

# Part 2, NFS monitoring and tuning

Article Number: 216 | Rating: Unrated | Last Updated: Mon, Jun 4, 2018 12:18 PM

Optimizing AIX 7 network performance

# Part 2, NFS monitoring and tuning

Ken Milberg and Martin Brown

Published on November 30, 2010

[Facebook](#)[Twitter](#)[Linked In](#)[Google+](#)[E-mail this page](#)



## Content series:

# This content is part of the series: **Optimizing AIX 7 network performance**

Often performance tuning is the last thing on your mind when you consider Network File Systems (NFS). You're so preoccupied with configuring your file systems properly and ensuring the security of your systems that you might routinely forget that the N in NFS stands for *network*. Unfortunately, if you don't pay attention to NFS tuning, you might have a poorly performing NFS. Fortunately, IBM has integrated several tools to help monitor and tune NFS on AIX, including `nfsstat` and `nfsmo`. This article describes that monitoring and tuning.

As in previous versions of AIX, AIX 7 supports NFS v2, v3, and v4 (with NFS v4 as the default). In general, NFS v4 should be used in preference to previous versions, since it is more efficient and generally handles the performance of large loads and busy network environments. Most modern client operating systems support NFS v4 so there should be no need to support the older versions.

You'll have to tune both the client and the server. This article explains how you can use `netpmon` to

monitor read subroutines and write subroutines for NFS clients, as well as NFS servers. You can also use `nmon` for a quick snapshot of NFS activity, and you can describe how to use `nmon` to analyze historically based data. Using `netstat`, you can validate that your network is healthy, because poor network utilization (or a poorly designed configuration) can result in poor NFS performance. This article also examines utilities such as `nfs4cl` that can be used for a specific version of NFS. And you'll learn some best practices, including spreading I/O across as many spindles as possible. In this case, you want the CPU load to be the bottleneck, not your I/O system.

Unlike other tuning examples in this series, for NFS you must monitor (and possibly tune) all subsystems including CPU, memory, I/O, and the network. From a client perspective, NFS file systems use disks that are remotely attached. Anything that affects the performance of that mounted disk will affect the performance of the NFS clients. The article also discusses important daemons, such as `nfsd` and `biod`, and how they actually tune themselves. You can see the basic interactions between the client and server to help you understand what's going on behind the scenes. Finally, the article emphasizes that, regardless of which subsystem you are tuning, systems tuning is an on-going process. Of course, the best time to start monitoring your systems is from the beginning, before you run into problems and users scream. Because there are so many factors that can influence NFS performance, make changes only one at a time so that you can accurately assess the impact of your change.

## Reviewing NFS

This section offers an overview of the NFS as it relates to AIX 7. You can learn how the client and server relate to one another and the various factors that influence NFS performance.

You can choose the version type during the actual mounting of the file system, and you can have different versions of NFS running on the server. NFS now supports both TCP and UDP. Because UDP is faster (it does less), some environments that demand optimum performance (on a LAN) over reliability might perform better with UDP. TCP is more reliable (by establishing connections), and it also provides better performance over a WAN, because its flow control helps minimize network latency.

A benefit of NFS is that it acts independently of machine types and operating systems. It does this through its use of Remote Procedure Calls (RPCs), as shown below in Figure 1.

### **Figure 1. Interaction between client and server**

Figure 1 shows how NFS clients A and B access the data off NFS server Z. The client computers first request access to the exported data by mounting the file system. Then, when the client thread tries to process data within the NFS mounted file system, the data is redirected to biod, which takes the data

through the LAN to the NFS server and its nfsd daemon. The server uses nfsd to export the directories that are available to its clients.

As you can tell, you'll need to tune the network and I/O parameters. If Server Z is performing poorly, that obviously affects all of its NFS clients. If possible, tune the server specifically to function as an NFS server (more about that later).

What about the biod daemon? This daemon is required to perform both read-ahead and write-behind requests. The biod daemon improves overall NFS performance as it either empties or fills up the buffer cache, acting as a liaison of the client applications. As shown in [Figure 1](#), the biod daemon sends the requests to the server. On the other hand, nfsd is the liaison that provides NFS services to clients. When the server receives biod communication from the client, the server uses the nfsd daemon until the request is completed.

How is it that NFS was not stateful until Version 4, even though it could use TCP as early as Version 2? Figure 2 shows where NFS resides in relation to the TCP/IP stack and the OSI model.

## **Figure 2. OSI - TCP/IP - NFS**

NFS does not reside on the transport stack, because NFS uses Remote Procedure Calls (RPCs). RPCs are a library of procedures that allow the client and server processes to execute system calls as if they were executed in their own address spaces. In a typical UDP NFS Version 2 or 3 implementation, the NFS

server sends its client a type of cookie after the clients are authorized to share the volume. This helps minimize network traffic. The problem is that if the server goes down, clients will continue to inundate the network with requests. That is why there is a preference for using TCP. Only Version 4 can use stateful connections, and only Version 4 uses TCP as its transport protocol.

NFS Version 4 has no interaction with portmapper or other daemons, such as lockd and statd, because they are rolled into the kernel. In versions other than Version 4, portmapper is used to register RPC services and to provide the port numbers for the communication between clients and servers. External Data Representation (XDR) provides the mechanism that RPC and NFS use to ensure reliable data exchange between the client and the server. It does so in a way that is platform independent for the exchange of binary data. This addresses the possibility of systems representing data in different ways. Using XDR, data can be interpreted correctly, even on platforms that are not alike.

# Monitoring

This section provides an overview of the tools available to you to monitor your NFS systems. These tools enable you to troubleshoot a performance problem quickly and capture data for historical trending and analysis. Some tools are used more often for the server, while other tools are used more for the client. This section covers nmon, topas, nfsstat, nfs, nfs4cl, and netpmon.

## nmon and topas

For NFS tuning, you could use a tool like topas or nmon initially, because they provide a nice dashboard view of what is happening in your system. Remember that NFS performance problems might not be related to your NFS subsystem at all. Your bottleneck might be on the network or, from a server perspective, on your CPU or Disk I/O. Running a tool like topas or nmon can quickly enable you to get a sense of what the real issues are. The example system in this article has two CPUs, and it is running AIX 5.3 TL6.

Figure 3 shows nmon output from an NFS perspective.

### Figure 3. nmon output

Using nmon, there is a lot of information that is available to you from an NFS (client and server) perspective. There are no current bottlenecks at all on this system. Though topas has improved recently with its ability to capture data, nmon might still be a better choice. While nmon provides a front-end

similar to topas, nmon is more useful in terms of long-term trending and analyses. Further, nmon provides the system administrator the capability to output data to a Microsoft® Excel spreadsheet that yields good looking charts (tailored for senior management and functional teams) that clearly illustrate your bottlenecks. This is done through a tool called *nmon analyzer*, which provides the hooks into nmon. The nmon analyzer is available as a free download (see [Related topics](#)).

How can you use nmon to capture data and import it into the analyzer? Using sudo, first run nmon for 3 hours, taking a snapshot every 60 seconds: # sudo nmon -f -t -r test1 -s 60 -c 180. Then sort the output file that gets created: # sort -A systemnfs\_yymmdd.nmon > systemnfs\_yymmdd.csv.

Next, ftp the .csv file to your computer, start the nmon analyzer (enable macros), and click **analyze nmon data**.

## nfsstat

The nfsstat tool is arguably the most important tool you'll use. This command displays all types of information about NFS and RPC calls. nfsstat is used as a monitoring tool to troubleshoot problems and for performance tuning. Depending on the flags you use, you can use nfsstat to display NFS client or server information. It can also show the actual usage count of file system operation. This helps you understand exactly how each file system is utilized, so that you can understand how to best tune your system.

Look at the client flag (c) first. The r flag gives the RPC information (see [Listing 1](#)).

### Listing 1. Running nfsstat with the c flag

```
1 root@lpar24ml162f_pub[/] > nfsstat -cr
2
3 1488pp065_pub[/tmp] > nfsstat -cr
4
5 Client rpc:
6
7 Connection oriented
8
9 calls badcalls badxids timeouts newcreds
10 badverfs timers
```

```

7          279    0    0    0    0    0
          0
8
          nomem  cantconn  interrupts
9
          0    0    0
10
          Connectionless
11
          calls  badcalls  retrans  badxids  timeouts
12
          newcreds  badverfs
13
          24    0    0    0    0    0
          0
          timers  nomem  cantsend
          0    0    0

```

What does this mean? Here are some of the connection-oriented parameters:

- calls—Number of received RPC calls
- badcalls—Number of calls rejected by the RPC layers
- badxids—Number of times a server reply was received that did not correspond to any outstanding call
- timeouts—Number of times calls timed-out while waiting for replies from the server
- newcreds—Number of times authentication information was refreshed
- badverfs—Number of times the call failed due to a bad verifier in the response

If you notice a large number of timeouts or badxids, you could benefit by increasing the timeo parameter with the mount command (details to come).

# nfs

Now look at the nfs information (the n flag) in [Listing 2](#).

## Listing 2. nfs n flag information

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
1488pp065_pub[/tmp] > nfsstat -cn

Client nfs:

calls    badcalls  clgets   cltoomany

279      0         0        0

Version 2: (0 calls)

null     getattr  setattr  root     lookup
readlink read

0 0%     0 0%     0 0%     0 0%     0 0%     0
0%      0 0%

wrcache  write    create   remove   rename
link     symlink

0 0%     0 0%     0 0%     0 0%     0 0%     0
0%      0 0%

mkdir    rmdir    readdir  statfs

0 0%     0 0%     0 0%     0 0%

Version 3: (279 calls)

null     getattr  setattr  lookup   access
readlink read
```

17	0 0%	126 45%	14 5%	9 3%	17 6%	0
	0%	1 0%				
18						
	write	create	mkdir	symlink	mknod	
19	remove	rmdir				
20	51 18%	6 2%	0 0%	0 0%	0 0%	0
	0%	0 0%				
	rename	link	readdir	readdir+	fsstat	
	fsinfo	pathconf				
	0 0%	0 0%	1 0%	1 0%	1 0%	1
	0%	0 0%				
	commit					
	51 18%					

What does this mean? Version 3 parameters include:

- calls—Number of received NFS calls
- badcalls—Number of calls rejected by the NFS layers
- clgets—Number of times a client handle was received
- cltoomany—Number of times the client handle had no unused entries

You should check that the badcalls number is not too high (you can calculate a simple percentage against the calls statistic). Unfortunately, the cltoomany statistic provides evidence of a problem, but there is no way to address the problem.

For the remainder of the statistics, you get a count of the individual operations. This is useful only to determine if you are handling high numbers of reads or writes, in which case you may want to reorganize your disk and NFS sharing allocations.

# nfs4cl

If you're running NFS Version 4, you might be using `nfs4cl` more often. This command displays NFS Version 4 statistics and properties (see [Listing 3](#)).

## Listing 3. Using `nfs4cl`

```
1
2                                     1488pp065_pub[/tmp] > nfs4cl showfs
3
4                                     Server   Remote Path   fsid         Local
5                                     Path
6
7                                     -----
8                                     -----
```

After running this command, you see that there is no output. Run the `mount` command to see more detail (see [Listing 4](#)).

## Listing 4. Running the `mount` command

```
1
2                                     1488pp065_pub[/tmp] > mount
3
4                                     node     mounted   mounted over  vfs
5                                     date     options
6
7                                     -----
8                                     -----
```

```

7          /dev/hd4      /          jfs2 Jul 30 03:16
          rw,log=/dev/hd8
8
9          /dev/hd2      /usr       jfs2 Jul 30 03:16
          rw,log=/dev/hd8
10         /dev/hd9var    /var       jfs2 Jul 30 03:16
          rw,log=/dev/hd8
11
12         /dev/hd3      /tmp       jfs2 Jul 30 03:16
          rw,log=/dev/hd8
13         /dev/hd1      /home      jfs2 Jul 30 03:16
          rw,log=/dev/hd8
14
15         /dev/hd11admin /admin     jfs2 Jul 30
          03:16 rw,log=/dev/hd8
16         /proc         /proc      procfs Jul 30 03:16
          rw
17
18         /dev/hd10opt   /opt       jfs2 Jul 30 03:16
          rw,log=/dev/hd8
19         /dev/livedump  /var/adm/ras/livedump jfs2
          Jul 30 03:16 rw,log=/dev/hd8
20
21         192.168.1.11 /userdata/20009539
          /home/u0009539 nfs3 Jul 30 03:22
          bg,hard,intr,rw
          /dev/fslv00    /wpars/webserver1 jfs2 Aug
          13 09:42 rw,log=INLINE
          /dev/fslv01    /wpars/webserver1/home jfs2
          Aug 13 09:42 rw,log=INLINE
          /opt          /wpars/webserver1/opt namefs

```

```
Aug 13 09:42 ro
                /proc          /wpars/webserver1/proc namefs
Aug 13 09:42 rw
                /dev/fslv02     /wpars/webserver1/tmp jfs2
Aug 13 09:42 rw,log=INLINE
                /usr           /wpars/webserver1/usr namefs
Aug 13 09:42 ro
                /dev/fslv03     /wpars/webserver1/var jfs2
Aug 13 09:42 rw,log=INLINE
```

As you can see, there are no file systems mounted using NFS Version 4. The `nfs4cl` command can also be used to set tunable parameters for NFS v4 filesystems, only NFS Version 3. You do this by using the `setfsoptions` subcommand to tune NFS Version 4.

Some of the key parameters you can configure include:

- `acdirmax` and `acdirmin`—Specifies the upper and lower limit for the directory attribute cache time out value. You can tune these values to set the timeout in seconds that directory entries are retained in the cache after files have been changed. Setting too high a value can lead to file information being stale.
- `acregmax` and `acregmin`—Specifies the upper and lower limit for the file attribute cache time out value. You can tune these values to set the timeout in seconds that file cache information is kept.
- `cio`—Specifies that the filesystem should be mounted with support for concurrent readers and writers.
- `dio`—Specifies that I/O on the filesystem behaves as if all of the files were opened using the direct I/O method, which can provide much faster performance with certain workloads.
- `rbr`—Enables the release-behind-when-reading capability. This frees up NFS cache pages used to hold file data as soon as they have been copied to internal buffers. It can be useful to set this option if you are using NFS as temporary or transient storage for development servers. When the

contents of the files are regularly changing, or you are regularly reading a wide variety of files, caching the file information may not achieve any performance benefits.

[Listing 5](#) below shows you how to do it.

### Listing 5. Tuning timeo with nfs4cl

```
1
root@lpar24ml162f_pub[/] > nfs4cl setfsoptions
mnt rbr=7
```

Remember that you can also use the mount command to tune timeo.

## netpmon

The netpmon command can be used to help troubleshoot NFS bottlenecks. In addition to monitoring many other types of network statistics, netpmon also monitors for clients: both read and write subroutines and NFS RPC requests. For servers, netpmon monitors read and write requests. netpmon starts a trace and runs in the background until you stop it (see [Listing 6](#)).

### Listing 6. Using netpmon

```
1
1488pp065_pub[/tmp] > netpmon -T 2000000 -o
2
/tmp/net.out
3
Run trestop command to signal end of trace.
4
Sun Aug 15 04:58:06 2010
System: AIX 7.1 Node: 1488pp065_pub Machine:
00F604884C00
```

[Listing 7](#) below shows you how to stop the trace.

### Listing 7. Stopping a trace

```
1
2
3
4
5
6
1488pp065_pub[/tmp] > trcstop[netpmon: Reporting
started]
23675650 Missed Entries found
[netpmon: Reporting completed]
[ 4 traced cpus ]
[ 0.091 secs total preempt time ]
```

## The output file

Now look at the NFS-specific information provided for in the output file (see [Listing 8](#)).

### Listing 8. NFS-specific information in the output file

```
1
2
NFSv3 Client RPC Statistics (by Server):
```

```
3 -----
4
5 Server          Calls/s
6 -----
7 p650            126.68
8 -----
9 -----
10 Total (all servers)  126.68
11
12
13 Detailed NFSv3 Client RPC Statistics (by Server):
14 -----
15
16 SERVER: p650
17
18 calls:          5602
19
20 call times (msec):  avg 1.408  min 0.274  max
979.611 sdev 21.310
21
22
23 COMBINED (All Servers)
24
25 calls:          5602
```

```
call times (msec):  avg 1.408  min 0.274  max
                    979.611  sdev 21.310
```

Using netpmon, you can see the NFS Version 3 Client statistics by Server. Note that while netpmon is a useful trace utility, the performance overhead can sometimes outweigh its benefits; particularly when there are other ways to get similar information, so be aware of that when using this utility.

## Tuning with nfsmo and mount

This section describes specific nfs tuning commands. Use nfsmo to set and display your nfsstat tuning parameters. Use mount to tune NFS server-based resources, which take effect only after mounting the NFS file system.

### Client

The biod daemon plays an important role in connectivity. While biod self-tunes the number of threads (the daemon process creates and kills threads as needed), the maximum number of biod threads can be tuned, depending on the overall load. An important concept to understand here is that only increasing the number of threads will not alleviate performance problems caused by CPU, I/O, or memory bottlenecks. For example, if your CPU is near 100% utilization, increasing the amount of threads does not help you at all. Increasing the number of threads can help when multiple application threads access the same files, and you do not find any other types of bottlenecks. Using lsof can help you further determine which threads are accessing which files.

In earlier tuning sections, you might remember the Virtual Memory Manager (VMM) parameters minperm and maxperm. Unlike when you tune database servers, with NFS you want to allow the VMM to use as much RAM as possible for NFS data caching. Most NFS clients have little need for working segment pages. To ensure that all memory is used for file caching, set the maxperm and maxclient both to 100%. In AIX 7 (and AIX 6), both parameters are restricted tunables, which means their limits are

tightly controlled and limited (see [Listing 9](#)).

### Listing 9. Setting maxperm and maxclient to 100 percent

```
1
2
3
4
l488pp065_pub[/tmp] > vmo -o
maxperm%=100Setting maxperm% to 100

Warning: a restricted tunable has been modified

l488pp065_pub[/tmp] > vmo -o
maxclient%=100Setting maxclient% to 100

Warning: a restricted tunable has been modified
```

Note that in the event that your application uses databases, and it could benefit from the application performing its own file data cache, you should not set maxperm and maxclient to 100%. In this instance, set these numbers low, and mount your file systems using Concurrent I/O over NFS. Note also that NFS maintains caches on each client system that contain attributes of the most recently accessed files and directories. The mount command controls the length of time that the entries are kept in cache. The mount parameters you can change are similar to those supported by nfs4cl.

Mount parameters rsize and wsize define the maximum sizes of RPC packets for read and write directories. The default value is 32768 bytes. With NFS Versions 3 and 4, if your NFS volumes are mounted on high-speed networks, you should increase this value to 65536. On the other hand, if your network is extremely slow, you might think about decreasing your default to reduce the amount of packet fragmentation by sending shorter packets. However, if you decrease the default, more packets will need to be sent, which could increase overall network utilization. Understand your network and tune it accordingly!

## Server

Before looking at specific NFS parameters, always try to decrease the load on the network, while also looking at CPU and I/O subsystems. Bottlenecks often contribute to what appears to be an NFS-specific problem. For example, NFS can use either TCP or UDP, depending on the version and your preference. Make sure that your `tcp_sendspace` and `tcp_recvspace` are set to values higher than the defaults, because this can have an impact on your server by increasing network performance. These are not tuned with `nfso`, but with `no` (see [Listing 10](#)).

### Listing 10. Setting `tcp_sendspace` and `tcp_recvspace` to values higher than the defaults

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

If you are running Version 4 of NFS, make sure you turn on `nfs_rfc1323` (see [Listing 11](#)). This allows

for TCP window sizes greater than 64KB. Set this on the client, as well.

### Listing 11. Turning on nfs\_rfc1323

```
1
                                     1488pp065_pub[/tmp] > no -o rfc1323=1Setting
2                                     rfc1323 to 1

                                     Change to tunable rfc1323, will only be effective
                                     for future connections
```

Setting the rfc1323 with nfso sets the TCP window to affect only NFS (as opposed to no, which sets this across the board). If you already set this with no, you don't need to change this, though you might want to, in case some other UNIX® administrator decides to play around with the no commands.

Similar to the client, if the server is a dedicated NFS server, make sure that you tune your VMM parameters accordingly. Modify the maxperm and maxclient parameters to 100% to make sure that the VMM controls the caching of the page files using as much memory as possible in the process. On the server, tune nfsd, which is multithreaded, like you tuned biod (other daemons you can tune include rpc.mountd and rpc.lockd). Like biod, nfsd self-tunes, depending on the load. Increase the number of threads with the nfso command. One such parameter is nfs\_max\_read\_size, which sets the maximum size of RPCs for read replies.

Look at what nfs\_max\_read\_size is set to in [Listing 12](#).

### Listing 12. Setting the nfs\_max\_read\_size parameter

```
1
                                     1488pp065_pub[/tmp] > nfso -L nfs_max_read_size
2
                                     NAME          CUR  DEF  BOOT  MIN
3                                     MAX  UNIT  TYPE
```

```

4                                     DEPENDENCIES
5                                     -----
6                                     -----
nfs_max_read_size      64K  64K  64K  512
64K  Bytes            D
7                                     -----
8                                     -----

```

There are more parameters you can modify. To list all the parameters, use the `-a` or `-L` flag. `-L` provides more information in a nicer format (see [Listing 13](#)).

**Listing 13. Using the `-L` flag with the `nfso` command**

```

1                                     1488pp065_pub[/tmp] > nfso -L
2
3                                     NAME          CUR  DEF  BOOT  MIN
4                                     MAX  UNIT    TYPE
5                                     DEPENDENCIES
6                                     -----
7                                     -----
client_delegation      1   1   1   0   1
8                                     On/Off      D
9                                     -----
10                                    -----

```

10	nfs_max_read_size	64K	64K	64K	512
	64K Bytes				D
11	-----				
12	-----				
13	nfs_max_write_size	64K	64K	64K	512
	64K Bytes				D
14	-----				
15	-----				
16	nfs_rfc1323	1	1	1	0 1
	On/Off				D
17	-----				
18	-----				
19	nfs_securenfs_authtimeout	0	0	0	0
	60 Seconds				D
20	-----				
21	-----				
22	nfs_server_base_priority	0	0	0	31 125
	Pri				D
23	-----				
24	-----				
25	nfs_server_cread	1	1	1	0 1
	On/Off				D
26	-----				
27	-----				
28	nfs_use_reserved_ports	0	0	0	0 1
	On/Off				D

29

-----  
-----

30

nfs\_v3\_server\_readdirplus 1 1 1 0 1  
On/Off D

-----  
-----

nfs\_v4\_fail\_over\_timeout 0 0 0 0  
3600 Seconds D

-----  
-----

portcheck 0 0 0 0 1  
On/Off D

-----  
-----

server\_delegation 1 1 1 0 1  
On/Off D

-----  
-----

utf8\_validation 1 1 1 0 1  
On/Off D

-----  
-----

Now you have a nice list of parameters you can modify!

## Summary

This article discussed the Network File System (NFS), including its history and versions. It defined and discussed the NFS I/O stack and how the stack relates to both the OSI model and the TCP/IP stack. The article discussed best practices for disk configuration and VMM tuning in an NFS environment.

You examined the differences between tuning your clients and servers. You monitored your overall network and drilled down to the NFS layer during the monitoring. Further, you tuned your systems using `nfso` and `mount`.

In the next part of the series, you'll drill down to the actual networking packets. This will include a more detailed discussion of `netstat`. You'll also learn how to tune your network subsystem using the `no` utility.

Posted - Mon, Jun 4, 2018 12:18 PM. This article has been viewed 7594 times.

Online URL: <http://kb.ictbanking.net/article.php?id=216>