
ALLRT
ROUTE DEST(0.0.0.0)
ROUTE(141.211.132.3)
NET('FDDI-subnet')
SERVICE NAME(TELNET)
MODULE(STELNET)
PORT(23)
QLISTEN(100)
AUTOLOGON(TSO)
IDLE(30)
TCP
SERVICE NAME(VTAMAPPL)
MODULE(VTAMAPPL)
PORT(23)
NOCPASSWORD
TCP
SERVICE NAME(FTP)
MODULE(FTPS)
```

```
PORT(21)
TCP
SERVICE NAME(LPR)
MODULE(USPOOL)
PORT(515)
TCP
SERVICE NAME(TCPECHO)
MODULE(STECHO)
PORT(7)
TCP
SERVICE NAME(TCPDISC)
MODULE(STECHO)
PORT(9)
TCP
SERVICE NAME(TCPCGEN)
MODULE(SCHARGEN)
PORT(19)
TCP
SERVICE NAME(UDPECHO)
MODULE(UDPSERV)
PORT(7)
UDP
SERVICE NAME(UDPDISC)
MODULE(UDPSERV)
PORT(9)
UDP
SERVICE NAME(UDPTIME)
MODULE(UDPSERV)
PORT(13)
UDP
TELNET PORT(23)
PASS(DATA)
DEFAULT
APPL NAME(ACCES) APPLID(A03ACCA)
PASS(DATA)
APPL NAME(NETSTAT)
CHARS(3)
APPL NAME(SYSSTAT)
CHARS(3)
APPL NAME(ACTEST)
APPL NAME(LOGON)
APPLID(TSO)
PASS(DATA)
APPL NAME(TSO)
APPLID(TSO)
PASS(DATA)
PORT(1023)
FTP GUEST(GUEST XXXXXXXX)
ACCOUNT(OVERHEAD 0000)
PORT(21)
IBUF(64 2048 )
OBUF(64 2048 )
DEFGAT(FILE)
AUTORECALL
PAD(B B)
UNIT(SYSALLDA)
CHARSET(ENGLISH)
GAT TYPE(SOURCE) RECFM(FB) LRECL(80) BLK(6160)
GAT TYPE(PRINT) RECFM(VBA) LRECL(137) BLK(9076)
GAT TYPE(CARDS) RECFM(FB) LRECL(80) BLK(3120)
GAT TYPE(OBJECT) REC(FB) LRECL(80) BLK(2960)
GAT TYPE(FORTRAN) RECFM(FB) LRECL(80) BLK(6160)
GAT TYPE(LOADLIB) RECFM(U) BLK(6233)
GAT TYPE(ASM) RECFM(FB) LRECL(80) BLK(6160)
GAT TYPE(CNTL) RECFM(FB) LRECL(80) BLK(6160)
GAT TYPE(PL1) RECFM(FB) LRECL(80) BLK(6160)
GAT TYPE(FILE) RECFM(FB) LRECL(80) BLK(3120)
SPACE(TRK(5 15))
GAT TYPE(LIBRARY) RECFM(FB) LRECL(80) BLK(6160)
SPACE(BLK(5 15 46))
SMTP CYCLE(3 10 1)
DIVERT(X 9999999)
LRECL(132)
NOMX
```

```
PATH(SNSTCP.V2.EMAIL)
ROUTE(X LOCAL)
RETURN(Y LOCAL)
UNIT(SYSALLDA)
LPR PORT(515)
LPR PORT(516)
SMF TYPE(130)
VTAM TERMINALS(256)
TIB PROTOCOL(TCP)
TADDRUSE(1:4095)
TADDRASGN(4096:8191)
TIB PROTOCOL(UDP)
TADDRUSE(1:4095)
TADDRASGN(4096:8191)
MAXLSEND(108000)
MAXLRECV(108000)
MAXLTSND(9000)
MAXLTRCV(9000)
MAXQSEND(12)
MAXQRECV(12)
```

4. Test Results

The following reports summarize the results of the NFS performance testing. The NFS files used in testing were pre-allocated on MVS. Advice was sought from both IBM and Interlink on configuration of NFS and TCP/IP, and common parameter values were used whenever possible. The testing was accomplished with customized script routines, described in Section 2, NFS Test Procedures.

The IBM NFS server for MVS used IBM (Pascal) TCP/IP and a 3172-3 communications controller running in ICP mode. The Interlink NFS Server for MVS used Interlink TCP/Access and a 3762 ELC3 communications controller. It's important to keep in mind that the NFS performance figures shown reflect not only the difference between IBM and Interlink NFS offerings, but also reflect differences between IBM and Interlink TCP/IP performance. At the time of our tests, neither vendor offered an MVS NFS server that would run on a competitor's TCP/IP stack.

Tables 2 and 3 explain each column heading in the reports (except for the NFS CPU Utilization report).

TABLE 2. *Report Headings*

Column Heading	Description of Column
File Size	Size of the file used in NFS test (KB=kilobytes, MB=megabytes)
Test Routine	The IBM-supplied test routine rpct5d, used for binary mode tests. The CITI-created test routine, nfstext, used for text mode tests.
bin txt	NFS binary or text mode
ps vs	physical sequential or VSAM file
fb vb sm bi	fixed block or variable blocked; small or big Note: The terms small and big describe the recordsize of VSAM files. See Table 3 for specific interpretation of small and big.
sms non	System Managed Storage file or non-SMS file
fast nofa	NFS fastfile processing or nofastfile processing
rd wr	NFS read or write file operation
IBM/Int	IBM or Interlink (identifies columns to right)

TABLE 2. *Report Headings*

Column Heading	Description of Column
Average Bytes Per Second	Average NFS throughput, measured in bytes per second.
Percent Difference	The percent difference in NFS throughput, between IBM and Interlink. Calculated by dividing Interlink throughput by IBM throughput (x 100).

Table 3 defines record length and block size for physical sequential files. For VSAM (ESDS) files, record length and maximum record length are given. The notation SM and BI refer to small and big respectively, and represent two different test parameters.

TABLE 3. *Record and Block Sizes*

File Type	Record Length	Block Size/Max Length
PS, FB, TXT	80	24560
PS, FB, BIN	8192	24576
PS, VB, TXT	84	27998
PS, VB, BIN	8196	27998
VS, SM, TXT	1024	1536
VS, SM, BIN	8192	12288
VS, BI, TXT	8092	12288
VS, BI, BIN	32760	32760

4.1 NFS Server Throughput

The term session is used throughout this paper to denote the logical connection between an NFS client and an NFS server. A session is established after the client has performed an mvslogin to the server, and mounted a server directory on the client file system. A single session implies that only one NFS client is logged onto the NFS server. The term multiple session refers to multiple NFS clients simultaneously logged onto a single NFS server. To insure maximum performance, all NFS clients participating in multiple client tests ran on separate workstations.

The single session NFS thoughput measurements were conducted over a dedicated, counter-rotating FDDI ring, using an IBM RS/6000 NFS client. IBM supplied a test program, rpct5d, which was used to measure NFS binary file throughput. CITI supplied the nfstext program, which was used to measure NFS text file throughput. Each row in Table 4 represents a variation of file type, file size, vendor, and iteration, for a total of approximately 3,088 separate tests. A number of tests failed. If either the IBM or Interlink test failed, the Average Bytes Per Second column is labeled BROKE.

TABLE 4. *NFS Single Session Throughput (FDDI)*

File Size	Test Routine	bin txt	ps vs	fb vb sm bi	sms non	fast nova	rd wr	IBM	#	Average Bytes Per Second	Int	#	Average Bytes Per Second	Percent Difference: Interlink/ IBM
48KB	rpct5d	bin	ps	fb	non	fast	wr	IBM	6	116091	Int	6	120653	103%
256KB	rpct5d	bin	ps	fb	non	fast	wr	IBM	6	530468	Int	6	653210	123%
20MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	6	850434	Int	6	645472	75%
100MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	6	785374	Int	6	241785	30%
250MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	1	787219	Int	1	117553	14%
48KB	rpct5d	bin	ps	fb	non	fast	rd	IBM	6	159068	Int	6	239739	150%
256KB	rpct5d	bin	ps	fb	non	fast	rd	IBM	6	284571	Int	6	627113	220%
20MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	6	335550	Int	6	488405	145%
100MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	6	320073	Int	6	209024	65%
250MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	1	245913	Int	1	113482	46%
48KB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	6	150029	Int	6	110840	73%
256KB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	6	478921	Int	6	550432	114%
20MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	6	732787	Int	5	639898	87%
100MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	5	702659	Int	6	241820	34%
250MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	1	775573	Int	1	117553	15%
48KB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	6	193739	Int	6	281836	145%
256KB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	6	459008	Int	6	677364	147%
20MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	6	533225	Int	6	524395	98%
100MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	6	453639	Int	6	224556	49%
250MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	1	293225	Int	1	116044	39%
48KB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	6	142717	Int	6	117957	82%
256KB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	6	446138	Int	6	643653	144%
20MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	6	843590	Int	6	647727	76%
100MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	6	775244	Int	6	241354	31%
250MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	1	782519	Int	1	114673	14%
48KB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	6	191681	Int	6	231147	120%
256KB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	6	485992	Int	6	623360	128%
20MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	6	696007	Int	6	481854	69%
100MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	6	596847	Int	6	204950	34%
250MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	1	530655	Int	1	110098	20%
48KB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	6	101190	Int	6	119807	118%
256KB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	6	456949	Int	6	622002	136%
20MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	6	744518	Int	6	646677	86%
100MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	6	748961	Int	6	241447	32%
250MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	1	784862	Int	1	114124	14%
48KB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	6	192067	Int	6	268021	139%
256KB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	6	465862	Int	6	673295	144%
20MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	6	644436	Int	6	510883	79%
100MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	6	693563	Int	6	223811	32%
250MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	1	658653	Int	1	103901	15%
48KB	rpct5d	bin	ps	vb	non	fast	wr	IBM	6	116863	Int	6	126723	108%
256KB	rpct5d	bin	ps	vb	non	fast	wr	IBM	6	525983	Int	6	652425	124%
20MB	rpct5d	bin	ps	vb	non	fast	wr	IBM	6	821013	Int	6	624945	76%
100MB	rpct5d	bin	ps	vb	non	fast	wr	IBM	6	777055	Int	6	240114	30%
250MB	rpct5d	bin	ps	vb	non	fast	wr	IBM	1	759837	Int	1	116508	15%

48KB	rpct5d	bin	ps	vb	non	fast	rd	IBM	6	161909	Int	6	237246	146%
256KB	rpct5d	bin	ps	vb	non	fast	rd	IBM	6	284180	Int	6	631234	222%
20MB	rpct5d	bin	ps	vb	non	fast	rd	IBM	6	330374	Int	6	487416	147%
100MB	rpct5d	bin	ps	vb	non	fast	rd	IBM	6	318908	Int	6	211822	66%
250MB	rpct5d	bin	ps	vb	non	fast	rd	IBM	1	250615	Int	1	112604	44%
48KB	rpct5d	bin	ps	vb	non	nofa	wr	IBM	6	121158	Int	6	106943	88%
256KB	rpct5d	bin	ps	vb	non	nofa	wr	IBM	6	503808	Int	6	509448	101%
20MB	rpct5d	bin	ps	vb	non	nofa	wr	IBM	6	753618	Int	6	533447	70%
100MB	rpct5d	bin	ps	vb	non	nofa	wr	IBM	6	715411	Int	6	225063	31%
250MB	rpct5d	bin	ps	vb	non	nofa	wr	IBM	1	771011	Int	1	116767	15%
48KB	rpct5d	bin	ps	vb	non	nofa	rd	IBM	6	210375	Int	6	273810	130%
256KB	rpct5d	bin	ps	vb	non	nofa	rd	IBM	6	482870	Int	6	619521	128%
20MB	rpct5d	bin	ps	vb	non	nofa	rd	IBM	6	478869	Int	6	426577	89%
100MB	rpct5d	bin	ps	vb	non	nofa	rd	IBM	6	543736	Int	6	205526	37%
250MB	rpct5d	bin	ps	vb	non	nofa	rd	IBM	1	367663	Int	1	110562	30%
48KB	rpct5d	bin	ps	vb	sms	fast	wr	IBM	6	152820	Int	6	127369	83%
256KB	rpct5d	bin	ps	vb	sms	fast	wr	IBM	6	450641	Int	6	656177	145%
20MB	rpct5d	bin	ps	vb	sms	fast	wr	IBM	6	827294	Int	6	621546	75%
100MB	rpct5d	bin	ps	vb	sms	fast	wr	IBM	6	777699	Int	6	239697	30%
250MB	rpct5d	bin	ps	vb	sms	fast	wr	IBM	1	764268	Int	1	114025	14%
48KB	rpct5d	bin	ps	vb	sms	fast	rd	IBM	6	201254	Int	6	234363	116%
256KB	rpct5d	bin	ps	vb	sms	fast	rd	IBM	6	504466	Int	6	627225	124%
20MB	rpct5d	bin	ps	vb	sms	fast	rd	IBM	6	648355	Int	6	484507	74%
100MB	rpct5d	bin	ps	vb	sms	fast	rd	IBM	6	708344	Int	6	205366	28%
250MB	rpct5d	bin	ps	vb	sms	fast	rd	IBM	1	539390	Int	1	111361	20%
48KB	rpct5d	bin	ps	vb	sms	nofa	wr	IBM	6	154073	Int	6	114080	74%
256KB	rpct5d	bin	ps	vb	sms	nofa	wr	IBM	6	446367	Int	6	455988	102%
20MB	rpct5d	bin	ps	vb	sms	nofa	wr	IBM	6	788492	Int	6	559315	70%
100MB	rpct5d	bin	ps	vb	sms	nofa	wr	IBM	6	741155	Int	6	229243	30%
250MB	rpct5d	bin	ps	vb	sms	nofa	wr	IBM	1	647269	Int	1	114025	17%
48KB	rpct5d	bin	ps	vb	sms	nofa	rd	IBM	6	201401	Int	6	269994	134%
256KB	rpct5d	bin	ps	vb	sms	nofa	rd	IBM	6	492042	Int	6	598429	121%
20MB	rpct5d	bin	ps	vb	sms	nofa	rd	IBM	6	697894	Int	6	428888	61%
100MB	rpct5d	bin	ps	vb	sms	nofa	rd	IBM	6	673026	Int	6	205751	30%
250MB	rpct5d	bin	ps	vb	sms	nofa	rd	IBM	1	639375	Int	1	108818	17%
48KB	rpct5d	bin	vs	bi	non	fast	wr	IBM	5	66603	Int	1	BROKE	
256KB	rpct5d	bin	vs	bi	non	fast	wr	IBM	5	229711	Int	1	BROKE	
20MB	rpct5d	bin	vs	bi	non	fast	wr	IBM	5	385710	Int	1	BROKE	
100MB	rpct5d	bin	vs	bi	non	fast	wr	IBM	5	383137	Int	1	BROKE	
48KB	rpct5d	bin	vs	bi	non	fast	rd	IBM	2	197716	Int	1	BROKE	
256KB	rpct5d	bin	vs	bi	non	fast	rd	IBM	2	553292	Int	1	BROKE	
20MB	rpct5d	bin	vs	bi	non	fast	rd	IBM	2	468228	Int	1	BROKE	
48KB	rpct5d	bin	vs	bi	non	nofa	wr	IBM	5	56101	Int	1	BROKE	
256KB	rpct5d	bin	vs	bi	non	nofa	wr	IBM	5	200483	Int	1	BROKE	
20MB	rpct5d	bin	vs	bi	non	nofa	wr	IBM	5	339068	Int	1	BROKE	
100MB	rpct5d	bin	vs	bi	non	nofa	wr	IBM	5	351601	Int	1	BROKE	
48KB	rpct5d	bin	vs	bi	non	nofa	rd	IBM	5	169972	Int	1	BROKE	
256KB	rpct5d	bin	vs	bi	non	nofa	rd	IBM	5	424504	Int	1	BROKE	
20MB	rpct5d	bin	vs	bi	non	nofa	rd	IBM	5	385693	Int	1	BROKE	
100MB	rpct5d	bin	vs	bi	non	nofa	rd	IBM	2	277322	Int	1	BROKE	

48KB	rpct5d	bin	vs	bi	sms	fast	wr	IBM	6	57988	Int	1	BROKE
256KB	rpct5d	bin	vs	bi	sms	fast	wr	IBM	6	210320	Int	1	BROKE
20MB	rpct5d	bin	vs	bi	sms	fast	wr	IBM	6	387801	Int	1	BROKE
100MB	rpct5d	bin	vs	bi	sms	fast	wr	IBM	6	384779	Int	1	BROKE
48KB	rpct5d	bin	vs	bi	sms	fast	rd	IBM	6	177886	Int	1	BROKE
256KB	rpct5d	bin	vs	bi	sms	fast	rd	IBM	6	407196	Int	1	BROKE
20MB	rpct5d	bin	vs	bi	sms	fast	rd	IBM	6	540283	Int	1	BROKE
100MB	rpct5d	bin	vs	bi	sms	fast	rd	IBM	2	331403	Int	1	BROKE
48KB	rpct5d	bin	vs	bi	sms	nofa	wr	IBM	6	50022	Int	1	BROKE
256KB	rpct5d	bin	vs	bi	sms	nofa	wr	IBM	6	183334	Int	1	BROKE
20MB	rpct5d	bin	vs	bi	sms	nofa	wr	IBM	6	378119	Int	1	BROKE
100MB	rpct5d	bin	vs	bi	sms	nofa	wr	IBM	6	384175	Int	1	BROKE
48KB	rpct5d	bin	vs	bi	sms	nofa	rd	IBM	6	172906	Int	1	BROKE
256KB	rpct5d	bin	vs	bi	sms	nofa	rd	IBM	6	379473	Int	1	BROKE
20MB	rpct5d	bin	vs	bi	sms	nofa	rd	IBM	6	558133	Int	1	BROKE
100MB	rpct5d	bin	vs	bi	sms	nofa	rd	IBM	2	485196	Int	1	BROKE
48KB	nfstext	bin	vs	sm	non	fast	wr	IBM	6	207171	Int	1	BROKE
256KB	nfstext	bin	vs	sm	non	fast	wr	IBM	6	236628	Int	1	BROKE
20MB	nfstext	bin	vs	sm	non	fast	wr	IBM	6	221920	Int	1	BROKE
100MB	nfstext	bin	vs	sm	non	fast	wr	IBM	6	220481	Int	1	BROKE
48KB	nfstext	bin	vs	sm	non	fast	rd	IBM	6	BROKE	Int	1	BROKE
256KB	nfstext	bin	vs	sm	non	fast	rd	IBM	6	BROKE	Int	1	BROKE
20MB	nfstext	bin	vs	sm	non	fast	rd	IBM	6	BROKE	Int	1	BROKE
100MB	nfstext	bin	vs	sm	non	fast	rd	IBM	6	BROKE	Int	1	BROKE
48KB	nfstext	bin	vs	sm	non	nofa	wr	IBM	6	209508	Int	1	BROKE
256KB	nfstext	bin	vs	sm	non	nofa	wr	IBM	6	238892	Int	1	BROKE
20MB	nfstext	bin	vs	sm	non	nofa	wr	IBM	6	222076	Int	1	BROKE
100MB	nfstext	bin	vs	sm	non	nofa	wr	IBM	6	220274	Int	1	BROKE
48KB	nfstext	bin	vs	sm	non	nofa	rd	IBM	6	145041	Int	1	BROKE
256KB	nfstext	bin	vs	sm	non	nofa	rd	IBM	6	443468	Int	1	BROKE
20MB	nfstext	bin	vs	sm	non	nofa	rd	IBM	6	404951	Int	1	BROKE
100MB	nfstext	bin	vs	sm	non	nofa	rd	IBM	6	433787	Int	1	BROKE
48KB	nfstext	bin	vs	sm	sms	fast	wr	IBM	6	197824	Int	1	BROKE
256KB	nfstext	bin	vs	sm	sms	fast	wr	IBM	6	234446	Int	1	BROKE
20MB	nfstext	bin	vs	sm	sms	fast	wr	IBM	6	222038	Int	1	BROKE
100MB	nfstext	bin	vs	sm	sms	fast	wr	IBM	6	220613	Int	1	BROKE
48KB	nfstext	bin	vs	sm	sms	fast	rd	IBM	6	117846	Int	1	BROKE
256KB	nfstext	bin	vs	sm	sms	fast	rd	IBM	6	370244	Int	1	BROKE
20MB	nfstext	bin	vs	sm	sms	fast	rd	IBM	6	651414	Int	1	BROKE
100MB	nfstext	bin	vs	sm	sms	fast	rd	IBM	6	616563	Int	1	BROKE
48KB	nfstext	bin	vs	sm	sms	nofa	wr	IBM	6	206866	Int	1	BROKE
256KB	nfstext	bin	vs	sm	sms	nofa	wr	IBM	6	236501	Int	1	BROKE
20MB	nfstext	bin	vs	sm	sms	nofa	wr	IBM	6	221842	Int	1	BROKE
100MB	nfstext	bin	vs	sm	sms	nofa	wr	IBM	6	220644	Int	1	BROKE
48KB	nfstext	bin	vs	sm	sms	nofa	rd	IBM	6	123255	Int	1	BROKE
256KB	nfstext	bin	vs	sm	sms	nofa	rd	IBM	6	364188	Int	1	BROKE
20MB	nfstext	bin	vs	sm	sms	nofa	rd	IBM	6	656068	Int	1	BROKE
100MB	nfstext	bin	vs	sm	sms	nofa	rd	IBM	6	665421	Int	1	BROKE

48KB	nfstext	txt	ps	fb	non	fast	wr	IBM	6	125219	Int	6	199566	159%
256KB	nfstext	txt	ps	fb	non	fast	wr	IBM	6	142761	Int	6	244398	171%
20MB	nfstext	txt	ps	fb	non	fast	wr	IBM	6	144948	Int	6	191989	132%
100MB	nfstext	txt	ps	fb	non	fast	wr	IBM	6	140928	Int	2	99071	70%
48KB	nfstext	txt	ps	fb	non	fast	rd	IBM	6	135741	Int	6	159083	117%
256KB	nfstext	txt	ps	fb	non	fast	rd	IBM	6	226969	Int	6	420839	185%
20MB	nfstext	txt	ps	fb	non	fast	rd	IBM	6	252720	Int	6	391767	155%
100MB	nfstext	txt	ps	fb	non	fast	rd	IBM	6	201595	Int	2	190941	94%
48KB	nfstext	txt	ps	fb	non	nofa	wr	IBM	6	108664	Int	6	201465	185%
256KB	nfstext	txt	ps	fb	non	nofa	wr	IBM	6	138062	Int	6	244589	177%
20MB	nfstext	txt	ps	fb	non	nofa	wr	IBM	6	145215	Int	6	191881	132%
100MB	nfstext	txt	ps	fb	non	nofa	wr	IBM	6	140870	Int	1	98236	69%
48KB	nfstext	txt	ps	fb	non	nofa	rd	IBM	6	150546	Int	6	166290	110%
256KB	nfstext	txt	ps	fb	non	nofa	rd	IBM	6	242306	Int	6	332537	137%
20MB	nfstext	txt	ps	fb	non	nofa	rd	IBM	6	275766	Int	6	306618	111%
100MB	nfstext	txt	ps	fb	non	nofa	rd	IBM	6	192937	Int	2	181443	94%
48KB	nfstext	txt	ps	fb	sms	fast	wr	IBM	6	120728	Int	6	200660	166%
256KB	nfstext	txt	ps	fb	sms	fast	wr	IBM	6	140166	Int	6	247349	176%
20MB	nfstext	txt	ps	fb	sms	fast	wr	IBM	6	144931	Int	6	192051	132%
100MB	nfstext	txt	ps	fb	sms	fast	wr	IBM	6	140983	Int	2	99029	70%
48KB	nfstext	txt	ps	fb	sms	fast	rd	IBM	6	161791	Int	6	153823	95%
256KB	nfstext	txt	ps	fb	sms	fast	rd	IBM	6	322150	Int	6	434474	134%
20MB	nfstext	txt	ps	fb	sms	fast	rd	IBM	6	406691	Int	6	390662	96%
100MB	nfstext	txt	ps	fb	sms	fast	rd	IBM	6	374831	Int	2	192336	51%
48KB	nfstext	txt	ps	fb	sms	nofa	wr	IBM	6	121699	Int	6	198742	163%
256KB	nfstext	txt	ps	fb	sms	nofa	wr	IBM	6	141235	Int	6	243594	172%
20MB	nfstext	txt	ps	fb	sms	nofa	wr	IBM	6	144931	Int	6	192105	132%
100MB	nfstext	txt	ps	fb	sms	nofa	wr	IBM	6	141184	Int	2	99141	70%
48KB	nfstext	txt	ps	fb	sms	nofa	rd	IBM	6	161891	Int	6	161954	100%
256KB	nfstext	txt	ps	fb	sms	nofa	rd	IBM	6	330042	Int	6	335012	101%
20MB	nfstext	txt	ps	fb	sms	nofa	rd	IBM	6	406951	Int	6	304776	74%
100MB	nfstext	txt	ps	fb	sms	nofa	rd	IBM	6	390630	Int	2	172010	44%
48KB	nfstext	txt	ps	vb	non	fast	wr	IBM	6	114238	Int	6	182057	159%
256KB	nfstext	txt	ps	vb	non	fast	wr	IBM	6	125620	Int	6	222858	177%
20MB	nfstext	txt	ps	vb	non	fast	wr	IBM	6	128018	Int	6	183317	143%
100MB	nfstext	txt	ps	vb	non	fast	wr	IBM	6	127156	Int	2	98824	77%
48KB	nfstext	txt	ps	vb	non	fast	rd	IBM	6	124717	Int	6	170071	136%
256KB	nfstext	txt	ps	vb	non	fast	rd	IBM	6	180620	Int	6	381158	211%
20MB	nfstext	txt	ps	vb	non	fast	rd	IBM	6	194183	Int	6	330227	170%
100MB	nfstext	txt	ps	vb	non	fast	rd	IBM	6	140195	Int	2	177320	126%
48KB	nfstext	txt	ps	vb	non	nofa	wr	IBM	6	110077	Int	6	189912	172%
256KB	nfstext	txt	ps	vb	non	nofa	wr	IBM	6	125403	Int	6	230865	184%
20MB	nfstext	txt	ps	vb	non	nofa	wr	IBM	6	127901	Int	6	183829	143%
100MB	nfstext	txt	ps	vb	non	nofa	wr	IBM	6	126960	Int	2	98118	77%
48KB	nfstext	txt	ps	vb	non	nofa	rd	IBM	6	119409	Int	6	159400	133%
256KB	nfstext	txt	ps	vb	non	nofa	rd	IBM	6	166757	Int	6	272709	163%
20MB	nfstext	txt	ps	vb	non	nofa	rd	IBM	6	183640	Int	6	236155	128%
100MB	nfstext	txt	ps	vb	non	nofa	rd	IBM	6	54734	Int	2	153255	279%

48KB	nfstext	txt	ps	vb	sms	fast	wr	IBM	6	110293	Int	6	189185	171%
256KB	nfstext	txt	ps	vb	sms	fast	wr	IBM	6	125934	Int	6	231580	183%
20MB	nfstext	txt	ps	vb	sms	fast	wr	IBM	6	127888	Int	6	184149	143%
100MB	nfstext	txt	ps	vb	sms	fast	wr	IBM	6	127010	Int	2	98647	77%
48KB	nfstext	txt	ps	vb	sms	fast	rd	IBM	6	148166	Int	6	173079	116%
256KB	nfstext	txt	ps	vb	sms	fast	rd	IBM	6	247442	Int	6	367863	148%
20MB	nfstext	txt	ps	vb	sms	fast	rd	IBM	6	249595	Int	6	331487	132%
100MB	nfstext	txt	ps	vb	sms	fast	rd	IBM	6	248167	Int	2	177399	71%
48KB	nfstext	txt	ps	vb	sms	nofa	wr	IBM	6	112647	Int	6	189308	168%
256KB	nfstext	txt	ps	vb	sms	nofa	wr	IBM	6	124096	Int	6	227679	183%
20MB	nfstext	txt	ps	vb	sms	nofa	wr	IBM	6	127836	Int	6	183746	143%
100MB	nfstext	txt	ps	vb	sms	nofa	wr	IBM	6	126982	Int	2	98323	77%
48KB	nfstext	txt	ps	vb	sms	nofa	rd	IBM	6	146858	Int	6	156937	106%
256KB	nfstext	txt	ps	vb	sms	nofa	rd	IBM	6	244420	Int	6	272138	111%
20MB	nfstext	txt	ps	vb	sms	nofa	rd	IBM	6	208876	Int	6	247653	118%
100MB	nfstext	txt	ps	vb	sms	nofa	rd	IBM	6	227702	Int	2	152686	67%
48KB	nfstext	txt	vs	bi	non	fast	wr	IBM	6	203143	Int	6	247404	121%
256KB	nfstext	txt	vs	bi	non	fast	wr	IBM	6	169475	Int	6	332233	196%
20MB	nfstext	txt	vs	bi	non	fast	wr	IBM	5	152827	Int	6	223452	146%
100MB	nfstext	txt	vs	bi	non	fast	wr	IBM	5	150783	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	non	fast	rd	IBM	6	BROKE	Int	6	165347	
256KB	nfstext	txt	vs	bi	non	fast	rd	IBM	6	BROKE	Int	6	453555	
20MB	nfstext	txt	vs	bi	non	fast	rd	IBM	6	BROKE	Int	6	358397	
100MB	nfstext	txt	vs	bi	non	fast	rd	IBM	6	BROKE	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	non	nofa	wr	IBM	6	208504	Int	6	246830	118%
256KB	nfstext	txt	vs	bi	non	nofa	wr	IBM	6	170182	Int	6	337542	198%
20MB	nfstext	txt	vs	bi	non	nofa	wr	IBM	5	153957	Int	6	224832	146%
100MB	nfstext	txt	vs	bi	non	nofa	wr	IBM	4	150374	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	non	nofa	rd	IBM	6	133903	Int	6	174609	130%
256KB	nfstext	txt	vs	bi	non	nofa	rd	IBM	6	290470	Int	6	431793	148%
20MB	nfstext	txt	vs	bi	non	nofa	rd	IBM	6	264267	Int	6	358411	135%
100MB	nfstext	txt	vs	bi	non	nofa	rd	IBM	6	251345	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	sms	fast	wr	IBM	6	192905	Int	6	242297	125%
256KB	nfstext	txt	vs	bi	sms	fast	wr	IBM	6	169418	Int	6	337098	198%
20MB	nfstext	txt	vs	bi	sms	fast	wr	IBM	4	153501	Int	6	222188	144%
100MB	nfstext	txt	vs	bi	sms	fast	wr	IBM	3	150549	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	sms	fast	rd	IBM	6	160694	Int	6	154999	96%
256KB	nfstext	txt	vs	bi	sms	fast	rd	IBM	6	354158	Int	6	444451	125%
20MB	nfstext	txt	vs	bi	sms	fast	rd	IBM	6	433627	Int	6	355219	81%
100MB	nfstext	txt	vs	bi	sms	fast	rd	IBM	6	414570	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	sms	nofa	wr	IBM	6	202998	Int	6	240554	118%
256KB	nfstext	txt	vs	bi	sms	nofa	wr	IBM	6	170229	Int	6	334880	196%
20MB	nfstext	txt	vs	bi	sms	nofa	wr	IBM	6	153581	Int	6	222345	144%
100MB	nfstext	txt	vs	bi	sms	nofa	wr	IBM	4	150266	Int	6	BROKE	
48KB	nfstext	txt	vs	bi	sms	nofa	rd	IBM	6	153098	Int	6	160334	104%
256KB	nfstext	txt	vs	bi	sms	nofa	rd	IBM	6	351719	Int	6	443752	126%
20MB	nfstext	txt	vs	bi	sms	nofa	rd	IBM	6	438278	Int	6	353596	80%
100MB	nfstext	txt	vs	bi	sms	nofa	rd	IBM	6	410068	Int	6	BROKE	

48KB	nfstext	txt	ps	vb	sms	fast	rd	IBM	6	148166	Int	6	173079	116%
48KB	nfstext	txt	vs	sm	non	fast	wr	IBM	6	110955	Int	6	56445	50%
256KB	nfstext	txt	vs	sm	non	fast	wr	IBM	6	125739	Int	6	214633	170%
20MB	nfstext	txt	vs	sm	non	fast	wr	IBM	6	74008	Int	5	75989	102%
100MB	nfstext	txt	vs	sm	non	fast	wr	IBM	5	57425	Int	5	BROKE	
48KB	nfstext	txt	vs	sm	non	fast	rd	IBM	6	BROKE	Int	6	16041	
256KB	nfstext	txt	vs	sm	non	fast	rd	IBM	6	BROKE	Int	6	336595	
20MB	nfstext	txt	vs	sm	non	fast	rd	IBM	6	BROKE	Int	6	136168	
100MB	nfstext	txt	vs	sm	non	fast	rd	IBM	6	BROKE	Int	6	BROKE	
48KB	nfstext	txt	vs	sm	non	nofa	wr	IBM	6	120808	Int	6	231598	191%
256KB	nfstext	txt	vs	sm	non	nofa	wr	IBM	6	126247	Int	6	217285	172%
20MB	nfstext	txt	vs	sm	non	nofa	wr	IBM	6	74979	Int	6	77343	103%
100MB	nfstext	txt	vs	sm	non	nofa	wr	IBM	5	57600	Int	1	47000	81%
48KB	nfstext	txt	vs	sm	non	nofa	rd	IBM	6	26941	Int	6	13716	50%
256KB	nfstext	txt	vs	sm	non	nofa	rd	IBM	6	236835	Int	6	347272	146%
20MB	nfstext	txt	vs	sm	non	nofa	rd	IBM	6	202067	Int	6	137618	68%
100MB	nfstext	txt	vs	sm	non	nofa	rd	IBM	6	127515	Int	1	75797	59%
48KB	nfstext	txt	vs	sm	sms	fast	wr	IBM	5	117012	Int	6	67043	57%
256KB	nfstext	txt	vs	sm	sms	fast	wr	IBM	5	125347	Int	6	211094	168%
20MB	nfstext	txt	vs	sm	sms	fast	wr	IBM	5	74835	Int	6	75217	100%
100MB	nfstext	txt	vs	sm	sms	fast	wr	IBM	4	57315	Int	6	BROKE	
48KB	nfstext	txt	vs	sm	sms	fast	rd	IBM	5	55570	Int	6	15215	27%
256KB	nfstext	txt	vs	sm	sms	fast	rd	IBM	5	242307	Int	6	334759	138%
20MB	nfstext	txt	vs	sm	sms	fast	rd	IBM	5	358102	Int	6	137309	38%
100MB	nfstext	txt	vs	sm	sms	fast	rd	IBM	5	226254	Int	6	BROKE	
48KB	nfstext	txt	vs	sm	sms	nofa	wr	IBM	6	101275	Int	6	118060	116%
256KB	nfstext	txt	vs	sm	sms	nofa	wr	IBM	6	125385	Int	6	214822	171%
20MB	nfstext	txt	vs	sm	sms	nofa	wr	IBM	6	75075	Int	6	77334	103%
100MB	nfstext	txt	vs	sm	sms	nofa	wr	IBM	5	57115	Int	6	BROKE	
48KB	nfstext	txt	vs	sm	sms	nofa	rd	IBM	6	72801	Int	6	16042	22%
256KB	nfstext	txt	vs	sm	sms	nofa	rd	IBM	6	321665	Int	6	353096	109%
20MB	nfstext	txt	vs	sm	sms	nofa	rd	IBM	5	362968	Int	6	136926	37%
100MB	nfstext	txt	vs	sm	sms	nofa	rd	IBM	5	275700	Int	6	BROKE	

The data from the NFS Single Session Throughput table suggest several general trends:

- For small file sizes (48KB and 256KB), Interlink NFS, on average, provides faster throughput than IBM NFS. However, there are a number of exceptions to this trend.
- For large file sizes (100MB and 250MB), IBM NFS, on average, provides faster throughput than Interlink NFS.
- For medium file size (20MB), IBM and Interlink NFS, on average, are about even in throughput performance, although individual tests exhibit significant variation.
- Although neither vendor completed all the tests without failure, Interlink NFS was considerably less reliable than IBM NFS for certain groups of tests. Interlink's weak points appear to be related to VSAM files and very large files.

4.2 NFS Multiple Session Throughput (Ethernet)

Throughput benchmarks for concurrent multiple NFS clients were conducted over Ethernet, primarily because the FDDI ring was not configured with three RS/6000 clients. Each NFS client ran on a separate workstation. The workstations were located at CITI (refer to Figure 1), and the NFS

test packets traversed the UM campus FDDI backbone in addition to the two Ethernet links at either end of of the backbone. The communications controllers used were an IBM 3172-3 and an Interlink BTI controller. Due to the relatively slow speed of Ethernet, large files (100MB and 250MB) were not used in the multiple NFS client tests. To save time, tests were limited to MVS physical sequential data sets.

Benchmarks were taken for one, two, and three concurrent NFS sessions. For a test to be considered successful, each session used in the test had to complete successfully. More specifically, the mount, read or write, and unmount operations had to be successful.

TABLE 5. *One Session (Ethernet)*

File Size	Test Routine	bin txt	ps vs	fb vb sm bi	sms non	fast nova	rd wr	IBM	#	Average Bytes Per Second	Int	#	Average Bytes Per Second	Percent Difference: Interlink/ IBM
256KB	rpct5d	bin	ps	fb	non	fast	wr	IBM	1	193036	Int	1	251819	130%
20MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	1	234580	Int	1	290867	123%
256KB	rpct5d	bin	ps	fb	non	fast	rd	IBM	1	165182	Int	1	319687	193%
20MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	1	147686	Int	1	304376	206%
256KB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	1	195047	Int	1	251819	129%
20MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	1	231473	Int	1	290062	125%
256KB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	1	195192	Int	1	346751	177%
20MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	1	175347	Int	1	305707	174%
256KB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	1	194324	Int	1	246607	126%
20MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	1	239674	Int	1	290464	121%
256KB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	1	213995	Int	1	313944	146%
20MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	1	251155	Int	1	305707	121%
256KB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	1	208050	Int	1	245913	118%
20MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	1	237234	Int	1	286888	120%
256KB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	1	212434	Int	1	328501	154%
20MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	1	248183	Int	1	305262	122%

TABLE 6. *Two Sessions (Ethernet)*

File Size	Test Routine	bin txt	ps vs	fb vb sm bi	sms non	fast nova	rd wr	IBM	#	Average Bytes Per Second	Int	#	Average Bytes Per Second	Percent Difference: Interlink/ IBM
256KB	rpct5d	bin	ps	fb	non	fast	wr	IBM	2	183671	Int	2	208217	113%
20MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	2	188180	Int	2	204897	108%
256KB	rpct5d	bin	ps	fb	non	fast	rd	IBM	2	150431	Int	2	288526	191%
20MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	2	140043	Int	2	215980	154%
256KB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	2	187081	Int	2	207561	110%
20MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	2	184048	Int	2	209797	113%
256KB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	2	179000	Int	2	299275	167%
20MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	2	181997	Int	2	196638	108%
256KB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	2	182107	Int	2	202624	111%
20MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	2	184181	Int	2	201628	109%
256KB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	2	168869	Int	2	314074	185%
20MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	2	217772	Int	2	216872	99%
256KB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	2	176885	Int	2	245178	138%
20MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	2	189959	Int	2	202530	106%
256KB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	2	183370	Int	2	321810	175%
20MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	2	220288	Int	2	213995	97%

TABLE 7. *Three Sessions (Ethernet)*

File Size	Test Routine	bin txt	ps vs	fb vb sm bi	sms non	fast nova	rd wr	IBM	#	Average Bytes Per Second	Int	#	Average Bytes Per Second	Percent Difference: Interlink/ IBM
256KB	rpct5d	bin	ps	fb	non	fast	wr	IBM	3	149973	Int	3	234073	156%
20MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	3	156701	Int	3	272161	173%
256KB	rpct5d	bin	ps	fb	non	fast	rd	IBM	3	113937	Int	3	289042	253%
20MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	3	105504	Int	2	215596	N/A
256KB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	3	122953	Int	3	160210	130%
20MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	3	156665	Int	2	202933	N/A
256KB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	3	141829	Int	3	298672	210%
20MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	3	154605	Int	2	224128	N/A
256KB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	3	151741	Int	2	201464	N/A
20MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	3	173227	Int	1	271300	N/A
256KB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	3	170063	Int	2	305223	N/A
20MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	3	165802	Int	1	301314	N/A
256KB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	3	160048	Int	2	205934	N/A
20MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	3	173318	Int	2	217074	N/A
256KB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	3	177544	Int	2	319551	N/A
20MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	3	177518	Int	2	233610	N/A

The results indicate that for one and two concurrent NFS sessions over Ethernet, and for the file types and sizes tested, Interlink NFS provides faster average throughput than IBM. For three concurrent NFS sessions, one or more of the Interlink sessions failed to complete in eleven of the sixteen test cases. The Percent Difference column in these test cases is thus labled N/A. As previously mentioned, NFS throughput times are based upon performance of both NFS and TCP/IP.

4.3 CPU Utilization

CPU utilization on MVS was measured for 100 MB physical sequential files running over an FDDI ring. Tests were conducted with smaller files, but the precision of CPU activity measurements (due to short transfer times) was deemed inadequate for inclusion in this report. In the NFS CPU utilization tests, CPU busy time is broken down into NFS task time and TCP/IP task time. To provide a more useful metric, the CPU busy time is normalized by dividing the CPU time (in milliseconds) by the number of bytes transfered during an NFS read or write. Thus the CPU times listed are in units of milliseconds per megabyte of NFS data. The last column of the tables provides a comparison of total CPU utilization between the IBM and Interlink NFS servers.

Table 8 only includes explanations for column headings that are unique to the NFS CPU utilization tests.

TABLE 8. *NFS CPU Utilization Report Headings*

Column Heading	Description of Column
Ave KB Per Sec	Average NFS throughput, measured in kilobytes per second.
NFS CPU Per MB	Milliseconds of MVS NFS server CPU time per megabyte of NFS data transferred.
TCP CPU Per MB	Milliseconds of MVS TCP/IP task CPU time per megabyte of NFS data transferred.
% MB Dif	The percent difference in NFS throughput, between IBM and Interlink. Calculated by dividing Interlink throughput by IBM throughput (x 100). Numbers less than 100 favor IBM and numbers over 100 favor Interlink.
% NFS Dif	The percent difference for NFS server CPU time, between IBM and Interlink, taken from the NFS CPU Per MB column. Calculated by dividing Interlink NFS CPU by IBM NFS CPU (x 100). Numbers over 100 favor IBM and numbers under 100 favor Interlink.
% TCP Dif	The percent difference for the TCP/IP task CPU time, between IBM and Interlink, taken from the TCP CPU Per MB column. Calculated by dividing Interlink TCP/IP CPU by IBM TCP/IP CPU (x 100). Numbers over 100 favor IBM and numbers under 100 favor Interlink.
%CPU Dif	The percent difference for all MVS CPU (NFS and TCP/IP), between IBM and Interlink. Calculated by dividing Interlink total CPU by IBM total CPU (x 100). Numbers over 100 favor IBM and numbers under 100 favor Interlink.

TABLE 9. *NFS CPU Utilization Test Results*

File Size	Test Routine	bin txt	ps vs	fb vb sm bi	sms non	fast nofa	rd wr	IBM	#	Ave KB Per Sec	NFS CPU Per MB	TCP CPU Per MB	Int	#	Ave KB Per Sec	NFS CPU Per MB	TCP CPU Per MB	% MB Dif	% NFS Dif	% TCP Dif	% CPU Dif
100MB	rpct5d	bin	ps	fb	non	fast	wr	IBM	2	743	84	58	Int	2	235	1511	58	31	1798	100	1104
100MB	rpct5d	bin	ps	fb	non	fast	rd	IBM	2	321	454	184	Int	2	217	1741	62	67	383	33	282
100MB	rpct5d	bin	ps	fb	non	nofa	wr	IBM	2	750	71	49	Int	2	236	1551	64	31	2184	130	1345
100MB	rpct5d	bin	ps	fb	non	nofa	rd	IBM	2	359	199	130	Int	2	222	1742	63	61	875	48	548
100MB	rpct5d	bin	ps	fb	sms	fast	wr	IBM	2	746	89	60	Int	2	235	1527	60	31	1715	100	1065
100MB	rpct5d	bin	ps	fb	sms	fast	rd	IBM	2	664	90	67	Int	2	215	1793	63	32	1992	94	1182
100MB	rpct5d	bin	ps	fb	sms	nofa	wr	IBM	2	752	78	53	Int	2	232	1497	59	30	1919	111	1187
100MB	rpct5d	bin	ps	fb	sms	nofa	rd	IBM	2	645	103	77	Int	2	222	1743	64	34	1692	83	1003
Averages (bin)								IBM		623	146	85	Int		227	1638	62	**40**	1570	87	**965**
100MB	nfstext	txt	ps	fb	non	fast	wr	IBM	2	157	1172	385	Int	2	107	7608	219	67	649	56	502
100MB	nfstext	txt	ps	fb	non	fast	rd	IBM	2	233	667	151	Int	2	182	2565	80	78	384	52	323
100MB	nfstext	txt	ps	fb	non	nofa	wr	IBM	2	150	1833	527	Int	2	106	7622	222	70	415	42	332
100MB	nfstext	txt	ps	fb	non	nofa	rd	IBM	2	211	1150	207	Int	2	159	3292	96	75	286	46	249
100MB	nfstext	txt	ps	fb	sms	fast	wr	IBM	2	153	1183	372	Int	2	99	8086	231	65	683	62	534
100MB	nfstext	txt	ps	fb	sms	fast	rd	IBM	2	383	255	78	Int	2	173	2752	86	45	1079	110	852
100MB	nfstext	txt	ps	fb	sms	nofa	wr	IBM	2	162	1182	385	Int	2	105	7624	215	65	645	55	500
100MB	nfstext	txt	ps	fb	sms	nofa	rd	IBM	2	357	305	87	Int	2	160	3210	95	44	1052	109	843
Averages (txt)								IBM		226	968	274	Int		136	5345	156	**64**	649	67	**517**
Averages (total)								IBM		424	557	179	Int		182	3492	109	**52**	1109	77	**741**

The NFS CPU utilization data suggests several trends. These trends may only be valid for the 100 MB files used in the CPU tests. Tests with smaller file sizes may have shown different trends, but instrumenting such tests proved to be beyond the scope of this project.

- For both vendors, NFS text mode tests used significantly more CPU time, and ran more slowly, than equivilant NFS binary mode tests. This was not a suprise, because in NFS text mode data must be translated between ASCII and EBCDIC, and end-of-line characters must be inserted/stripped. The Interlink UFS Support option was not used in these tests, nor was the IBM OpenEdition file system. Because both of these systems store data in ASCII format, NFS client access to text data might have been quicker if they had been used in testing rather than the traditional MVS file system.
- On average, Interlink NFS throughput was only 52 percent of IBM NFS throughput. As noted earlier, this was typical behavior for large test files. Interlink performed better with smaller test files, which are not part of the NFS CPU utilization test results.
- TCP/IP CPU utilization, while running the Interlink and IBM NFS servers, was relatively close. On average, Interlink used about 77 percent of the TCP/IP task CPU cycles (per megabyte of data transferred) that IBM used.
- The IBM NFS server used significantly less CPU than the Interlink NFS server for 100 MB test files. On average, the Interlink NFS server used about 1109 percent more CPU cycles (per megabyte of data transferred) than the IBM NFS server.

Overall NFS CPU utilization (TCP/IP task plus NFS server task) shows that the Interlink NFS server used about 741 percent more CPU cycles (per megabyte of data transferred) than the IBM NFS server.

4.4 Average NFS Command Response Time

To test performance of low-level NFS commands, IBM supplied a program named nfstest. The nfstest program used hard coded NFS attributes, specifically: recfm(vb), lrecl(9000), blksize(32000), space(1,0), trks, and binary. The nfstest program ran on an RS/6000 workstation, networked via an FDDI ring. The data sets used in the test were not SMS data sets.

Listed below is a chart summarizing the low-level NFS command results, followed by logs of the nfstest runs (two for IBM and two for Interlink). Note that operation times are given in milliseconds per operation call, so lower numbers indicate better performance. The last column compares Interlink and IBM performance by dividing the Interlink operation call time by the IBM operation call time.

TABLE 10. *Low-Level NFS Command Response Time*

NFS Command	Interlink (msec/call)	IBM (msec/call)	Interlink/IBM (% difference)
NULL	4.4	11.6	38%
GETATTR	5.4	13.1	41%
SETATTR	112.1	39.8	281%
LOOKUP	34.1	30.1	113%
READ	7.1	17.1	41%
WRITE	7.6	19.9	38%
CREATE	199.9	182.3	110%
REMOVE	80.8	90.4	89%
RENAME	174.9	152.7	115%
MKDIR	177.9	243.3	73%
RMDIR	206.3	159.4	129%
READDIR	37.0	19.7	188%
STATFS	5.3	12.9	41%

4.5 Interlink NFS Server (nfstest client)

TABLE 11. *Interlink nfstest #1*

```
<vegas> Test run on vegas to server elc3
<vegas> Started at Tue Apr 18 05:52:46 1995

<vegas> elc3:nfs.bench94 mounted ok.
Creating vegas.root.TESTFILE ...
Writing vegas.root.TESTFILE ...
Reading vegas.root.TESTFILE ...
Removing vegas.root.TESTFILE ...
Statfs vegas.root.TESTFILE ...
<vegas>          NULL   Proc    100  calls,      442  msec,     4.42  msec/call
<vegas>       GETATTR   Proc     25  calls,      136  msec,     5.44  msec/call
<vegas>       SETATTR   Proc     25  calls,     2792  msec,   111.68  msec/call
<vegas>        LOOKUP   Proc     25  calls,      844  msec,    33.76  msec/call
<vegas>          READ   Proc    100  calls,      698  msec,     6.98  msec/call,  143.27
Kbytes/sec
<vegas>         WRITE   Proc    100  calls,      759  msec,     7.59  msec/call,  131.75
Kbytes/sec
<vegas>        CREATE   Proc     27  calls,     5450  msec,   201.85  msec/call
<vegas>        REMOVE   Proc     27  calls,     2189  msec,    81.07  msec/call
<vegas>        RENAME   Proc     24  calls,     4162  msec,   173.42  msec/call
<vegas>         MKDIR   Proc     15  calls,     2606  msec,   173.73  msec/call
<vegas>         RMDIR   Proc     15  calls,     3101  msec,   206.73  msec/call
<vegas>       READDIR   Proc     50  calls,     1863  msec,    37.26  msec/call
<vegas>        STATFS   Proc    100  calls,      526  msec,     5.26  msec/call
<vegas> elc3:nfs.bench94 unmounted ok.

<vegas> Test elapsed time : 27 seconds
```

TABLE 12. *Interlink nfstest #2*

```
<vegas> Test run on vegas to server elc3
<vegas> Started at Tue Apr 18 05:53:38 1995

<vegas> elc3:nfs.bench94 mounted ok.
Creating vegas.root.TESTFILE ...
Writing vegas.root.TESTFILE ...
Reading vegas.root.TESTFILE ...
Removing vegas.root.TESTFILE ...
Statfs vegas.root.TESTFILE ...
<vegas>          NULL   Proc    100  calls,      438  msec,     4.38  msec/call
<vegas>       GETATTR   Proc     25  calls,      134  msec,     5.36  msec/call
<vegas>       SETATTR   Proc     25  calls,     2812  msec,   112.48  msec/call
<vegas>        LOOKUP   Proc     25  calls,      863  msec,    34.52  msec/call
<vegas>          READ   Proc    100  calls,      715  msec,     7.15  msec/call,  139.86
Kbytes/sec
<vegas>         WRITE   Proc    100  calls,      768  msec,     7.68  msec/call,  130.21
Kbytes/sec
<vegas>        CREATE   Proc     27  calls,     5344  msec,   197.93  msec/call
<vegas>        REMOVE   Proc     27  calls,     2174  msec,    80.52  msec/call
<vegas>        RENAME   Proc     24  calls,     4232  msec,   176.33  msec/call
<vegas>         MKDIR   Proc     15  calls,     2731  msec,   182.07  msec/call
<vegas>         RMDIR   Proc     15  calls,     3088  msec,   205.87  msec/call
<vegas>       READDIR   Proc     50  calls,     1841  msec,    36.82  msec/call
<vegas>        STATFS   Proc    100  calls,      527  msec,     5.27  msec/call
<vegas> elc3:nfs.bench94 unmounted ok.

<vegas> Test elapsed time : 27 seconds

***** END Interlink *****
```

4.6 IBM NFS server (nfstest client)

TABLE 13. *IBM nfstest #1*

```
<vegas> Test run on vegas to server justinf
<vegas> Started at Tue Apr 18 05:59:09 1995

<vegas> justinf:nfs.bench94 mounted ok.
Creating vegas.root.TESTFILE ...
Writing vegas.root.TESTFILE ...
Reading vegas.root.TESTFILE ...
Removing vegas.root.TESTFILE ...
Statfs vegas.root.TESTFILE ...
<vegas>          NULL   Proc   100   calls,   1151   msec,   11.51   msec/call
<vegas>       GETATTR   Proc    25   calls,    331   msec,   13.24   msec/call
<vegas>       SETATTR   Proc    25   calls,    983   msec,   39.32   msec/call
<vegas>        LOOKUP   Proc    25   calls,    799   msec,   31.96   msec/call
<vegas>          READ   Proc   100   calls,   1708   msec,   17.08   msec/call,   58.55
Kbytes/sec
<vegas>         WRITE   Proc   100   calls,   1988   msec,   19.88   msec/call,   50.30
Kbytes/sec
<vegas>        CREATE   Proc    27   calls,   5105   msec,  189.07   msec/call
<vegas>        REMOVE   Proc    27   calls,   2457   msec,   91.00   msec/call
<vegas>        RENAME   Proc    24   calls,   3489   msec,  145.38   msec/call
<vegas>         MKDIR   Proc    15   calls,   3579   msec,  238.60   msec/call
<vegas>         RMDIR   Proc    15   calls,   2396   msec,  159.73   msec/call
<vegas>       READDIR   Proc    50   calls,    969   msec,   19.38   msec/call
<vegas>        STATFS   Proc   100   calls,   1269   msec,   12.69   msec/call
<vegas> justinf:nfs.bench94 unmounted ok.

<vegas> Test elapsed time : 27 seconds
```

TABLE 14. *IBM nfstest #2*

```
<vegas> Test run on vegas to server justinf
<vegas> Started at Tue Apr 18 06:00:05 1995

<vegas> justinf:nfs.bench94 mounted ok.
Creating vegas.root.TESTFILE ...
Writing vegas.root.TESTFILE ...
Reading vegas.root.TESTFILE ...
Removing vegas.root.TESTFILE ...
Statfs vegas.root.TESTFILE ...
<vegas>          NULL   Proc   100   calls,   1172   msec,   11.72   msec/call
<vegas>       GETATTR   Proc    25   calls,    326   msec,   13.04   msec/call
<vegas>       SETATTR   Proc    25   calls,   1008   msec,   40.32   msec/call
<vegas>        LOOKUP   Proc    25   calls,    705   msec,   28.20   msec/call
<vegas>          READ   Proc   100   calls,   1714   msec,   17.14   msec/call,   58.34
Kbytes/sec
<vegas>         WRITE   Proc   100   calls,   1994   msec,   19.94   msec/call,   50.15
Kbytes/sec
<vegas>        CREATE   Proc    27   calls,   4739   msec,  175.52   msec/call
<vegas>        REMOVE   Proc    27   calls,   2425   msec,   89.81   msec/call
<vegas>        RENAME   Proc    24   calls,   3838   msec,  159.92   msec/call
<vegas>         MKDIR   Proc    15   calls,   3720   msec,  248.00   msec/call
<vegas>         RMDIR   Proc    15   calls,   2385   msec,  159.00   msec/call
<vegas>       READDIR   Proc    50   calls,   1001   msec,   20.02   msec/call
<vegas>        STATFS   Proc   100   calls,   1316   msec,   13.16   msec/call
<vegas> justinf:nfs.bench94 unmounted ok.

<vegas> Test elapsed time : 27 seconds

***** END IBM *****
```

Performance for the lookup, create, remove, and rename operations was close (within 15 percent). For the setattr, rmdir and readdir operations, IBM was significantly (greater than 15 percent) faster. For the null, getattr, read, write, mkdir and statfs operations, Interlink was significantly faster. The null operation result may imply that Interlink enjoyed an advantage TCP/IP performance, because the null operation performs no significant NFS activity; this is, of course, conjecture, since other unknown MVS performance factors may also be relevant to the null operation test.

5. Benchmark Category Ratings

This section offers an explanation of worksheet benchmark features that were rated below average. The rating system defines ratings from several perspectives, as shown in Table 15, which is the ratings key for Table 16. In most cases, the scale ranges from 1 to 5, with 1 being the worst case, and 5 being the best case. Ratings of 1 or 2 are defined as below average. The Comparison Indicator denotes IBM's performance with respect to Interlink. A comparison rating of Poor is considered below average. All rating catagories also have a N/A rating, which means the rating system was somehow not applicable, or did not apply in the context of the evaluated feature.

TABLE 15. *Ratings Key*

Category	Rating	Description	Below Average
Satisfaction	S1	very unsatisfied	yes
	S2	unsatisfied	yes
	S3	neutral	no
	S4	satisfied	no
	S5	very satisfied	no
Ease of Use	E1	very difficult	yes
	E2	difficult	yes
	E3	easy	no
	E4	very easy	no
	E5	the best	no
Accessibility and Completeness	A1	very incomplete	yes
	A2	incomplete	yes
	A3	adequate	no
	A4	complete	no
	A5	very complete	no
Clarity	C1	very unclear	yes
	C2	unclear	yes
	C3	adequate	no
	C4	clear	no
	C5	extremely clear	no
Comparison Indicator	(P)	poor	yes
	(S)	same	no
	(B)	better	no

TABLE 16. *Below Average Benchmark Ratings*

Vendor	Item	Rating	Explanation
Interlink	Support OpenEdition MVS file system	S1, N/A	Interlink NFS did not support MVS OpenEdition when these benchmarks were conducted.
Interlink	Ability to share data between NFS client user and OpenEdition MVS user	S1, N/A	Interlink NFS did not support MVS OpenEdition during the period of our benchmarks. Interlink UFS Support option does not share data with MVS users.
Interlink	Ability to link conventional MVS data sets to a POSIX path name	S2, N/A	No mention of POSIX support in Interlink NFS manuals. IBM OpenEdition, which provides POSIX names, is not supported by Interlink. Interlink UFS Support option, which offers UNIX file semantics, does not share data with MVS users.
Interlink	Support VSAM data set	S2	Problems with Interlink NFS and VSAM. Unable to dynamically allocate VSAM files. Could not read/write pre-allocated VSAM files in binary mode. Problems reading/writing large pre-allocated VSAM files in text mode.
Interlink	Supports SMS constructs (3 items)	S2, E2	Interlink NFS provides site default SMS support, but provides no mechanism to override Data Class, Management Class, or Storage Class attributes with NFS parameters.
Interlink	Persistence of file system attributes and mount handles across a NFS server shutdown and restart for OpenEdition file system	S1, N/A	Interlink NFS does not support MVS OpenEdition files. However, Interlink NFS does provide persistent properties with its UFS Support option.
Interlink	Support Accounting SMF records	S2, E2	Interlink provides an accounting exit. SMF not mentioned in the Installation Guide.
Interlink	Support Translation Table Customization	E2	Interlink requires an exit routine to change the translation table.
Interlink	Support option NONE to bypass security	S2, N/A	No Interlink NFS security option to designate NONE. This could be handled in an exit, however, to provide equivilant functionality.
Interlink	Support UNIX permission checking for MVS OpenEdition file system	S1, N/A	Interlink NFS does not support MVS OpenEdition files.
Interlink	Create a PDSE directory from a client.	S2, A2	We did not have a chance to test this. The Interlink NFS manuals made no mention of PDSE data sets.
Interlink	Create a PDSE member from a client.	S2, A2	(Same as above.)
Interlink	Create a VSAM data set from a client.	S1	All attempts to dynamically (from a client) create a VSAM data set with Interlink NFS failed. We called Interlink, but they could not solve our problem.
Interlink	Suspend the writing of SMF records for NFS.	S1, A1	Interlink NFS does not directly support such an option. It may be possible via an exit, but no specific mention is made of SMF records.
Interlink	Resume the writing of SMF records for NFS.	S1, A1	(Same as above.)
Interlink	Change the logout attribute values to 2100 seconds.	S2, A2	Could not find a reference to changing the logout attribute value in the Interlink NFS manuals.

TABLE 16. *Below Average Benchmark Ratings*

Vendor	Item	Rating	Explanation
Interlink	NFS providing the needed function to do your work	S2	In general, Interlink has too many functional shortcomings, as listed in this table, to be uniformly applicable for the wide range of applications typical in the diverse MVS environment. If one does not care about some of the items critiqued, then Interlink may provide the needed functionality.
IBM and Interlink	The efficiency of NFS in doing your work (i.e., number of commands that are required per actions task)	S2	For files which are the default type, and do not grow larger than the default size, using NFS is reasonable for a computer literate person. For non-defaults file types and sizes, however, entering an NFS mount command is equivilant to writing JCL from scratch. See Section 6, Concluding Remarks.
IBM and Interlink	The user interface (the visual presentation of information on the screens and the method(s) of interacting with the computer) of NFS.	S2	There is no visual interface beyond the command line. An intelligent GUI that presents available options and remembers user preferences and history is needed for the NFS client accessing an MVS NFS server.
Interlink	How easy can you find specific information using the index, table of contents, or headings?	S2	Using the table of contents and the headings of the Interlink manuals did not pose a problem. The index(s), however, were lacking in depth and did not provide adequate detail or cross-referencing to find many of the topics for which we were seeking answers.
Interlink	Was the product reliable from the server's perspective?	S2	Interlink NFS problems with VSAM files and inconsistency in handling very large files (100MB) are responsible for a below average score on this item. On the positive side, the Interlink NFS task did not crash (nor did IBM NFS).
Interlink	Were critical NFS functions consistently available?	S2	Of the functions allegedly supported, VSAM file access either did not work at all, or worked inconsistently, for Interlink NFS. Also, the Interlink NFS method of computing the file size for text files often produced a result different from an equivalent file created on AIX. In other words, UNIX file semantics were not preserved. (This remark does not apply to files created with UFS Support option.)
Interlink	If a failure occurred, what was the least useful source for resolving the failure?	S2	I was not a typical, paying customer, but of the two or three problems that I submitted to Interlink for assistance, not one was solved. To be fair, the person I spoke with was new to the job, and was pleasant and cordial. However, despite taking traces and gathering other diagnostics, we could not together solve the problem of accessing VSAM data sets with Interlink NFS.

5.1 IBM OpenEdition vs Interlink UFS Support

Both IBM and Interlink support NFS access to traditional MVS files, while also allowing local MVS users to access those files (with file sharing restrictions, of course). There is one area, however, where the IBM and Interlink NFS servers offer significantly different choices. In the area of UFS file semantics (POSIX), IBM has chosen to implement a POSIX-compliant file system, MVS OpenEdition, as part of MVS, and offer NFS support for that system. Interlink, on the other hand, has chosen to provide UNIX file semantics with a special UFS Support option. The comparisons

in the following table are an attempt to summarize the similarities and differences between the IBM OpenEdition and the Interlink UFS Support strategies for NFS.

TABLE 17. *IBM OpenEdition vs Interlink UFS Support*

Issue	IBM MVS Open Edition NFS support	Interlink UFS Support for NFS
Installation	Must install MVS OpenEdition file system. Requires MVS 5.1.	Must create one or more Attribute Database files (ADBs) and list the ADBs in the EXPORTS member.
POSIX compliance	Claims to be POSIX compliant.	Claims to be POSIX compliant.
UNIX NFS client access	Provides UFS file semantics. Sufficient OpenEdition disk space must be available. OpenEdition directory is mounted on NFS client.	Provides UFS file semantics. Client must first mount ADB on local file system. The ADB is then used to hold and keep track of files and directories created off that mount point.
MVS NFS access	MVS OpenEdition files are available to MVS users.	UFS Support files are not available to MVS users.
Data format	MVS OpenEdition files are stored in ASCII format.	UFS Support files are stored in ASCII format.
Attribute synchronization across NFS restarts	yes	yes
fast file size	yes	yes
LINK, SYMLINK, READLINK support	yes	yes

6. Concluding Remarks

At the present time, NFS is the only ubiquitous file system (the others being AFS and DFS) for MVS which allows access to all of the native MVS file types. Given the great diversity of file types on MVS, creating an MVS NFS server that reliably serves these files to the outside non-EBCDIC world is no small accomplishment. For the NFS client end user, however, navigating in what is probably a non-MVS environment, creating and accessing files on MVS can be a daunting chore. On MVS we often grab an old JCL member, perhaps making a few changes, and submit the member to run our job in the background. On other platforms, such as UNIX and PC operating systems, the information contained in the JCL member is not available, requiring the user to essentially create JCL on-the-fly.

What is needed to make NFS on MVS more accessible to end users is an intelligent front end to the standard NFS client. Such a front end would understand MVS file types, could communicate with MVS to determine available space, security requirements, etc. and would also provide a GUI interface where options would be presented. Personal profiles and history records could also be maintained. At a minimum, most NFS end users accessing MVS file types which are not the NFS defaults will need to save NFS mount commands for later reference, since typing in long commands is prone to error.

Special thanks to:

Both IBM and Interlink corporations for providing hardware, software and consulting support for the duration of this NFS benchmarking project.

The folks at IBM who provided needed guidance throughout the project: Debbra Markham, Diana Husband and Redem Chiang. Also to Dan Patel, Bob Perrone and Kevin Goin of IBM, Raleigh, for their helpful TCP/IP support.

Tom Mayo of Interlink, who responded to each question I posed with enthusiasm and a caring attitude.