A Practical Approach to NFS Response Time Monitoring

Gary L. Schaps and Peter Bishop
Cirrus Logic, Inc.

A Practical Approach to NFS Response Time Monitoring

Gary L. Schaps and Peter Bishop – Cirrus Logic, Inc.

ABSTRACT

NFS and the automounter simplify both filesystem access and filesystem management in large, distributed networks. In demanding environments, however, they require a network management strategy which ensures a consistent and acceptable level of performance.

This paper reviews available tools for measuring NFS response time including rpcspy, nfswatch, nfsstat, and WireTap. We then describe a heuristic technique we have implemented on top of basic NFS response time statistics. It is used to create a management measure of network quality based upon NFS response time, and to proactively focus on NFS performance bottlenecks.

Introduction

In a time-to-market driven environment where everyone's success depends upon networked computing resources, users expect transparent, reliable and fast access to NFS filesystems. While the automounter eases the administrative overhead of providing transparent and reliable access, managing NFS performance requires some additional tools. Why monitor NFS response time? Consider the impact of a change in average NFS response time on the order of only one millisecond per NFS operation. Is it significant? The answer is yes. A busy filesystem can service hundreds of thousands of NFS operations in a short time, and a small change in the response time of each operation quickly adds up. A network which exhibits poor or inconsistent response time wastes users time and ultimately hurts the performance of the organization. Often the cause may be relatively easy to correct, if detected. However, if no systematic effort is made to manage response time, bottlenecks will be overlooked until users complain that "the network is slow".

We have undertaken the task of designing and implementing a continuous improvement program based upon NFS response time. Our initial objective was to produce a management measure of service quality for each of our subnets. We chose NFS response time because it represents the best single measure of overall network performance. Our plan was to establish the measure, track it each month, and use it as a baseline for efforts at continuous improvement. This approach suggests a somewhat general view of NFS response time, but in fact we use a "bottom up" approach designed to enable us to detect the signatures of specific system and network problems and to measure our success at dealing with them. We routinely gather the most detailed data we can collect on all our subnets and analyze it with tools developed in perl, creating

1) a single NFS response time "figure of merit" for each subnet,
2) a list of filesystems which exhibit "poor" performance, and
3) a detailed response time profile for each filesystem partitioned by NFS operation category (read, write, etc.) and performance threshold (excellent, good, etc.).

Collecting NFS Response Time Data

A large network, comprised of many subnets, presents a challenging environment for reliable, automated network monitoring. An often cited design criteria is to avoid a centralized monitoring and data collection point [Lehman92]. We agreed that a distributed system was desirable and began to look at existing tools for collecting NFS response time data. Some earlier work [Keith90, Shein89, Watson92] used nfsstat statistics or "similar kernel meters" to characterize NFS file server performance. With "nfsstat -m" one can track smoothed round trip time for NFS Lookup, Read, and Write operation categories on all mounted filesystems (see Figure 1).

This is close to what we wanted, and an earlier version of our NFS response time monitor used nfsstat. The trouble with nfsstat is that it groups NFS operations into predefined categories and deprives one of the ability to create categories based upon locally observed operation mixes and performance. In addition, getting a network perspective on NFS response time using nfsstat implies a lot of overhead – running it on each client and collecting the results over the network. Lastly, nfsstat presents a slightly confusing picture of what filesystems are actually mounted and what server exports them when the automounter is in use.

Another useful tool for monitoring NFS response time is "nfswatch" [Curry93]. Unlike nfsstat, it is a passive network monitoring tool which observes all NFS traffic on a network and logs useful statistics including the frequency of each type of NFS operation and its average response time. While frequency is reported by filesystem, average response time is not. Passive network monitoring virtues include providing a network view of NFS response time and minimizing the runtime overhead associated with acquiring that view (see Figure 2).

Passive network monitoring is also the approach used by rpcspy [Blaze92] which produces a record for each transaction containing a timestamp, server name, client name, the length of time the NFS procedure took to execute, the name of the procedure, command specific arguments and return data. One of the arguments is an NFS filehandle within which is embedded enough information to determine which of the servers' filesystems is involved in the procedure. Computing the average response time for each NFS procedure by server and filesystem can be accomplished by analyzing the output of rpcspy.

A commercial software tool for monitoring NFS response time is WireTap [AIM92]. It also uses a distributed, passive network monitoring architecture and creates near real-time graphical display of NFS response time as well "expert alarms" which suggest causes and remedies for performance problems. In addition to NFS response time, WireTap also represents NFS load (operations per second), NFS retransmits, a derived metric called "file server efficiency" and Ethernet wire loading over both short and long term time intervals. No filesystem or individual NFS operation level detail is available from WireTap.

We initially used nfswatch to collect NFS response time data from our networks and characterized the local latencies of each NFS operation type. Subsequently, we changed the format of rpcspy's output to include frequency, average response time, and worst case response time for each NFS operation by filesystem. We now run this modified rpcspy during the first thirteen days of each month on all our subnets, averaging response times in twenty minute intervals and collecting the data on a single NFS filesystem for analysis.

Analyzing NFS Response Time Data

NFS defines 18 remote procedure call (RPC) operations, most of which are analogous to Unix system calls. These include operations to read and write a file (read, write), obtain a files attributes (getattr), obtain the contents of directories (lookup,

```
Server (pid104@/news),  (Addr 127.0.0.1)
Flags: hard int  read size=8192, write size=512,  count = 5
Lookups: srtt=2 (5ms), dev=2 (10ms), cur=1 (20ms)
Reads: srtt=0 (0ms), dev=0 (0ms), cur=0 (0ms)
Writes: srtt=0 (0ms), dev=0 (0ms), cur=0 (0ms)
All: srtt=2 (5ms), dev=4432 (22160ms), cur=2216 (44320ms)
```

Figure 1: Output from "nfsstat -m" (srtt = smoothed round trip time)

Procedure	int	pct	total	completed	ave.resp	var.resp	max.resp
CREATE	16	0%	75	16	69.02	5182.62	262.05
GETATTR	925	26%	4029	925	13.45	362.51	189.72
GETROOT	0	0%	0				
LINK	0	0%	0				
LOOKUP	759	21%	3327	759	16.28	718.31	189.65
MKDIR	0	0%	0				
NULLPROC	22	1%	64	22	3.48	7.09	8.87
READ	1263	35%	2789	1262	11.97	466.80	329.74
READDIR	109	3%	225	109	10.17	116.09	61.57
READLINK	21	1%	47	21	8.80	76.02	25.08
REMOVE	0	0%	0				
RENAME	1	0%	7	1	64.86		64.86
RMDIR	0	0%	0				
SETATTR	0	0%	0				
STATFS	11	0%	24	11	2.47	0.87	4.01
SYMLINK	0	0%	0				
WCACHE	0	0%	0				
WRITE	500	14%	1444	500	69.28	1116.28	316.14

Figure 2: Nfswatch data for a twenty minute interval on one subnet

readdir), create files (create), and others. Our strategy in creating an analysis tool was to aggregate those NFS operations which exhibited similar response times, as observed within our environment, into "read", "write", "create" and "lookup" groupings (Figure 3). We then established parameterized thresholds in each grouping for latencies we considered "excellent", "good", "poor" and "bad" (Figure 4). The groupings and their respective response time thresholds undergo continuous refinement as we analyze our data.

	Performance Criteria (in milliseconds)			
	Excellent	Good	Poor	Bad
LOOKUP	<4	<10	<30	>30
READ	<20	<40	<60	>60
CREATE	<35	<70	<100	>100
WRITE	<50	<100	<150	>150

Figure 4: Current thresholds for operation groupings

Our analysis tools have two distinct purposes. The first is to characterize all networks in a very general way for "continuous improvement" purposes. We want to be able to quantify the response time character of each subnet in a way that can be presented (perhaps too) simply to management, and use this "figure of merit" to benchmark, in a very general way, our month to month success at providing an ever better level of service. It is very important that such a figure of merit "improve" whenever there is a real improvement and "degrade" whenever service has deteriorated. We would like a kind of average response time, but we need to identify unacceptable response times whenever they occur. Providing thresholds for each operation grouping helps identify "bad" response times, and helps normalize the response times to allow averages across all operations without having variations in operation mix generate false signals of network quality variation.

Our second objective is to have the ability to diagnose server and/or filesystem level problems at the operation grouping level, apply appropriate remedies, and verify their success. We have have created two perl tools "rpcspy2fom.pl" and "rpcspy2art.pl" to support these objectives.

Figure 5 illustrates the results of our analysis. Filesystems which exhibit an aggregate delay in the "bad" operation category of more than five seconds are placed on a "bad filesystem list", ordered by the size of their aggregate delay. The behavior of "excellent", "good" and "poor" operations is summarized in a single "figure of merit" for the subnet. We characterize the service quality of a subnet both by the "figure of merit" and by the number of filesystems which show up on its list. To focus our attention on the worst problems, we sort the lists of bad filesystems globally across all networks for cumulative time spent in "bad" operations.

Average Response Time for NFS Operations

			(Read)	(Write)	(Create)	(Lookup)	(n/a)
(1)	CREATE	=			16.6		
(2)	GETATTR	=				3.4	
(3)	GETROOT	=					0
(4)	LINK	=			13.9		
(5)	LOOKUP	=				7.0	
(6)	MKDIR	=		102.9			
(7)	NULL	=				2.8	
(8)	READ	=	11.7				
(9)	READDIR	=	11.5				
(10)	READLINK	=				4.5	
(11)	REMOVE	=			30.9		
(12)	RENAME	=			28.1		
(13)	RMDIR	=			40.7		
(14)	SETATTR	=	11.2				
(15)	STATFS	=				3.4	
(16)	SYMLINK	=			13.4		
(17)	WCACHE	=					0
(18)	WRITE	=		60.2			

Figure 3: Aggregate NFS response time data from 12 subnets (4 months)

Results and Conclusion

Our NFS response time analysis framework has enabled us to detect and remedy specific write, read and lookup category problems. For example, installing a write cache on an Auspex server showed a dramatic effect in its response time profile. In other cases we have seen the need to add memory to machines exhibiting slow lookup category response times and to devise new filesystem distributions for machines with slow read category response times. In each case, we are able to use historical data to perform a before-and-after analysis of our success at dealing with these problems. We are also able to present each month a management measure of network quality based upon NFS response time.

Availability

Nfswatch v4.0 and rpcspy can be obtained from Internet archive sites. Our modifications to rpcspy, and the perl programs used in our analysis can be obtained by sending an email request to gls@cirrus.com.

Author Information

Gary L. Schaps is an analyst in the computing resources department at Cirrus Logic. He has an MSCS degree from the University of Miami and enjoys writing software, managing networks and playing raquetball. Reach Gary at: gls@cirrus.com.

Peter Bishop holds a Ph.D. in computer science from MIT, and manages the computing resources department at Cirrus Logic. Reach Peter at: peter@cirrus.com.

References

R. Lehman, G. Carpenter, & H. Hien, "Concurrent Network Management with a Distributed Management Tool", *USENIX LISA VI Conference Proceedings, 1992.*

NFS RESPONSE TIME REPORT NET: 141.131.99 DATES: 8/1/93 - 8/13/93 FIGURE OF MERIT: 0.130 THRESHOLD: 5 seconds				
Server Filesys Op	Excellent	Good	Poor	Bad
ss212 sd2c Write	3079 <50ms 40.5ms 0.6%	41072 <100ms 70.4ms 7.5%	157114<150ms 129.ms 28.7%	346748>150ms 184.ms 63.3%
ss212 sd3c Write	653 <50ms 44.4ms 0.3%	7551 <100ms 82.8ms 3.6%	76833 <150ms 134.ms 36.6%	125067>150ms 174.ms 59.5%
sungraf sd12d Write	7 <50ms 34.9ms 0.0%	7751 <100ms 89.6ms 7.6%	27601 <150ms 108.ms 27.1%	66482 >150ms 186.ms 65.3%
ss301 sd2c Write	71 <50ms 31.8ms 0.2%	1746 <100ms 74.5ms 5.5%	5593 <150ms 147.ms 17.6%	24447 >150ms 177.ms 76.7%
ss212 sd2c Lookup	86885 <10ms 2.7ms 93.5%	3179 <20ms 14.5ms 3.4%	1178 <50ms 23.0ms 1.3%	1688 >50ms 2147ms 1.8%
sungraf sd8a Write	40 <50ms 36.2ms 0.4%	2 <100ms 81.ms 0.0%	706 <150ms 119.ms 7.8%	8310 >150ms 266.ms 91.7%
tango vp30 Write	401503<50ms 31.3ms 50.4%	307670<100ms 68.9ms 38.6%	80988 <150ms 112.ms 10.2%	6664 >150ms 151.ms 0.8%
talisman sd5g Write	19191 <50ms 33.8ms 26.9%	19003 <100ms 86.8ms 26.6%	29137 <150ms 113.ms 40.8%	4027 >150ms 226.ms 5.6%
sunrise sd7g Write	2 <50ms 48.ms 0.1%	80 <100ms 71.1ms 2.4%	3 <150ms 116.ms 0.1%	3201 >150ms 233.ms 97.4%
tango ad0f Write	163 <50ms 47.9ms 4.2%	738 <100ms 61.4ms 19.0%	165 <150ms 145.ms 4.3%	2813 >150ms 264.ms 72.5%
talisman sd6c Create	9136 <40ms 29.4ms 76.7%	2710 <70ms 46.4ms 22.8%	9 <100ms 86.3ms 0.1%	52 >100ms 142.ms 0.4%
talisman id000a Create	1429 <40ms 17.3ms 36.3%	2102 <70ms 57.1ms 53.5%	348 <100ms 77.7ms 8.9%	53 >100ms 129.ms 1.3%
talisman id001g Lookup	6185 <10ms 1.9ms 75.7%	1754 <20ms 15.3ms 21.5%	182 <50ms 30.9ms 2.2%	48 >50ms 114.ms 0.6%

Figure 5: Annotated NFS Repsonse Time Report

Bruce E. Keith, "Perspectives on NFS File Server Performance Characterization", *USENIX Summer Conference Proceedings, 1990*.

Barry Shein, Mike Callahan & Paul Woodbury, "NFSSTONE A Network File Server Performance Benchmark", *USENIX Summer Conference Proceedings, 1989*.

Andy Watson & Bruce Nelson, "LADDIS: A Multi-Vendor and Vendor Neutral SPEC NFS Benchmark", *USENIX LISA VI Conference Proceedings, 1992*.

Matt Blaze, "NFS Tracing By Passive Network Monitoring", *USENIX Winter Conference Proceedings, 1992*.

David A. Curry and Jeffrey C. Mogul, nfsstat man page, 1993.

AIM Technology, *WireTap User Guide (Version 1.1.3)*, 1992.