

Fstress User Manual

Darrell Anderson and Jeff Chase
Department of Computer Science
Duke University
{anderson, chase}@cs.duke.edu

March 16, 2002

Fstress is a synthetic, flexible, self-scaling NFS file service benchmark whose primary goal is flexibility. Fstress exports control over several dimensions in both data set and workload, enabling a wide range of tests for fundamental evaluation of file service scalability, sizing, configuration, and other factors. Fstress includes several important “canned” workloads.

Contents

1	Installation	3
1.1	Advanced Options	3
2	Cluster Configuration	4
2.1	Configuring the NFS Server	4
2.2	Configuring the Fstress Clients	4
3	Running the Benchmark	5
3.1	Benchmark Options	5
3.1.1	Configuration Options	5
3.1.2	Intensity Options	5
3.1.3	Log Data Options	7
3.1.4	Workload and Directory Tree Options	7
3.2	Benchmarking and the File Set	7
4	Analyzing the Results	9
4.1	Output Log Files	9
4.1.1	Per-Operation Counts	9
4.1.2	Latency Histograms	10
4.1.3	Summary Line	10
5	Troubleshooting	11
5.1	Mount-test	11
5.2	Readdir-test	11
5.3	Operation-test	12
5.4	Createtree-test	12
5.5	Other Fstress Test Programs	12
5.6	External Tests	12
6	Sample Workloads	13

1 Installation

Any user may run the Fstress benchmark. Because Fstress runs on an array of client machines, it is best to install the distribution in shared space where all clients can access it (e.g., on an NFS server). Chose an appropriate directory, and follow these instructions:

1. Fetch the Fstress distribution from `http://www.cs.duke.edu/ari/fstress`
2. Unpack the Fstress distribution:
`prompt> gzip -dc fstress.tgz | tar xf -`
3. Set the `FSTRESS_HOME` environment variable to the new directory:
`prompt> setenv FSTRESS_HOME "/path/to/fstress"`
4. Build the Fstress benchmark and related tools:
`prompt> cd $FSTRESS_HOME ; gmake ; gmake clean`
5. Create the sample workloads:
`prompt> gmake sample_workloads`

1.1 Advanced Options

Fstress can use the GNU general precision math library (GMP) for better statistics. This package is included with FreeBSD and enabled when building Fstress on a FreeBSD host.

If you wish to use GMP on other systems, install the GMP library, set `USE_GMP=yes` in the Fstress Makefile, and rebuild the Fstress benchmark.

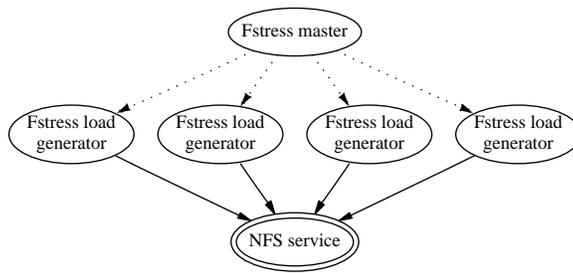


Figure 1: Fstress control structure.

2 Cluster Configuration

Fstress benchmarks a NFS version 3 server using an array of load generating clients. We developed the load generator and toolset under FreeBSD and ported them to Linux and Solaris.

Fstress uses a simple master/slave control structure, depicted in Figure 1. The master process invokes and coordinates remote slave processes, with one process per client machine. The master acts only as a control point, all load generation is shared between slaves.

2.1 Configuring the NFS Server

Fstress exercises one NFS version 3 volume. The exported NFS volume should start empty; Fstress will create all of its files before benchmarking the server. If Fstress will run as an unprivileged user, the NFS server must be configured to allow non-root mounts (on FreeBSD, this is done via the `-n` option to `mountd`).

The NFS server must run NFS version 3, version 3 of the mount protocol, and each must be registered with the server's RPC portmap service. The NFS server is named in two parts: the server, and the exported volume. For example, `servername:/export/path`.

2.2 Configuring the Fstress Clients

The master must be able to initiate a remote shell to each slave. The `fstress.csh` script uses `rsh`. Other remote shell tools (such as `ssh`) may be substituted.

Remote shells must have the `FSTRESS_HOME` environment variable; define it in the `.cshrc` or similar file. Slave clients must also be able to access and write to the `FSTRESS_HOME` directory.

Each slave client must have permission to mount and write to the NFS server.

3 Running the Benchmark

Run the `fstress.csh` script on the master client. For readability, these examples assume the NFS server and Fstress load generating clients are encapsulated in the following environment variables:

```
prompt> setenv SERVER "servername:/export/path"
prompt> setenv CLIENTS "client1name client2name ... clientNname"
```

Unless configured to do otherwise, Fstress will emulate the SPECsfs97 benchmark testing a UDP mount with five minute warmup and run times. Load starts at 250 NFS operations per second and augments by an additional 250 operations per second between test samples. This increase repeats until 10,000 operations per second or until the average latency exceeds 20 milliseconds per operation. This default behavior may be achieved via:

```
prompt> fstress.csh -clients $CLIENTS -server $SERVER
```

This will create a `$FSTRESS_HOME/output` directory containing log files for each load generating client. (The output directory name may be changed with the `-runname` option.)

3.1 Benchmark Options

Table 1 summarizes the `fstress.csh` command line options.

3.1.1 Configuration Options

Configuration options specify the machines used and NFS properties.

- clients C1 ... Cn:** One or more load generating client machines by name (or IP address).
- server S:/path:** The NFS server with exported volume.
- transp X:** NFS transport protocol, either `udp` or `tcp`. You may wish to experiment with different socket buffer sizes, etc, for better TCP performance.
- rexmitmax N:** Retransmit a request at most `N` times before giving up. Such aged requests are reported as “cancel” in the client logs. Set to zero to disable retransmission.
- rexmitage N:** Retransmit a request if no response is received within `N` milliseconds. Retransmitted requests are ignored for timing purposes. Retransmission counts are reported as “rexmit” in the client logs.
- quitfile X:** If specified, create this file when the benchmark terminates due to an error. This signals other load generating clients to quit early, and may also act as a signal to external monitors.
- ssh:** If specified, use `ssh` for remote shells instead of `rsh`.
- root:** If specified, invoke remote shells with the “-l root” flag to run Fstress as root on the load generating clients.

3.1.2 Intensity Options

Fstress starts by exercising the NFS server at a particular load level, iterating to higher and higher loads until either reaching the desired maximum load, or exceeding the maximum allowed average latency. The intensity options affect these parameters.

- low N:** The starting load level, in operations per second.
- high N:** The ending load level.
- incr N:** Load level increase between benchmarking tests.

option	default	description
<i>configuration options:</i>		
-clients C1 ... Cn	—	set of client machines
-server S:/path	—	NFS server host and export path
-transp X	udp	transport protocol
-rexitmax N	2	retransmit request at most N times
-rexitage N	2000 ms	retransmit request after N msec
-quitfile X	—	create this file after an error
-ssh	—	use ssh instead of rsh
-root	—	use root instead of current user
<i>intensity options:</i>		
-low N	250 ops/s	initial load
-high N	1000 ops/s	ending load
-incr N	250 ops/s	incremental load between tests
-warmup N	300 s	pre-run warmup time
-run N	300 s	benchmarking phase run time
-cooldown N	10 s	post-run cooldown time
-fixfileset N	—	fix size, do not increase with load
-maxlat N	20 ms/op	stop if latency exceeds N msec/op
-maxios N	16 ops	per-file prefetch/write-behind bound
-maxinuse N	8192 files	limit on active files
<i>log data options:</i>		
-rusage	off	report detailed resource utilization
-runname X	—	output directory suffix
<i>pre-defined workload:</i>		
-workload X	—	specify a sample workload
<i>directory tree options:</i>		
-maxdepth N	2	maximum directory tree depth
-fcntdist "v:w ... v:w"	SPEC-like	files-per-dir distribution
-fpopdist "v:w ... v:w"	SPEC-like	file popularity distribution
-fsizedist "v:w ... v:w"	SPEC-like	file size distribution
-dcntdist "v:w ... v:w"	SPEC-like	dirs-per-dir distribution
-dpopdist "v:w ... v:w"	SPEC-like	dir popularity distribution
-lcntdist "v:w ... v:w"	SPEC-like	symlinks-per-dir distribution
-lpopdist "v:w ... v:w"	SPEC-like	symlink popularity distribution
<i>runtime workload options:</i>		
-opdist "v:w ... v:w"	SPEC-like	operation distribution
-rsizedist "v:w ... v:w"	SPEC-like	read size distribution
-wsizedist "v:w ... v:w"	SPEC-like	write size distribution

Table 1: Fstress command line options.

-warmup N: Exercise the NFS server at the desired load level for N seconds before starting measurements. Set to zero to disable warmup. Because Fstress uses remote shells to start runs, we recommend at least a few seconds of warmup time to account for variances in client start times so that non-overlapping measurement phases are faithful to the desired load level.

-run N: Benchmark run time, in seconds, where Fstress measures per-operation latencies.

-cooldown N: After the run time has elapsed, continue to exercise the server at the desired load level for N seconds. Set to zero to disable cooldown. Because Fstress uses remote shells, we recommend at least a few seconds of cooldown time to account for variances in client start times so that non-overlapping measurement phases are faithful to the desired load level.

-fixfileset N: Normally Fstress increases the overall file set size with increasing load levels. If this option is set, the file set size is fixed to what would be created for the listed level. The actual load levels follow the normal increasing trend, but the file set does not change. Note: the file set may change during benchmarking if the operation mix includes create or remove operations.

-maxlat N: Terminate the benchmark if the average operation latency exceeds this value.

-maxios N: Limit the number of per-file outstanding operations when performing I/O series. This applies to all I/O, because multiple operations may be necessary if the I/O size exceeds block size (8 KB).

-maxinuse N: Limit the number of active files with outstanding I/O operations. If Fstress attempts to generate a new I/O and there are already too many active files with the maximum number of outstanding I/O operations (maxios, above), then the new request is dropped, reducing the generated request load.

3.1.3 Log Data Options

Each Fstress load generating client generates a log file with detailed per-operation latencies and error counters. The log data options affect where logs are written, and allow additional information to be recorded.

-rusage: If specified, this flag causes Fstress to report detailed resource utilization results in the client log files. This is useful to verify that the benchmark is not blocking unintentionally (given by the page fault and involuntary context switch results).

-runname X: By default, Fstress writes per-client logs in `FSTRESS_HOME/output`. This option lets the user give an output directory name suffix.

3.1.4 Workload and Directory Tree Options

These options affect the runtime workload and file set properties.

-workload X: Specify a workload directory, or name one of the subdirectories in `FSTRESS_HOME/sample_workloads`.

-???dist: Override specific distributions. Distributions may be specified as a series of value:weight pairs (in quotes so they are passed as a single argument), or as the name of a file containing one value, weight pair (with a space between them) per line. For convenience, the workload option applies the proper distributions from the included sample workloads.

3.2 Benchmarking and the File Set

Fstress creates an initial file set and augments it with increasing load levels. The directory tree and file sizes are governed by workload distributions and may vary between runs. In our experience such variance does not significantly affect the results.

Another source of variation stems from the benchmarking phase. If the tested workload includes created and remove operations, it is possible that the file set will change during runs. This may have some effect on results (e.g., longer runtimes shrink the file set more). Furthermore, at high load levels or long run times, it is possible to remove the entire file set, or exceed the limits on maximal file set size.

The client log files report the total number of files, directories, and symlinks before and after each benchmarking phase. Check the client log files to see if the overall set sizes are changing significantly. If so, it may be necessary to restart the benchmark from scratch at higher load levels in order to preserve the desired file set properties.

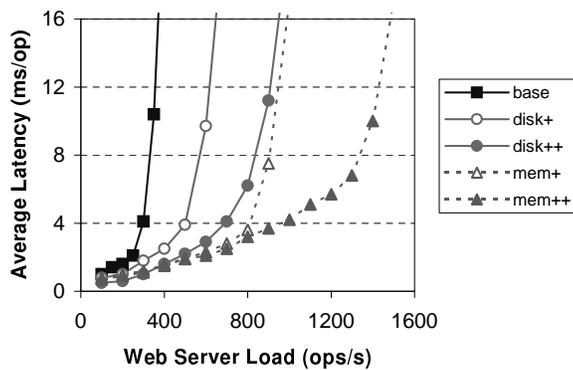


Figure 2: Fstress Web server workload.

4 Analyzing the Results

The `fstress_multistich` tool extracts average latencies from output directories. Run it naming one or more output directories to generate a textual and graphical output plot.

```
prompt> fstress_multistich.csh output1 ... outputN
```

The `fstress_multistich` tool uses `gnuplot` to generate an immediate graph. Figure 2 shows a similar graph created with Microsoft Excel.

NOTE: The `fstress_multistich` textual output assumes all data has the same start and incremental load. If they differ, rows will not reflect reality. Despite this, the graphical output is correct. For correct textual output, run on each output directory separately and manually combine the results. Alternatively, replace `paste` with `join` (take care to account for all load values). This feature may appear in a later version.

4.1 Output Log Files

At the end of each benchmarking phase Fstress spits out three results:

4.1.1 Per-Operation Counts

good: the number of successful (NFS_OK) responses.

error: the number of error (NFSERR_*) responses.

rexmit: the number of retransmissions.

cancel: the number of requests aborted after too many retransmissions. This value also reflects requests aborted because Fstress recycled their entry when generating extreme load levels.

avg: average latency (ms/op).

stddev: latency standard deviation.

contrib: percentage contribution to total average latency.

name: operation name.

4.1.2 Latency Histograms

For each operation, the operation name followed by a series of A:B pairs, where A is a latency in milliseconds, and B is the count of (successful) operations finishing in that much time. I keep track of these in a multi-resolution table. That is, in tenth-millisecond resolution from 0 to 10 ms, then single-millisecond resolution from 10 to 100 ms, then ten-millisecond resolution out to 1000 ms. There is also a one entry to count operations completing in more than 1000 ms.

4.1.3 Summary Line

Combined average for this load level. Numbers are:

wanted: the desired number of nfs operations per second.

called: the number of nfs operations per second issued.

got: the number of nfs responses per second.

avg: combined average latency across all operation types.

nclients: the number of clients used (each at this level).

5 Troubleshooting

If the benchmark does not run correctly, examine the client logs in `$FSTRESS_HOME/output` for clues. Also, `Fstress` includes several test programs to verify critical features.

For each test program, we present a short description followed by successful output (in some cases abbreviated for clarity). In some cases we discuss common problems and demonstrate test program error result(s).

Run the tests from a client. In these examples we use `server.cs.duke.edu` as our NFS version 3 server, exporting the `/export` directory.

5.1 Mount-test

The `mount-test` program attempts to retrieve a mount-point file handle from an NFS version 3 server. This program identifies most common errors (the errors reported here may also show up for the remaining programs).

```
prompt> mount-test server.cs.duke.edu /export
server.cs.duke.edu:/export (fhlen=32) 0x635cffffffe1ffff...more...
mount-test succeeded
```

If the server does not exist, DNS will fail:

```
prompt> mount-test does-not-exist.cs.duke.edu /export
dns_name2addr gethostbyname("does-not-exist.cs.duke.edu"): Unknown host
<terminating after fatal error>
```

If the “server” is not running an NFS version 3 server, the portmap request for the the NFS port will fail (for portmap, 0 indicates failure):

```
prompt> mount-test not-a-server.cs.duke.edu /export
portmap_getport(prog=100005, ver=3): service not found
<terminating after fatal error>
```

If the server is not exporting the requested volume, the test program should return `MNT3ERR_ACCES` (errno 13):

```
prompt> mount-test server.cs.duke.edu /not-exported
mount_getroot: access error
<terminating after fatal error>
```

If the client lacks permission to mount the volume, the server should return `AUTH_TOOWEAK` (auth error 5):

```
prompt> mount-test server.cs.duke.edu /restricted-export
rpchr_parse_reply auth error 5
<terminating after fatal error>
```

5.2 Readdir-test

The `readdir-test` program mounts the given NFS directory and attempts an `NFSPROC_READDIR` of that directory.

```
prompt> readdir-test server.cs.duke.edu /export
readdir-test server.cs.duke.edu /export
0x00000002-00000000 "."
0x00000002-00000000 ".."
0x00000003-00000000 ".tags"
0x00000004-00000000 "quota.user"
...more...
readdir-test succeeded
```

5.3 Operation-test

To send a stream of 100 NFSPROC_NULL requests per second for 10 seconds using the Fstress metronome:

```
prompt> operation-test server.cs.duke.edu 100 10
calls=1000 replies=1000 (expect=1000)
operation-test succeeded
```

If the requested rate saturates something (client, network, or server), the overall reply count may be less than expected. (*calls* reflects the number of requests generated, *replies* is the number of responses received, and *expect* is the number of requests asked for by the command line arguments.)

Saturating the network or server (*calls* equal *expect*, but *replies* less than *calls*):

```
prompt> operation-test server.cs.duke.edu 4000 10
calls=40065 replies=28055 (expect=40000)
operation-test succeeded
```

Saturating the client (*calls* less than *expect*):

```
prompt> operation-test server.cs.duke.edu 6000 10
calls=42142 replies=15641 (expect=60000)
operation-test succeeded
```

5.4 Createtree-test

To create a directory tree:

```
prompt> createtree-test server.cs.duke.edu /export
THIS SECTION IS UNDER CONSTRUCTION
```

If the client does not have permission to create a directory, the server will return an NFSERR_PERM error:

```
prompt> createtree-test server.cs.duke.edu /export
cr_mkdir error 1: NFSERR_PERM
<terminating after fatal error>
```

5.5 Other Fstress Test Programs

Fstress includes several other test programs, but none make significant contributions beyond those test programs listed above. These remaining test programs are documented by their source code.

5.6 External Tests

One may employ various system utilities to test that the cluster is configured properly. Here are some suggestions with their associated FreeBSD tools:

- Verify the client can reach the server:
`ping server.cs.duke.edu`
- Verify portmap is enabled and the NFS service is registered with it:
`rpcinfo -p server.cs.duke.edu`
- Mount the server on the client manually:
`mount_nfs server.cs.duke.edu:/export /mnt ; umount /mnt`
- Verify the exported directory is writable by the client:
`mount_nfs server.cs.duke.edu:/export /mnt ; touch /mnt/foo ; umount /mnt`

workload	file popularities	file sizes	dir sizes	I/O accesses
SPECsfs97	random 10%	1 KB – 1 MB	large (thousands)	random r/w
Web server	Zipf ($0.6 < \alpha < 0.9$)	long-tail (avg 10.5 KB)	small (dozens)	sequential reads
Web proxy	Zipf ($0.6 < \alpha < 0.9$)	long-tail (avg 10.5 KB)	small (dozens)	sequential r/w
database	few files	large (GB - TB)	small	random r/w
peer-peer media server	Zipf (large α)	large (avg 3.7 MB)	large (100 – 1000)	sequential r/w
mail server	Zipf ($\alpha = 1.3$)	fat-tail (avg 4.7 KB)	large (500+)	seq r, append w
news server	Zipf (small α)	bimodal (5 or 45 KB)	large (100+)	random r, append w

Table 2: Some workload types.

6 Sample Workloads

Table 2 summarizes the included sample workloads. Detailed descriptions appear in the general Fstress paper.