A Trace-Based Study of SMB Network File System Workloads in an Academic Enterprise

Paul Wortman and John Chandy
Department of Electrical and Computer Engineering
University of Connecticut, USA
(paul.wortman, john.chandy)@uconn.edu
+1-860-486-5047/+1-860-486-2447

***Abstract*—Storage system traces are important for examining real-world applications, studying potential bottlenecks, as well as driving benchmarks in the evaluation of new system designs. While file system traces have been well-studied in earlier work, it has been some time since the last examination of the SMB network file system. The purpose of this work is to continue previous SMB studies to better understand the use of the protocol in a real-world production system in use at the University of Connecticut. The main contribution of our work is the exploration of I/O behavior in modern file system workloads as well as new examinations of the inter-arrival times and run times for I/O events. We further investigate if the recent standard models for traffic remain accurate. Our findings reveal interesting data relating to the number of read and write events. We notice that the number of read and write events is significantly less than creates and the average number of bytes exchanged per I/O is much smaller than what has been seen in previous studies. Furthermore, we find an increase in the use of metadata for overall network communication that can be taken advantage of through the use of smart storage devices.** ***Index terms*— Server Message Block, Storage System Tracing, Network Benchmark, Storage Systems, Distributed I/O.**

I. INTRODUCTION

Over the last twenty years, data storage provisioning has been centralized through the use of network file systems. The architectures of these storage systems can vary from storage area networks (SAN), network attached storage (NAS), clustered file systems, hybrid storage, amongst others. However, the front-end client-facing network file system protocol in most enterprise IT settings tends to be, for the most part, solely SMB (Server Message Block) because of the preponderance of Microsoft (MS) Windows clients. While there are other network file systems such as Network File System (NFS) and clustered file systems such as Ceph, PanFS, and OrangeFS, they tend to be used less extensively in most non-research networks.

In spite of the prevalence of SMB usage within most enterprise networks, there has been very little analysis of SMB workloads in prior academic research. The last major study of SMB was more than a decade ago [1], and the nature of storage usage has changed dramatically over the last decade. It is always important to revisit commonly used protocols to examine their use in comparison to the expected use case(s). This is doubly so for network communications because the nuances of networked data exchange can greatly influence the effectiveness and efficiency of a chosen protocol. Since an SMB-based trace study has not been undertaken recently, we took a look at its current implementation and use in a large university network.

Our study is based on network packet traces collected on the University of Connecticut's centralized storage facility over a period of three weeks in May 2019. This trace-driven analysis can help in the design of future storage products as well as providing data for future performance benchmarks. Benchmarks allow for the stress testing of various aspects of a system (e.g. network, single system). Aggregate data analysis collected from traces can lead to the development of synthetic benchmarks. Traces can also expose systems patterns that can also be reflected in synthetic benchmarks. Finally, the traces themselves can drive system simulations that can be used to evaluate prospective storage architectures.

We created a new tracing system to collect data from the university storage network system. The tracing system was built around the high-speed PF_RING packet capture system [2] and required the use of proper hardware and software to handle incoming data. We also created a new trace capture format based on the DataSeries structured data format developed by HP [3]. PF_RING acts as a kernel module that aids in minimizing packet loss/timestamping issues by not passing packets through the kernel data structures. DataSeries was modified to filter specific SMB protocol fields along with the writing of analysis tools to parse and dissect the captured packets. Specific fields were chosen to be the interesting fields kept for analysis. The DataSeries data format allowed us to create data analysis code that focuses on I/O events and ID tracking: e.g. Tree Identifier (TID) and User Identifier (UID). The future vision for this information is to combine ID tracking with the OpLock information in order to track resource sharing of the different clients on the network, as well as using IP information to recreate communication in a larger network trace to establish a better benchmark.

The contributions of this work are the new traces of SMB traffic over a large university network as well as new analysis of this traffic. Our new examination of the captured data reveals that despite the streamlining of the CIFS/SMB protocol to be less ”chatty”, the majority of SMB communication is still metadata based I/O rather than actual data I/O. We found that read operations occur in greater numbers and cause a larger

SummerSim-SPECTS 2020, July 20-22, Madrid, Spain; ©2020 Society for Modeling and Simulation (SCS) International

overall number of bytes to pass over the network. Additionally, the average number of bytes transferred for each write I/O is smaller than that of the average read operation. We also find that the current standard for modeling network I/O holds for the majority of operations, while a more representative model needs to be developed for reads.

II. RELATED WORK

We summarize major works in trace study in Table I. Tracing collection and analysis from previous studies have provided important insights and lessons such as an observations of read/write event changes, overhead concerns originating in system implementation, bottlenecks in communication, and other revelations found in the traces. Previous tracing work has shown that one of the largest and broadest hurdles to tackle is that traces (and benchmarks) must be tailored to the system being tested. There are always some generalizations taken into account, but these generalizations can also be a major source of error (e.g. timing, accuracy, resource usage) [9], [13], [17], [18], [19], [20], [21], [22], [23]. To produce a benchmark with high fidelity one needs to understand not only the technology being used but how it is being implemented within the system [11], [22], [23]. All these aspects lend to the behavior of the system; from timing and resource elements to how the managing software governs actions [8], [13], [17]. Furthermore, in pursuing this work one may find unexpected results and learn new things through examination [1], [11], [17]. These studies are required in order to evaluate the development of technologies and methodologies along with furthering knowledge of different system aspects and capabilities. As has been pointed out by past work, the design of systems is usually guided by an understanding of the file system workloads and user behavior.

Leung et al. [1] found that over 67% of files were never opened by more than one client and that read-write access patterns are more frequent. Anderson et al. [18] found that a source of decreased precision came from the kernel overhead for providing timestamp resolution. This would introduce substantial errors in the observed system metrics due to the use inaccurate tools when benchmarking I/O systems. These errors in perceived I/O response times can range from +350% to -15%. Issues of inaccuracies in scheduling I/O can result in as much as a factor 3.5 difference in measured response time and factor of 26 in measured queue sizes. These inaccuracies pose too much of an issue to ignore.

Orosz and Skopko [19] examined the effect of the kernel on packet loss and showed that when taking network measurements, the precision of the timestamping of packets is a more important criterion than low clock offset, especially when measuring packet inter-arrival times and round-trip delays at a single point of the network. One solution for network capture is the tool Dumpcap. However, the concern with Dumpcap is that it is a single threaded application and was suspected to be unable to handle new arriving packets due to the small size of the kernel buffer. Work by Dabir and Matrawy [20] attempted to overcome this limitation by using two semaphores to buffer incoming strings and improve the writing of packet information to disk. Skopkó [21] examined the concerns of software-based capture solutions and observed that software solutions relied heavily on OS packet processing mechanisms. Furthermore, depending on the mode of operation (e.g. interrupt or polling), the timestamping of packets would change.

As seen in previous trace work [1], [11], [17], the general perceptions of how computer systems are being used versus their initial purpose have allowed for great strides in eliminating actual bottlenecks rather than spending unnecessary time working on imagined bottlenecks. Without illumination of these underlying actions (e.g. read-write ratios, file death rates, file access rates) these issues cannot be readily tackled.

III. BACKGROUND

The Server Message Block (SMB) is an application-layer network protocol mainly used for providing shared access to files, shared access to printers, shared access to serial ports, miscellaneous communications between nodes on the network, as well as providing an authenticated inter-process communication mechanism. The SMB 1.0 protocol [24] has been found to have high/significant impact on performance due to latency issues. Monitoring revealed a high degree of "chattiness" and disregard of network latency between hosts. Solutions to this problem were included in the updated SMB 2.0 protocol which decreases "chattiness" by reducing commands and subcommands from over a hundred to nineteen [25]. Additional changes, most significantly increased security, were implemented in the SMB 3.0 protocol (previously named SMB 2.2).

The rough order of communication for SMB session file interaction contains five steps. First is a negotiation where a Microsoft SMB Protocol dialect is determined. Next, a session is established to determine the share-level security. After this, the Tree ID (TID) is determined for the share to be connected to as well as a file ID (FID) for a file requested by the client. From this establishment, I/O operations are performed using the FID given in the previous step.

The only data that needs to be tracked from the SMB traces are the UID (User ID) and TID for each session. The SMB commands also include a MID (Multiplex ID) value that is used for tracking individual packets in each established session, and a PID (Process ID) that tracks the process running the command or series of commands on a host. For the purposes of our tracing, we do not track the MID or PID information. Some nuances of the SMB protocol I/O to note are that SMB/SMB2 write requests are the actions that push bytes over the wire while for SMB/SMB2 read operations it is the response packets.

IV. PACKET CAPTURING SYSTEM

A. University Storage System Overview

We collected traces from the University of Connecticut Information Technology Services (ITS) centralized storage server, which consists of five Microsoft file server cluster nodes. These blade servers are used to host SMB file shares for various departments at the university as well as personal

Study	Date of Traces	FS/Protocol	Network FS	Trace Approach	Workload
Ousterhout, *et al.* [4]	1985	BSD		Dynamic	Engineering
Ramakrishnan, *et al.* [5]	1988-89	VAX/VMS	x	Dynamic	Engineering, HPC, Corporate
Baker, *et al.* [6]	1991	Sprite	x	Dynamic	Engineering
Gribble, *et al.* [7]	1991-97	Sprite, NFS, VxFS	x	Both	Engineering, Backup
Douceur and Bolosky [8]	1998	FAT, FAT32, NTFS		Snapshots	Engineering
Vogels [9]	1998	FAT, NTFS		Both	Engineering, HPC
Zhou and Smith [10]	1999	VFAT		Dynamic	PC
Roselli, *et al.* [11]	1997-00	VxFS, NTFS		Dynamic	Engineering, Server
Agrawal, *et al.* [12]	2000-2004	FAT, FAT32, NTFS		Snapshots	Engineering
Ellard, *et al.* [13]	2003	NFS	x	Dynamic	Engineering, Email
Leung, *et al.* [1]	2007	CIFS	x	Dynamic	Corporate, Engineering
Vrable, *et al.* [14]	2009	FUSE	x	Snapshots	Backup
Benson, *et al.* [15]	2010	AFS, MapReduce, NCP, SMB	x	Dynamic	Academic, Corporate
Chen, *et al.* [16]	2012	MapReduce	x	Dynamic	Corporate
This paper	2020	SMB	x	Dynamic	Academic, Engineering, Backup

TABLE I

SUMMARY OF MAJOR FILE SYSTEM STUDIES OVER THE PAST DECADES. FOR EACH STUDY THE TABLES SHOWS THE DATES OF THE TRACE DATA, THE FILE SYSTEM OR PROTOCOL STUDIED, WHETHER IT INVOLVED NETWORK FILE SYSTEMS, THE TRACE METHODOLOGY USED, AND THE WORKLOADS STUDIED. DYNAMIC TRACE STUDIES ARE THOSE THAT INVOLVE TRACES OF LIVE REQUESTS. SNAPSHOT STUDIES INVOLVE SNAPSHOTS OF FILE SYSTEM CONTENTS.

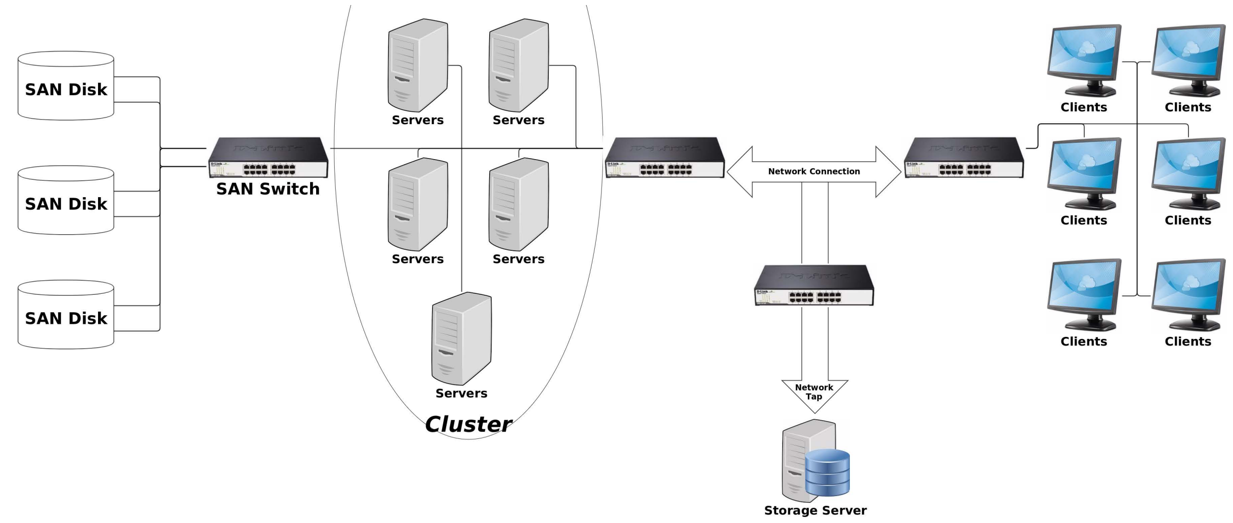

Fig. 1. Visualization of Packet Capturing System

drive share space for faculty, staff and students, along with at least one small group of users. Each server is capable of handling 1 Gb/s of traffic in each direction (e.g. outbound and inbound traffic). Altogether, the five-blade server system can in theory handle 5 Gb/s of data traffic in each direction. The blade servers serve as SMB heads, but the actual storage is served by a pair of NetApp SAN appliance nodes that sit behind the SMB heads. The NetApp storage provides 588 TB of usable disk storage fronted by 4 TB of flash cache. This system does not currently implement load balancing. Instead, the SMB servers are set up to spread the load with a static distribution across four of the active cluster nodes while the passive fifth node takes over in the case of any other nodes going down.

The actual tracing was performed with a tracing server connected to a switch outfitted with a packet duplicating element as shown in the topology diagram in Figure 1. A 10 Gbps network tap was installed in the file server switch, allowing our storage server to obtain a copy of all network traffic going to the 5 file servers. The reason for using 10 Gbps hardware is to help ensure that the system is able to capture information on the network at peak theoretical throughput.

B. High-speed Packet Capture

In order to maximize our faithful capture of the constant rate of traffic, we implement on the tracing server an ntop [26] solution called PF_RING [2] to dramatically improve the storage server's packet capture speed. We had to tune an implementation of `tshark` (wireshark's terminal pcap implementation) to maximize the packet capture rate. `tshark` outputs `.pcap` files which captures all of the data present in packets on the network. We configure `tshark` so that it only captures SMB packets. Furthermore, to optimize this step, a capture ring buffer flag is used to minimize the amount of space used to write `.pcap` files, while optimizing the amount of time to filter data from the `.pcap` files. The file size used was in a ring buffer where each file captured was 64000 kB.

The `.pcap` files from `tshark` do not lend themselves to easy data analysis, so we translate these files into `.ds` files using the DataSeries [3] format, an XML-based structured data format designed to be self-descriptive, storage and access

Total Days	21
Total Sessions	2,413,589
Number of SMB Operations	281,419,686 (100%)
Number of General SMB Operations	210,705,867 (74.87%)
Number of Creates	54,486,043 (19.36%)
Number of Read I/Os	8,355,557 (2.97%)
Number of Write I/Os	7,872,219 (2.80%)
R:W I/O Ratio	1.06
Total Data Read (GB)	0.97
Total Data Written (GB)	0.6
Average Read Size (B)	144
Average Write Size (B)	63

TABLE II
SUMMARY OF TRACE I/O STATISTICS FOR THE TIME OF APRIL 30TH, 2019 TO MAY 20TH, 2019

I/O Operation	SMB	SMB2	Both
General Operations	2,418,980	208,286,887	210,705,867
General %	99.91%	74.66%	74.87%
Create Operations	0	54,486,043	54,486,043
Create %	0.00%	19.53%	19.36%
Read Operations	1,931	8,353,626	8,355,557
Read %	0.08%	2.99%	2.97%
Write Operations	303	7,871,916	7,872,219
Write %	0.01%	2.82%	2.80%
Combine Protocol Operations	2,421,214	278,998,472	281,419,686
Combined Protocols %	0.86%	99.14%	100%

TABLE III
PERCENTAGE OF SMB AND SMB2 PROTOCOL COMMANDS FOR THE TIME OF APRIL 30TH, 2019 TO MAY 20TH, 2019

efficient, and highly flexible. For our purposes, there is no need to track all data that is exchanged, only information that illuminates the behavior of the clients and servers that interact over the network (i.e. I/O transactions). It should also be noted that all sensitive information being captured by the tracing system is hashed to protect the privacy of the users of the storage system. Furthermore, the DataSeries file retains only the first 512 bytes of the SMB packet - enough to capture the SMB header information that contains the I/O information we seek, while the body of the SMB traffic is not retained in order to better ensure privacy. The reasoning for this limit was to allow for capture of longer SMB AndX message chains due to negotiated *MaxBufferSize*. It is worth noting that in the case of larger SMB headers, some information is lost, however this is a trade-off by the university to provide, on average, the correct sized SMB header but does lead to scenarios where some information may be captured incompletely. This scenario only occurs in the cases of large AndX Chains in the SMB protocol, since the SMB header for SMB 2 is fixed at 72 bytes. In those scenarios the AndX messages specify only a single SMB header with the rest of the AndX Chain attached in a series of block pairs.

C. DataSeries Analysis

Building upon existing code for the interpretation and dissection of the captured `.ds` files, we developed C/C++ code to examine the captured traffic information. From this analysis, we are able to capture read, write, create and general I/O information at both a global scale and individual tracking ID (UID/TID) level. In addition, read and write buffer size information is tracked, as well as the inter-arrival and response times. Also included in this data is oplock information and IP addresses. The main contribution of this step is to aggregate observed data for later interpretation of the results. This step also creates an easily digestible output that can be used to re-create all tuple information for SMB/SMB2 sessions that are witnessed over the entire time period. Sessions are any communication where a valid UID and TID is used.

V. DATA ANALYSIS

Table II shows a summary of the SMB traffic captured, statistics of the I/O operations, and read/write data exchange observed for the network filesystem. This information is further detailed in Table III, which illustrates that the majority of I/O operations are general (74.87%). As shown in Table IV, general I/O includes metadata commands such as connect, close, query info, etc.

Our examination of the collected network filesystem data revealed interesting patterns for the current use of CIFS/SMB in a large academic setting. The first is that there is a major shift away from read and write operations towards more metadata-based ones. This matches the last CIFS observations made by Leung et. al. [1] that files were being generated and accessed infrequently. The change in operations are due to a movement of use activity from reading and writing data to simply checking file and directory metadata. However, since the earlier study, SMB has transitioned to the SMB2 protocol which was supposed to be less ”chatty”. As a result, we would expect fewer general SMB operations. Table III shows a breakdown of SMB and SMB2 usage over the time period of May. From this table, one can see that the SMB2 protocol makes up 99.14% of total network operations compared to just 0.86% for SMB, indicating that most clients have upgraded to SMB2. However, 74.66% of SMB2 I/O are still general operations. Contrary to the purpose of implementing the SMB2 protocol, there is still a large amount of general I/O.

Taking a deeper look at the SMB2 operations, shown in Table IV, we see that 9.06% of the general operations are negotiate commands. These are commands sent by the client to notify the server which dialects of the SMB2 protocol the client can understand. The three most common commands are close, tree connect, and query info. The latter two relate to metadata information of shares and files accessed. However, the close operation corresponds to the create operations. Note that the create command is also used as an open file. Notice that the number of closes is greater than the total number of create operations by 9.35%. These extra close operations are most likely due to applications doing multiple closes that do not need to be performed.

A. I/O Data Request Sizes

Each SMB Read and Write command is associated with a data request size that indicates how many bytes are to be read or written as part of that command. Figure 2 shows the probability density function (PDF) of the different sizes of bytes transferred for read and write I/O operations respectively, as well as showing cumulative distribution functions (CDF) for bytes read and bytes written. The most noticeable aspect of

SMB2 General Operation	Occurrences	Percentage of Total
Close	80,114,256	28.71%
Tree Connect	48,414,491	17.35%
Query Info	27,155,528	9.73%
Negotiate	25,276,447	9.06%
Tree Disconnect	9,773,361	3.5%
IOCtl	4,475,494	1.6%
Set Info	4,447,218	1.59%
Query Directory	3,443,491	1.23%
Session Setup	2,041,208	0.73%
Lock	1,389,250	0.5%
Flush	972,790	0.35%
Change Notify	612,850	0.22%
Logoff	143,592	0.05%
Oplock Break	22,397	0.008%
Echo	4,715	0.002%
Cancel	0	0.00%

TABLE IV

BREAKDOWN OF GENERAL OPERATIONS FOR SMB2 FROM APRIL 30TH, 2019 TO MAY 20TH, 2019.

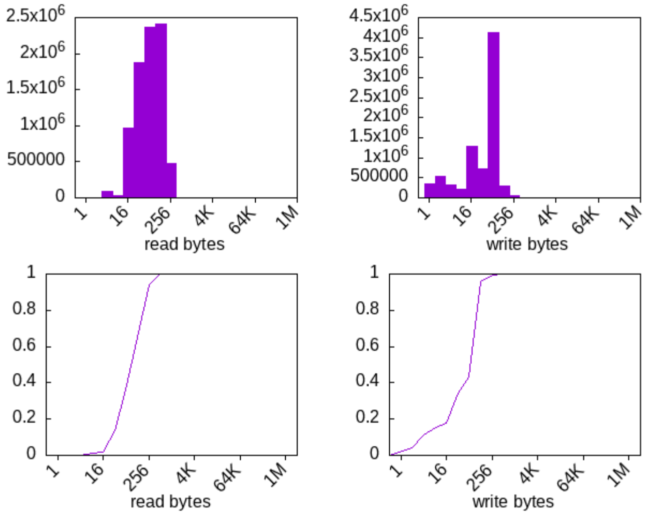

Fig. 2. PDF and CDF of Bytes Transferred for Read and Write I/O

these graphs is that the majority of bytes transferred for read and write operations is around 64 bytes. It is worth noting that write I/Os also have a larger number of very small transfer amounts. This is unexpected in terms of the amount of data passed in a frame. Part of the reason is due to a large number of long-term scripts that only require small but frequent updates, as we observed several running scripts creating a large volume of files. A more significant reason was because we noticed Microsoft Word would perform a large number of small reads at ever growing offsets. This was interpreted as when a user is viewing a document over the network and Word would load the next few lines of text as the user scrolled down the document; causing "loading times" amid use. Finally, a large degree of small writes were observed to be related to application cookies or other such smaller data communications.

Additionally, almost no read transfer sizes are less than 32 bytes, whereas 20% of the writes are smaller than 32 bytes. Table V shows a tabular view of this data. For reads, 34.97% are between 64 and 512 bytes, with another 28.86% at 64-byte request sizes. There are a negligible percentage of read requests larger than 512. This read data differs from the size of reads observed by Leung et al. by a factor of four smaller.

Transfer size	Reads	Writes
< 4	0.098%	11.16%
$= 4$	1.16%	4.13%
$> 4, < 64$	34.89%	28.14%
$= 64$	28.86%	52.41%
$> 64, < 512$	34.97%	4.15%
$= 512$	0.002%	2.54e-5%
$= 1024$	1.22e-5%	3.81e-5%

TABLE V

PERCENTAGE OF TRANSFER SIZES FOR READS AND WRITES

Leung et al. showed that 60-70% of writes were less than 4K in size and 90% less than 64K in size. In our data, however, we see that almost all writes are less than 1K in size. In fact, 11.16% of writes are less than 4 bytes, 52.41% are 64-byte requests, and 43.63% of requests are less than 64 bytes. In the ten years since the last study, it is clear that writes have become significantly smaller. In our analysis of a subset of the writes, we found that a significant part of the write profile was writes to cookies which are necessarily small files. The preponderance of web applications and the associated tracking is a major change in how computers and data storage are used compared to a decade ago. These small data reads and writes significantly alter the assumptions that most network storage systems are designed for.

In comparison of the read, write, and create operations we found that the vast majority of I/O belong to creates. By the fact that there are so many creates, it seems apparent that many applications create new files rather than updating existing files when files are modified. Furthermore, read operations account for the largest aggregate of bytes transferred over the network. However, the number of bytes transferred by write commands is not far behind, although, non-intuitively, including a larger number of standardized relatively smaller writes. The most unexpected finding of the data is that all the read and writes are performed using much smaller buffers than expected; about an order of magnitude smaller (e.g. bytes instead of kilobytes).

B. I/O Response Times

Most previous tracing work have not reported I/O response times or command latency, which is generally proportional to data request size, but under load, the response times give an indication of server load. In Table VI we show a summary of the response times for read, write, create, and general commands. We note that most general (metadata) operations occur fairly frequently, run relatively slowly, and happen at high frequency. We also observe that the number of writes is very close to the number of reads. The write response time for their operations is very small - most likely because the storage server caches the write without actually committing to disk. Reads, on the other hand, are in most cases probably not going to hit in the cache and require an actual read from the storage media. Although read operations are only a small percentage of all operations, they have the highest average response time. As noted above, creates happen more frequently, but have a slightly slower response time, because of the extra metadata operations required for a create as opposed to a simple write.

Figures 3 and 4 shows the inter arrival times CDFs and PDFs. As can be seen, SMB commands happen very fre-

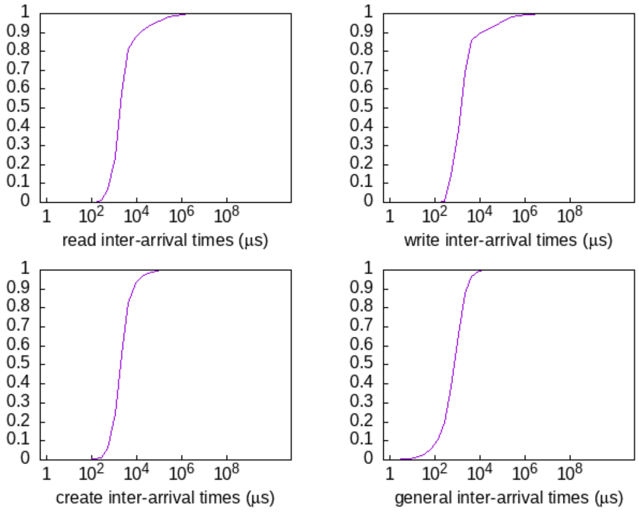

Fig. 3. CDF of Inter-Arrival Time for SMB I/O

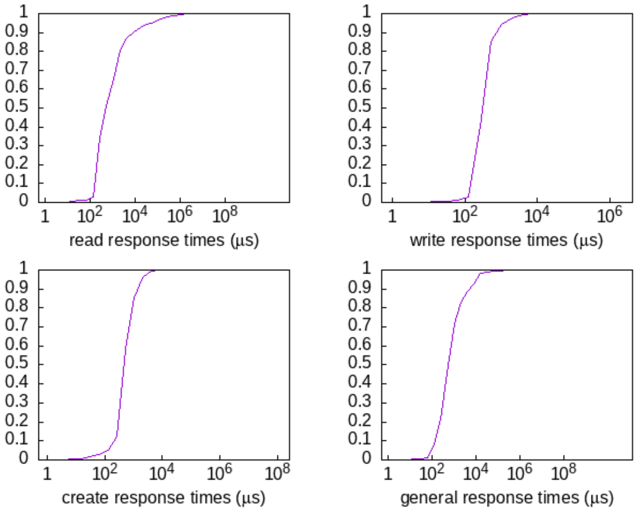

Fig. 5. CDF of Response Time for SMB I/O

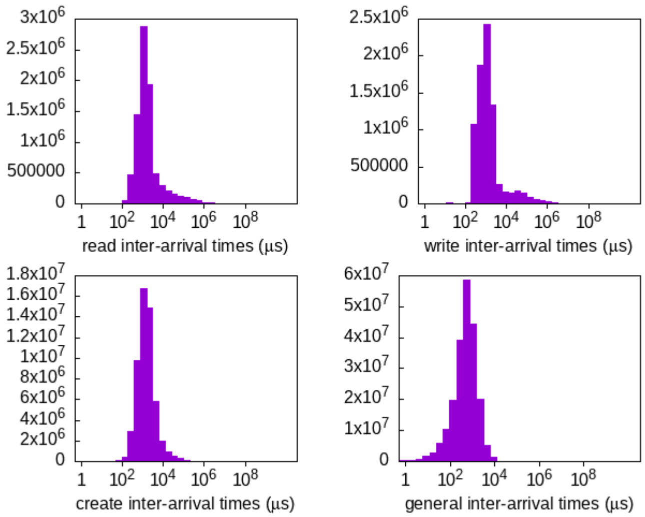

Fig. 4. PDF of Inter-Arrival Time for SMB I/O

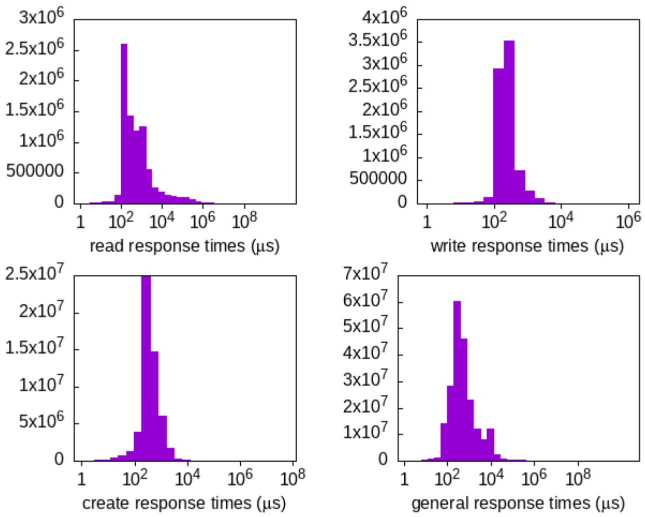

Fig. 6. PDF of Response Time for SMB I/O

quently - 85% of commands are issued less than 1000 μs apart. As mentioned above, SMB is known to be very chatty, and it is clear that servers must spend a significant amount of time dealing with these commands. For the most part, most of these commands are also serviced fairly quickly as seen in Figures 5 and 6. Interestingly, the response time for the general metadata operations follows a similar curve to the inter-arrival times.

The response time for write operations (shown in Figure 5) does not follow the step function similar to the bytes written CDF in Figure 2. This is understandable as the response time for a write would be expected to be a more standardized action and not necessarily proportional to the number of bytes written. However, the read response time is smoother than the bytes read CDF (Figure 2). This is most likely due to the fact that some of the reads are satisfied by server caches, thus eliminating some long access times to persistent storage. However, one should notice that the response time on read operations grows at a rate similar to that of write operations. This, again, shows a form of standardization in the communication patterns although some read I/O take a far greater period of time; due to larger amounts of read data sent over several standardized size packets.

	Reads	Writes	Creates	General
I/O %	2.97	2.80	19.36	74.87
Avg RT (μs)	59,819.7	519.7	698.1	7,013.4
Avg IAT (μs)	33,220.8	35,260.4	5,094.5	1,317.4

TABLE VI
SUMMARY OF TRACE STATISTICS: AVERAGE RESPONSE TIME (RT) AND INTER ARRIVAL TIME (IAT)

C. File Extensions

Tables VII and VIII show a summary of the various file extensions that were seen within the SMB2 traffic during the three-week capture period; following the *smb2.filename* field. The easier to understand is Table VIII, which illustrates the

SMB2 Filename Extension	Occurrences	Percentage of Total
-Travel	33,396,147	15.26
o	28,670,784	13.1
e	28,606,421	13.07
N	27,639,457	12.63
one	27,615,505	12.62
<No Extension>	27,613,845	12.62
d	2,799,799	1.28
l	2,321,338	1.06
x	2,108,279	0.96
h	2,019,714	0.92

TABLE VII

Top 10 File Extensions Seen Over Three Week Period

SMB2 Filename Extension	Occurrences	Percentage of Total
doc	352,958	0.16
docx	291,047	0.13
ppt	46,706	0.02
pptx	38,604	0.02
xls	218,031	0.1
xlsx	180,676	0.08
odt	28	0.000013
pdf	375,601	0.17
xml	1,192,840	0.54
txt	167,827	0.08

TABLE VIII

Common File Extensions Seen Over Three Week Period

number of common file extensions (e.g. doc, ppt, xls, pdf) that were part of the data. Originally, we expected that these common file extensions would be a much larger total of traffic. However, as seen in Table VIII, these common file extensions were less than 2% of total files seen. The top ten extensions that we saw (Table VII) comprised approximately 84% of the total seen. Furthermore, the majority of extensions are not readily identified. Upon closer examination of the tracing system it was determined that many files simply do not have a valid extension. These range from Linux-based library files, manual pages, odd naming schemes as part of scripts or back-up files, as well as date-times and IPs as file names. There are undoubtedly more, but exhaustive determination of all variations is seen as out of scope for this work.

D. Distribution Models

For simulations and analytic modeling, it is often useful to have models that describe storage systems I/O behavior. In this section, we attempt to map traditional probabilistic distributions to the data that we have observed. Specifically, taking the developed CDF graphs, we perform curve fitting to determine the applicability of Gaussian and Weibull distributions to the network filesystem I/O behavior. Note that an exponential distribution, typically used to model interarrival times and response times, is a special case of a Weibull distribution where $k = 1$. Table IX shows best-fit parametrized distributions for the measured data.

As indicated by the error bounds, one can see that the Weibull distributions are generally much better fits for all the different distributions. Furthermore, the Poisson arrival assumption (exponential distribution $k = 1$) is only valid for general I/O, but the write and create service times are exponential. Interestingly, the read and write buffer sizes are very close to exponential, though a Gaussian distribution is a close fit. The models for the write and create operations are similar, while those for read operations are not. Furthermore, there is less similarity between the modeled behavior of general operation inter arrival times and their response times, showing the need for a more refined model for each aspect of the network filesystem interactions.

E. System Limitations and Challenges

When initially designing the tracing system used in this paper, different aspects were taken into account, such as space limitations of the tracing system, packet capture limitations (e.g. file size), and speed limitations of the hardware. One limitation encountered in the packet capture system deals with the functional pcap (packet capture file) size. The concern being that the pcap files only need to be held until they have been filtered for specific protocol information and then compressed using the DataSeries format, but still allow for room for the DataSeries files being created to be stored. Another concern was whether or not the system would be able to function optimally during periods of high network traffic. All aspects of the system, from the hardware to the software, have been altered to help combat these concerns and allow for the most accurate packet capturing possible.

Because the data is being collected from an active network, there will be differing activity depending on the time, the day, the week, and the academic calendar. For example, although the first week or so of the academic year may contain a large amount of traffic, this does not mean that trends of that period of time will occur for every week of the year (except perhaps the final week of the semester). The trends and habits of the network will change based on the time of year, time of day, and even depend on the exam schedule. A comprehensive examination requires looking at all different periods of time to see how all these factors play into the storage system utilization.

VI. Conclusions and Future Work

Our analysis of this university network filesystem illustrated the current implementation and use of the CIFS/SMB protocol in a large academic setting. We notice the effect of caches on the ability of the filesystem to limit the number of accesses to persistent storage. The effect of enterprise storage disks access time can be seen in the response time for read and write I/O. Metadata operations dominate the majority of network communication, which is of less surprise since SMB is a known chatty protocol. We do notice that the CIFS/SMB protocol continues to be chatty with metadata I/O operations regardless of the version of SMB being implemented; 74.66% of I/O being metadata operations for SMB2. We also find that read and write transfer sizes are significantly smaller than would be expected and requires further study as to the impact on current storage systems. Examination of the return times for these different I/O operations shows that exponential distribution curve fitting equation is most accurate at modeling the CDF of the various I/O operations. This shows that the current model is still effective for the majority of I/O, but that for read operations there needs to be further research in modeling their behavior. Our work finds that write and create

Model	Gaussian		Weibull	
CDF	$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\frac{x-\mu}{\sigma}} e^{\frac{-t^2}{2}}\,dt$		$1-e^{(-x/\lambda)^k}$	
I/O Operation	μ	σ	k	λ
General RT	3606.66±20.6%	2.74931e+06±0.02%	0.5652±0.02%	980.9721±0.05%
General IAT	786.72±0.35%	10329.6±0.02%	0.9031±0.02%	743.2075±0.02%
Read RT	44718.5±26.2%	1.72776e+07±0.05%	0.0004±0.00%	1.5517±0.18%
Read IAT	24146±33.4%	1.189e+07±0.05%	0.0005±0.00%	3.8134±0.15%
Write RT	379.823±0.74%	4021.72±0.05%	0.8569±0.05%	325.2856±0.09%
Write IAT	25785.7±33.2%	1.22491e+07±0.05%	0.0004±0.00%	3.1287±0.17%
Create RT	502.084±1.15%	21678.4±0.02%	0.9840±0.02%	496.9497±0.03%
Create IAT	3694.82±33.5%	4.65553e+06±0.02%	0.0008±0.00%	2.3504±0.04%
Read Buff Transfer	82.9179±0.92%	1117.9±0.05%	1.0548±0.03%	85.2525±0.07%
Write Buff Transfer	46.2507±0.97%	640.621±0.05%	1.0325±0.04%	46.8707±0.07%

TABLE IX

COMPARISON OF μ, σ, k, AND λ VALUES FOR CURVE FITTING EQUATIONS ON CDF GRAPHS

response times can be modeled similarly, but that the read response times require the alteration of the general model. However, the general I/O can be modeled using the same standard; which has similar shape and scale to that of the write and create operations.

The analysis work will eventually incorporate oplocks and other aspects of resource sharing on the network to gain a more complete picture of the network's usage and bottlenecks. Network filesystem usage from an individual user scope has become simple and does not contain a greater deal of read, write, and create operations. Further analysis will be made in examining how the determined metrics change when examined at the scope of a per share (i.e. TID) or per user (i.e. UID). At this level of examination, we will be able to obtain a better idea of how each share is interacted with, as well as how files and directories are shared, and access control is implemented. Due to the large number of metadata operations, the use of smart storage solutions could be used to minimize the impact of these I/O. Smart storage elements can aid by performing metadata operations without the need to access persistent storage, thus causing shorter response times. In this manner, the use of smart storage can also help reduce bottlenecks with larger network filesystems and minimize the effect of traffic on overall network performance.

REFERENCES

[1] A. W. Leung, S. Pasupathy, G. R. Goodson, and E. L. Miller, "Measurement and analysis of large-scale network file system workloads," in *USENIX Annual Technical Conference*, 2008, pp. 213–226.

[2] "PF_RING - high-speed packet capture, filtering and analysis." [Online]. Available: http://www.ntop.org/products/packet-capture/pf_ring/

[3] E. Anderson, M. Arlitt, C. B. Morrey III, and A. Veitch, "Dataseries: An efficient, flexible data format for structured serial data," Hewlett-Packard, Technical Report HPL-2009-323, Sept. 2009.

[4] J. K. Ousterhout, H. Da Costa, D. Harrison, J. A. Kunze, M. Kupfer, and J. G. Thompson, "A trace-driven analysis of the UNIX 4.2 BSD file system," *ACM SIGOPS Operating Systems Review*, vol. 19, no. 5, Dec. 1985.

[5] K. Ramakrishnan, P. Biswas, and R. Karedla, "Analysis of file I/O traces in commercial computing environments," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 20, no. 1. ACM, 1992, pp. 78–90.

[6] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurements of a distributed file system," *ACM SIGOPS Operating Systems Review*, vol. 25, no. 5, pp. 198–212, 1991.

[7] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, and E. L. Miller, "Self-similarity in file systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 141–150, 1998.

[8] J. R. Douceur and W. J. Bolosky, "A large-scale study of file-system contents," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 1, pp. 59–70, 1999.

[9] W. Vogels, "File system usage in Windows NT 4.0," in *ACM SIGOPS Operating Systems Review*, vol. 33, no. 5. ACM, 1999, pp. 93–109.

[10] M. Zhou and A. J. Smith, "Analysis of personal computer workloads," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999. Proceedings. 7th International Symposium on*. IEEE, 1999, pp. 208–217.

[11] D. S. Roselli, J. R. Lorch, and T. E. Anderson, "A comparison of file system workloads." in *USENIX Annual Technical Conference*, 2000, pp. 41–54.

[12] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, "A five-year study of file-system metadata," *ACM Transactions on Storage (TOS)*, vol. 3, no. 3, p. 9, 2007.

[13] D. Ellard, J. Ledlie, P. Malkani, and M. Seltzer, "Passive NFS tracing of email and research workloads," *Proceedings of the USENIX Conference on File and Storage Technologies*, pp. 203–216, March 2003.

[14] M. Vrable, S. Savage, and G. M. Voelker, "Cumulus: Filesystem backup to the cloud," *ACM Transactions on Storage (TOS)*, vol. 5, no. 4, p. 14, 2009.

[15] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 267–280.

[16] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.

[17] M. Seltzer and D. Ellard, "NFS tricks and benchmarking traps," in *Proceedings of the FREENIX track, USENIX Annual Technical Conference*. USENIX Association, 2003.

[18] E. Anderson, M. Kallahalla, M. Uysal, and R. Swaminathan, "Buttress: A toolkit for flexible and high fidelity I/O benchmarking," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*. USENIX Association, 2004, pp. 4–4.

[19] P. Orosz and T. Skopko, "Multi-threaded packet timestamping for end-to-end QoS evaluation," in *ICSNC 2013, The Eighth International Conference on Systems and Networks Communications*, 2013.

[20] A. Dabir and A. Matrawy, "Bottleneck analysis of traffic monitoring using Wireshark," in *Innovations in Information Technology, 2007. IIT'07. 4th International Conference on*. IEEE, 2007, pp. 158–162.

[21] T. Skopkó, "Loss analysis of the software-based packet capturing," *Carpathian Journal of Electronic and Computer Engineering*, vol. 5, p. 107, 2012.

[22] A. Traeger, E. Zadok, N. Joukov, and C. P. Wright, "A nine year study of file system and storage benchmarking," *ACM Transactions on Storage (TOS)*, vol. 4, no. 2, p. 5, 2008.

[23] C. Ruemmler and J. Wilkes, *UNIX disk access patterns*. Hewlett-Packard Laboratories, 1992.

[24] Microsoft Corporation, "[MS-SMB]: Server message block (SMB) protocol," Sep. 2018. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/f210069c-7086-4dc2-885e-861d837df688

[25] ——, "[MS-SMB2]: Server message block (SMB) protocol versions 2 and 3," Sep. 2019. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962

[26] "ntop - high performance network monitoring solutions." [Online]. Available: http://www.ntop.org/