Trace Based Analysis of File System Effects on Disk I/O

Sumit Narayan, John A. Chandy
Department of Electrical & Computer Engineering, University of Connecticut
371 Fairfield Road Unit 2157, Storrs, CT 06269 – 2157
{sumit, chandy}@engr.uconn.edu
tel +1-860-486-5047, fax +1-860-486-2447

Keywords: I/O traces, file systems.

ABSTRACT

In this paper, we present a study of low-level disk access patterns for three workloads for a variety of file systems. While there has been a great deal of work on disk architectures, there has been very little work measuring actual low-level disk access. We have extended previous work to also consider different file systems to examine their effect on the workload. The analysis is done based on traces of disk access that have been captured at the driver level just before the data is transferred to disk. The workloads we examined were an email server, file server emulation and a database server and each workload was regenerated for different file systems. We found that depending on the workload, the access patterns could be read dominated or write dominated. The frequency of access in some cases was very high – with inter-arrival times well under 5ms. We also examined the burstiness behavior and read/write distribution. We found that the choice of file system was for the most part insignificant but in some cases can greatly influence the access characteristics. We believe the resulting traces and summary analysis can be used by storage system designers in their analysis of file system and disk organization designs. The consideration of file system also allows storage system designers to consider the style of file system when making design choices.

INTRODUCTION

As processor speed increases at Moore's law rates and disk access time barely change, the gap between processor speed and disk access time becomes more and more critical. Rapid increase in the speed of microprocessors and the relative slowness of disk access times have caused file servers to fail to deliver the speed they are designed for. These access times could be reduced by a variety of techniques including using larger caches, better file system utilization, and varying disk organization techniques. However, without an understanding of the actual disk usage patterns, it is somewhat difficult to make these decisions. Traces of the disk subsystem can provide the maximum information on how the data is treated by a particular file system. To improve the disk access times, one should be familiar with how the data is delivered to the storage system by the file system. There traces are important for many purposes other than the algorithms. These traces are difficult to obtain, and not much work has been published in this area. Much of the related work has collected traces at the operating system level, which does not clearly indicate when the actual disk was accessed for a particular request, since file system behaviors can mask much of the activity at disk. In other words, these traces do not provide the level of disk access detail needed to adequately understand the disk access behavior. These operating system traces are not useful when trying to improve disk storage system systems design for file system performance related to disk.

Our work intends to provide an updated analysis of I/O system behavior based on a variety of workloads, both real and simulated. We also examine whether the choice of file system has an effect on low-level I/O access. We have introduced a small thread in the driver code of the Linux kernel in order to capture the atomic details viz. the time of access, request size, block access number etc.

This paper is organized as follows; Section 2 discusses work done previously by other researchers. Section 3 defines our experiment, and the methods used, while Section 4 details out the system configuration used as clients and servers. Section 5 discusses the results, with analysis presented for all the workloads, and for different file systems. Section 6 concludes the paper with our observations and inferences.

RELATED WORK

There has been significant work done on the analysis of I/O system performance using traces collected from file systems [4,6,7,8,9,11]. One of the earliest studies was the BSD study and its follow up Sprite study [8,4]. The BSD paper provided much of the foundation for latter file system research. Roselli et al reexamined the conclusions from the BSD study with a set of traces collected from several HP/UX servers and a collection of Windows NT clients [9]. There have also been several studies of distributed file system access traces [5,13]. The procedure used in these and similar studies was to capture the trace at a high level – either at the file system level or at the network level. These traces do not illustrate when the access was actually made to the disk, if at all. The UNIX buffer system stores the data temporarily in cache. The data hence could have been hived away in cache, and later deleted from cache memory itself – thus a request for that particular file need not have been made to the disk.

The most relevant work at the disk level was that done by Ruemmler and Wilkes from HP Laboratories in 1993 [1,2]. Their work generated traces from the disk level access collected on HP-UX system with the BSD Fast File

System mounted. Their results indicated that a small non-volatile cache at each disk allowed writes to be serviced much faster that any regular disk. Similar work was done by researchers at IBM where low-level traces were collected on a wide variety of systems including Windows NT PCs and IBM AIX and HP-UX servers [10]. As in the HP study, they found a high degree of burstiness in the access pattern, as well as a write-dominated workload. Idleness in the storage system also suggested opportunities for background optimizations. While the IBM study did examine different file systems, it was difficult to determine the effect of the file system on I/O accesses since the workloads were changed as well. In our work, we intend to further examine the effect of the file system by tracing the same workload on different file systems.

TRACE COLLECTION

We sought to extend previous work by examining the effect of the file system on I/O access patterns. Since our system under study was a Linux server, we traced three popular file systems, namely EXT2, EXT3 and JFS. EXT2 is the most commonly used file system in client Linux systems [14]. EXT3 and JFS are both journaling file systems in that fast restart is enabled through file system metadata logging techniques [15,16]. We examined these file systems under three different workloads – an actual NFS email server, a synthetic CIFS file server workload and a database workload.

To capture the disk I/Os at the lowest level, we introduced a thread into the IDE driver of Linux kernel 2.6.0 (kernel 2.4.20 for NFS email server tests). The code was designed in such a way, that only one particular disk could be tested. The differentiation was done based on the drive numbers of the disk. The thread was transparent to user, and did not add much burden to the system's performance. This thread would record all the requests sent to the disk, along with many other details. Each record collected the following details:

1. Action (Read/Write)
2. Time of Request
3. Sector Number to which the request was made
4. Request Size

The collected trace was written at regular intervals to a completely different disk, to avoid this write as being recorded as another request on the test disk. The average size of each of such write was approximately 300KB.

To trace the NFS email server, we used the CAMPUS trace from the Harvard study [5]. Using this trace, we reconstructed the file tree which would have existed on the CAMPUS server at Harvard University at the time they collected the trace. Similar file tree reconstruction techniques are described in [3,5]. While this tree is not an exact duplicate of the CAMPUS server, it replicated enough information so that we can replay the trace and collect meaningful data. This tree was then replicated on three different servers each built with one of the targeted file systems. Upon generation of the file system tree, an NFS client was programmed to generate requests to the target as read from the CAMPUS trace file. This NFS client made requests to the server as if a real request for I/O was made to the server. For the purposes of this work, we replayed a week's worth of Harvard's trace. To make our framework more precise, we sent the request to the server with the same time interval, as the trace file stated. On the server side, the thread was initiated. The CAMPUS trace also included userid, groupid and IP address information – these were ignored, since they were not relevant to our experiment.

The second workload was the synthetic Netbench [17] benchmark which exercises CIFS file servers. We used the Disk Mix Test suite to generate the synthesized emulation of network file server access from Windows clients in an office environment. While Netbench may not be an ideal real-world trace, it is based on real office patterns. More importantly, it does exhibit more write oriented behavior as opposed to the CAMPUS NFS Server. The Netbench Disk Mix Test Suite was run to test the behavior of file systems.

The final workload was the ODSL's Database Test Suite (ODSL-DBT-2) [20], inspired from the TPC-C benchmark [21]. It is an online transactional processing test, which simulates an inventory control database with several workers accessing the database for purposes such as viewing, update etc. The test was conducted for 50 warehouses under single connection for 18,000 seconds. The database we used for data management in this test was PostgreSQL 7.4.

TRACED SYSTEM

The systems which acted as NFS clients were Pentium 4, 2.4 GHz computers with 512MB of RAM each, while the server configurations were Pentium III, 800 MHz computers with 512MB RAM. Each file system was mounted on one individual server and a unique client was connected to it to perform the experiment.

For the Netbench [17] workload we used Client and Controller Windows machines to run the Netbench software. Using Samba, these Windows systems were connected to the test Linux server and separate servers were again used with different file systems mounted and traces were collected.

For the OSDL-DBT2 workload, we used the same Pentium 4 configuration as servers for testing with a fresh mount of the file system. With each run, a fresh installation of PostgreSQL was done, a new disk was mounted and a new database was created.

ANALYSIS

In this section we analyze the traces collected in our experiments. Our analysis is based on the following categories:

1. Read/Write Frequency
2. Inter-arrival Time Distribution
3. Request Size Distribution
4. Burstiness

Read/Write Frequency

Read/Write frequency is the percentage of read and write requests which were made on the disk. We use this parameter to define the kind of workload we are using. We obtained this ration for each file system under each workload. The exact figures for the read/write frequency can be seen in Table 1.

Since the CAMPUS server is predominantly an email server, it is natural to expect it to be read-dominated. The Harvard data indicated the same, as they found three times as many reads as writes. At the disk level, however the ratio of reads and writes are significantly more due to the extra metadata activity that is not present in the NFS trace. Our observation showed that the CAMPUS email server [5] was read-dominated, Netbench [17] write dominated, and OSDL Database workload had equal read/write ratio. All three file system followed the same pattern. It is interesting to see that the type of workload can have such a dramatic effect on the read/write frequency. Particularly, the fact that the email server is heavily read-oriented should lead storage system designers to optimize reads even at the expense of writes. However, the Netbench numbers, though it may indicate that file servers should be write optimized, should be taken with a grain of salt. The Netbench data set is only 22MB in size, which can easily fit in cache, thus explaining why there are so few reads. In the context of the traces we are collecting, however, the Netbench data is useful since it serves as an example of a write-dominated workload. Finally, the database workload is a good example of a workload that is evenly divided between reads and writes.

In terms of raw numbers, there are some points that stand out. With the heavily write-oriented Netbench workload, we see that that two journaling file systems, EXT3 and JFS, have nearly three times as many writes than EXT2. These extra writes are due to the writes to the journal. However, one may notice that the database workload does not show a similar increase in writes. The reason is that the database workload does only writes without any file system/metadata operations such as deleting a file, renaming a file, moving a file etc. while on the other hand, the Netbench workload does a lot of these operations. File writes are nor journaled, but file system modifications are journaled, thus the differences in behavior.

Inter-arrival Time

Inter-arrival time is defined as the time between successive disk accesses request. It is a useful measure to help identify the load on the disk system. In particular, the average inter-arrival time determines the required service from the disk. The inter-arrival time can also be used to develop workload models that can drive queuing studies. Inter-arrival Time Distribution for the same test for different file systems showed different patterns. Figures below show the comparison of the different file systems (EXT2, EXT3 and JFS) under various workloads with respect to the inter-arrival time in each case.

Email Server (read dominated)

It is clear from the distribution graph shown in Figure 1 that most of the requests are on average less than a second apart, with the majority of them less than about 400ms apart for EXT2.

A steep rise was observed in the count of inter-arrival requests between 0.2 and 0.4 seconds for EXT3, while EXT2 and JFS showed spikes near 0.6 seconds and 1 second. Figure 2 shows the distribution graph plotted for the read requests alone, which showed similar characteristics to that of the overall inter-arrival cumulative distribution graph.

Since nearly half of the inter-arrival times are very short, we magnified the graph for the region between 0 and 0.5 seconds (Figure 3). As can be seen, nearly half of the inter-arrival times are at 0.01 seconds. The short intervals are due to back to back NFS requests caused by a large NFS read broken into smaller NFS protocol limitations. To know the inter-arrival distribution of write requests, we plotted the graph for write requests also (Figure 4). All three file systems indicated almost the same behavior with EXT3 showing slightly more frequent accesses.

Netbench (write dominated)

On Netbench, EXT2 and EXT3 showed almost the same pattern, with EXT3 having slightly more frequent accesses. (Figures 5a & 5b).

Since Netbench is write dominated, the write access distribution is very similar to the overall inter-arrival time distribution (Figures 6a & 6b). Looking at Figure 5b, one can see that EXT2 and EXT3 have inter-arrival times that are much shorter than JFS. Nearly 90% of the intervals with EXT3 are less than 0.02 seconds. JFS tends to delay writes, and thus eliminate some writes through cache, thus accounting for the difference. It would seem that because of this behavior, JFS-based storage system would have longer periods of idle time. Idle time can be used by disk systems to run background disk optimization schemes, and this graph shows that atleast for write-dominated data, the choice of file system can influence the length of idle time available. Figure 7 and 8 shows the same type of behavior for reads but on a different scale. Note that since Netbench has so few reads, this data is not meaningful.

Database Workload (~equal read/write ratio)

Figure 9 and 10 shows the inter-arrival time distribution for the database workload. The most striking feature is that the inter-arrival times are much more frequent. Interestingly, for the database workload, the JFS file system shows much more frequent access than EXT3. This behavior is the complete opposite of what was observed for the read dominated and write dominated workloads. Nearly, 50% of the inter-arrivals are less than 3ms for JFS, while for EXT3, the 50% mark is reached only at 20ms.

To evaluate this further, we looked at reads and writes separately as shown in Figures 11-14. It can be seen that there is very little difference between the file systems for reads. However for writes, there is a marked difference. For JFS, about 80% of the intervals are less than the 10ms, whereas for EXT3, the 80% mark is reached only at 100ms. It is clear that JFS does not do a good job of grouping writes, when reads are intermixed with the writes.

Request Size Distribution

In this section we examine the request size distribution, a measure of the number of blocks in each request sent to the disk. We discuss the request size in each file system under the different workloads.

Table 1: Read/Write Frequency for the three file systems under three different workloads.

	Email Server			*NetBench*			*Database Workload*		
	EXT2	**EXT3**	**JFS**	**EXT2**	**EXT3**	**JFS**	**EXT2**	**EXT3**	**JFS**
Read	8,998,227	8,464,299	7,215,712	30	48	21	2,940,367	2,893,065	2,331,008
Write	246,579	337,898	182,263	16,720	45,991	46,193	2,156,496	2,125,214	1,712,358
Total	9,244,806	8,802,197	7,397,975	16,750	46,039	46,214	5,096,863	5,018,279	4,043,366
% Read	97.33	96.16	97.54	0.18	0.10	0.05	57.68	57.65	57.65
% Writes	2.67	3.84	2.46	99.82	99.90	99.95	42.32	42.35	42.35

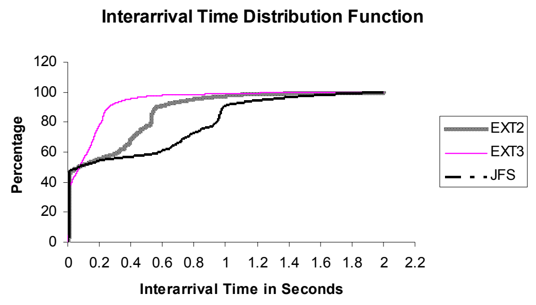

Figure 1: Inter-arrival Time Distribution for Email Server

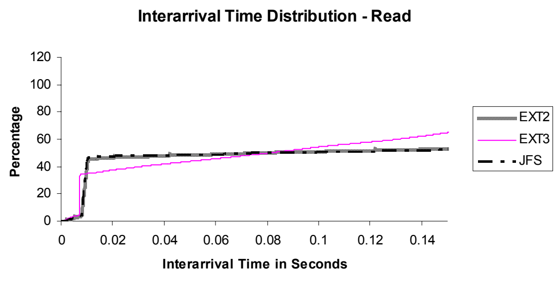

Figure 3: Inter-arrival Time Distribution for Read requests on Email Server

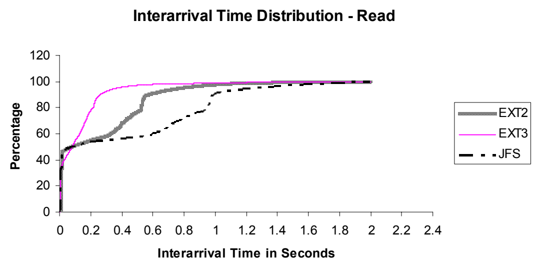

Figure 2: Inter-arrival Time Distribution for READ requests on Email Server

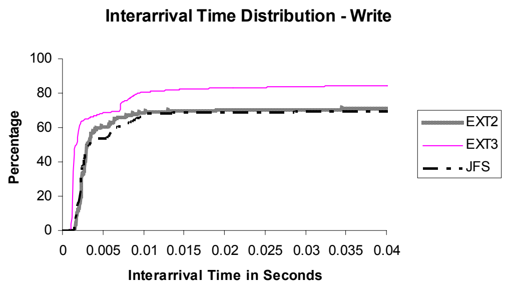

Figure 4: Inter-arrival Time Distribution for Write requests on Email Server

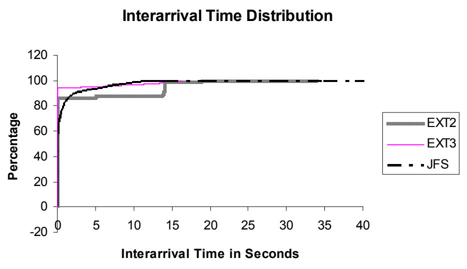

Figure 5a: Inter-arrival Time Distribution for Netbench

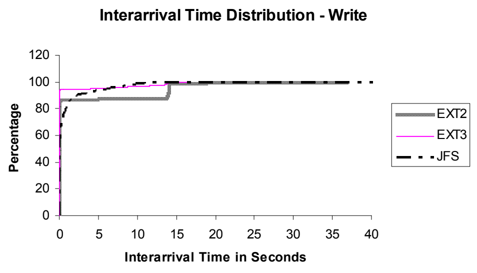

Figure 6a: Inter-arrival Time Distribution for Write requests on Netbench

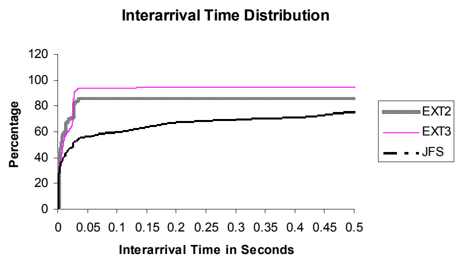

Figure 5b: Inter-arrival Time Distribution for Netbench

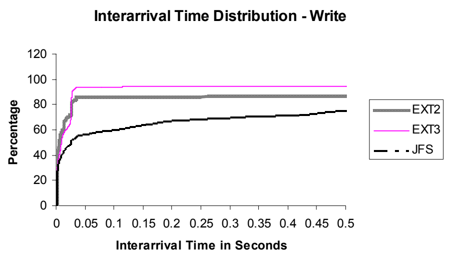

Figure 6b: Inter-arrival Time Distribution for Write requests on Netbench

Email Server (read dominated)

Request size distributed for the email server is shown in Figures 15-17. The read dominated email server mostly had large reads. There was a peak seen at 64Kbytes. All three file systems followed the same pattern for block.

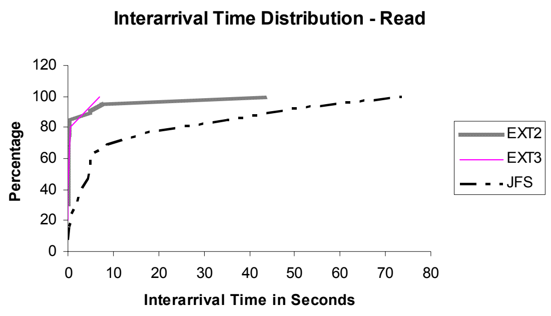

Figure 7: Inter-arrival Time Distribution for Read access on Netbench

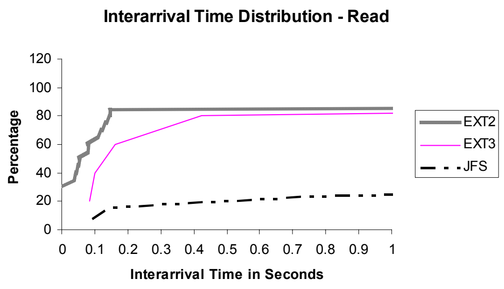

Figure 8: Inter-arrival Time Distribution for Read access on Netbench, magnified for 0-1 second zone.

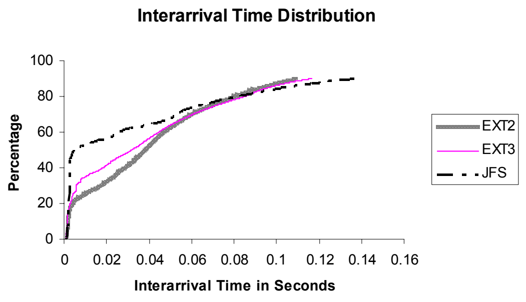

Figure 9: Inter-arrival Time Distribution for Database workload

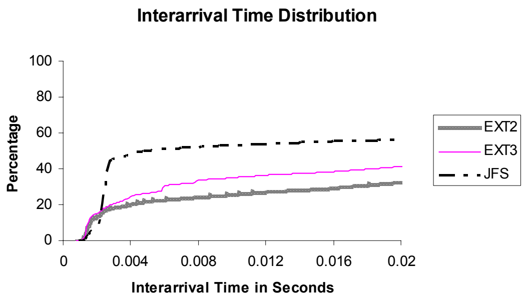

Figure 10: Inter-arrival Time Distribution for Database workload magnified over 0-0.02 sec.

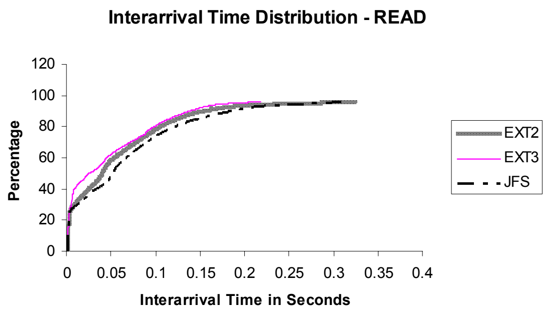

Figure 11: Inter-arrival Time Distribution for Read requests on Database workload

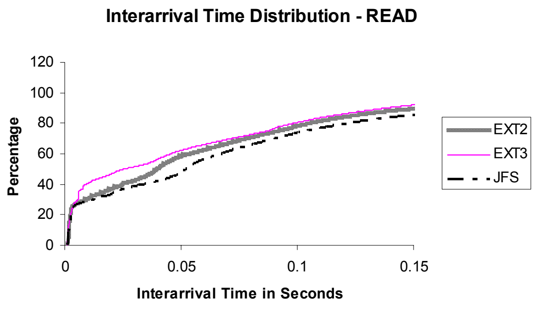

Figure 12: Inter-arrival Time Distribution for Read requests on Database workload for period of 0-0.15 seconds

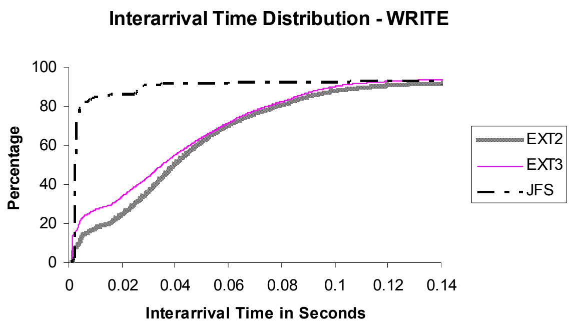

Figure 13: Inter-arrival Time Distribution for Write access on Database workload

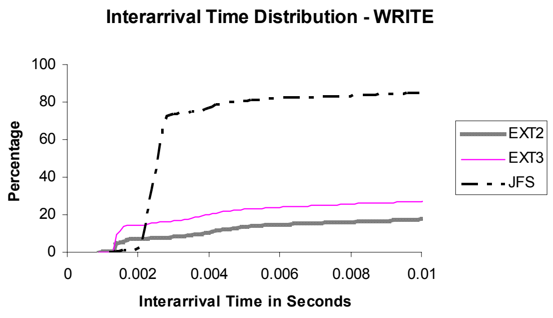

Figure 14: Inter-arrival Time Distribution for Write access on Database workload, magnified for the zone 0-0.01 seconds

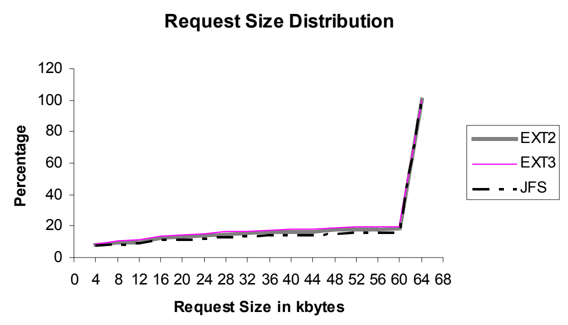

Figure 15: Request Size Distribution for Email Server

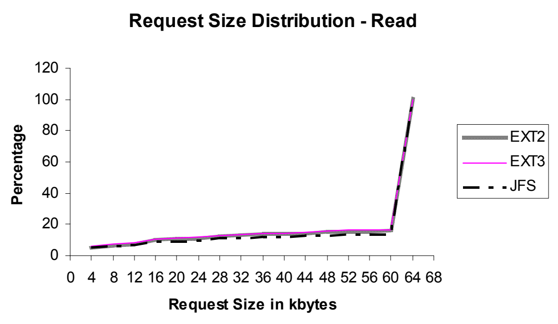

Figure 16: Request Size Distribution for Read requests on Email Server

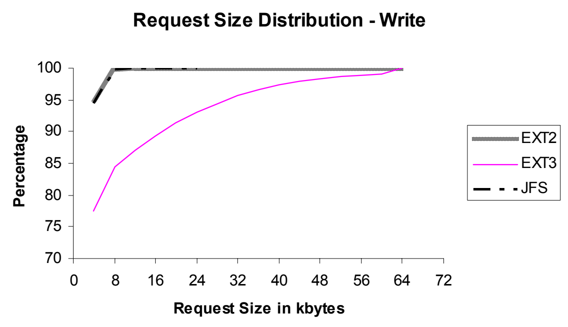

Figure 17: Request Size Distribution for Write requests on Email Server

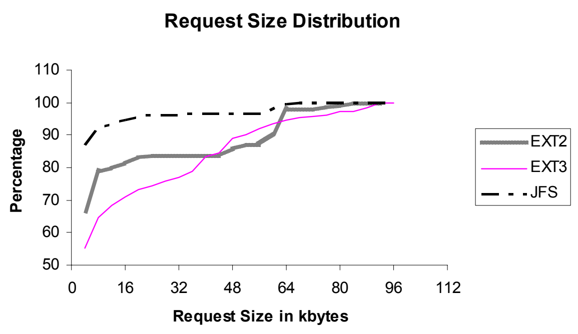

Figure 18: Request Size Distribution for Netbench

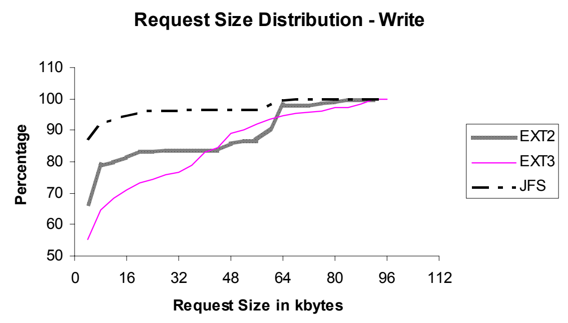

Figure 19: Request Size Distribution for Write requests on Netbench

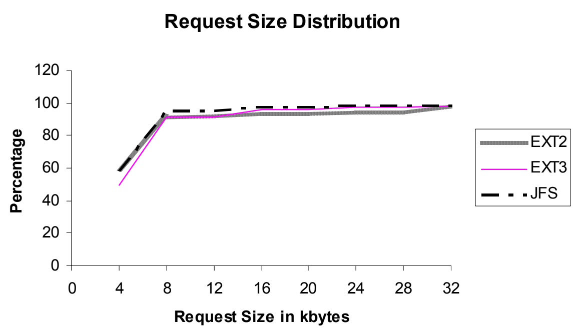

Figure 20: Request Size Distribution for Database workload

distribution. The read and write requests are separately shown in Figure 16 and 17 respectively. While the read behavior is the same for all file systems, for writes however we notice a difference. Most write requests in EXT2 and JFS are small in size when compared to EXT3 (Please note the change in scale on y-axis). It can also be observed that JFS did not show large writes at all. JFS for reliability reasons do not coalesce writes, thus explaining the lack of large writes.

Netbench (write dominated)

Netbench being a write dominated workload, read accesses on disk did not play much role in its request size distribution graph (Figures 18 and 19). All the read requests on disk under the Netbench workload were 4kb in size the default block size. All three file systems showed a small spike around 64 Kbytes, as was also observed in the read dominated email server.

Database Workload (~equal read/write ratio)

Figures 20-22 show the request size distribution for the database workload. As can be seen, the database workload issues were very small requests. Almost all its requests were 8 Kbytes in size. This behavior did not show much variation over the different file systems. EXT3 showed slightly fewer small read requests. The write pattern was dominated by small writes. Under this workload again, there was hardly any differentiation visible in the three different file systems distribution.

Burstiness

We define burstiness as a series of requests made on the disk, which lie within the time interval of 10ms of the previous request sent to the disk. It is a measure of how many accesses are in groups and provides a rough measure of how much idle time was available. The graphs above

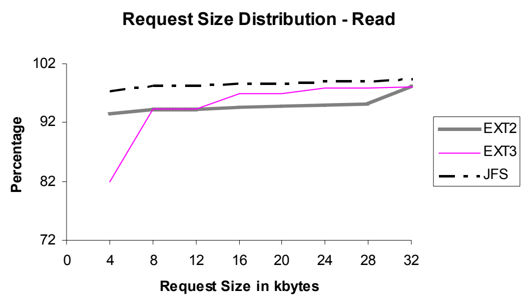

Figure 21: Request Size Distribution for Read requests under Database workload

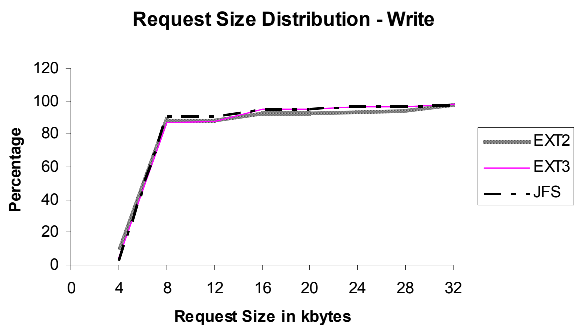

Figure 22: Request Size Distribution for Write requests under Database workload

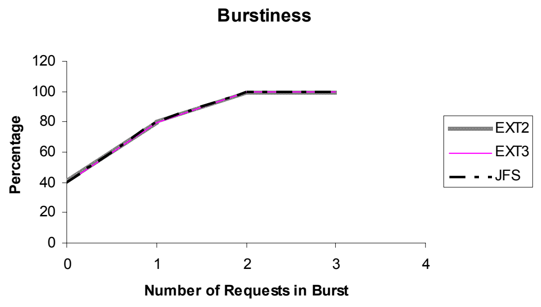

Figure 23: Burstiness for Email Server

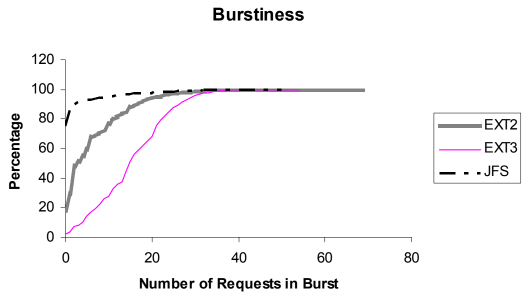

Figure 24: Burstiness for Netbench Workload

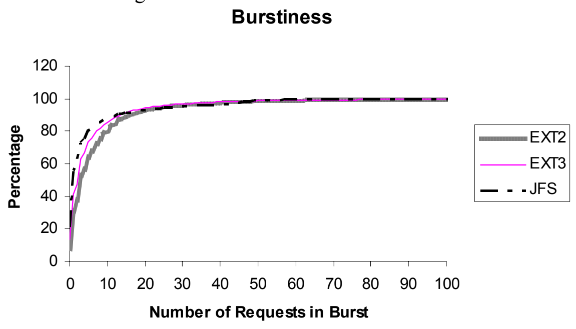

Figure: 25: Burstiness for Database workload

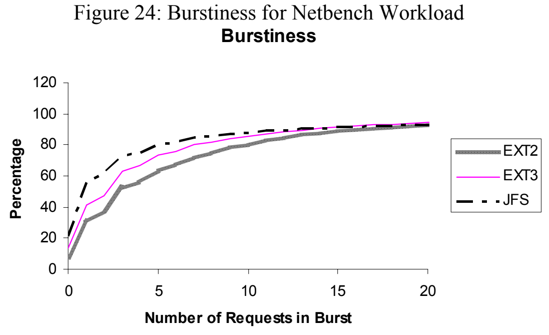

Figure 26: Burstiness for Database workload magnified for 0-20 requests in burst

show the cumulative distribution where the x-axis is the number of request in the burst.

Email Server (read dominated)

Under the read dominated workload, the three file systems did not show any variation from one another. No more than 3 requests occurred in any burst.

Netbench (write dominated)

Under the Netbench workload, the three file systems showed significant differences in the number of requests coming in burst (Figure 24). JFS had around 80% of its bursts in singletons, i.e. single-request bursts, whereas for EXT3, the bursts were typically much larger. Again, this behavior is because JFS is quite conservative in terms of coalescing writes to preserve reliability.

Database Workload (~equal read/write ratio)

Under the database workload, the three file systems showed similar patterns of very large amount of small bursts (Figure 25). The graph was hence magnified for better understanding (Figure 26). It can be seen that even in this workload, JFS seems to have shorter bursts than any other file systems.

CONCLUSIONS

In this paper we have presented new studies of disk I/O traffic under different workloads and different file systems. These findings can provide insights to storage and file system designers and moreover highlight the importance of file system choice in designing a storage system. System administrators can also analyze their workload and file system and pick a storage system that may match their particular workload and file system needs.

The key findings are:

- Email servers are heavily read dominated (>95% reads). This may indicate that in email servers, large caches either in the server or at the disk are warranted.

- File servers are very write dominated. Write dominated systems may be able to take advantage of log-structured file systems or disks [18.19]
- The file system has a very minor effect on read-dominated workloads.
- For write-dominated workloads, journaling file systems can cause an increase in small writes. This may lead an administrator to add NOVRAM to coalesce writes or not use RAID5 to address the small write problem with journaling disks.
- In an email server workload, there is not much burstiness, and what little there is consists mostly of 2-3 bursts. However, for a write dominated file serving workload, EXT2 and EXT3 show as significant burstiness as JFS. The presence of burstiness may indicate more available idle time to do any background work on disk – for example disk defragmentation, log cleaning etc.

ACKNOWLEDGEMENTS

Special thanks are due to Daniel Ellard for providing the NFS traces in the Harvard study.

REFERENCES

[1] C. Ruemmler and J. Wilkes, "UNIX Disk Access Patterns," *Proceedings of Winter USENIX,* Jan. 1993

[2] C. Ruemmler and J. Wilkes, "A Trace-Driven Analysis of Disk Working Set Sizes," Technical Report No. HPL-OSR-93-23, Hewlett Packard, Palo Alto, CA, Apr. 1993

[3] M.A. Blaze, "NFS Tracing by Passive Network Monitoring", *Proceedings of Winter USENIX,* pages 333-343, Jan. 1992

[4] M.G. Baker, J.H. Hartman, M.D. Kupfer, K.W. Shirriff, and J.K. Ousterhout, "Measurements of a Distributed File System," *Proceedings of the ACM Symposium on Operating System Principles*, pages 198-212, Oct. 1991.

[5] 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,* March 2003.

[6] K. Ramakrishnan, P. Biswas, and R. Karelda, "Analysis of File I/O Traces in Commercial Computing Environments," *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems,* pp. 78-90, 1992.

[7] W. Vogels, "File System Usage in Windows NT 4.0," *Proceedings of the ACM Symposium on Operating Systems Principles*, pp. 93-109, Dec. 1999.

[8] J.K. Ousterhout and H. Da Costa, D. Harrison, J.A. Kunze, M.D. Kupfer and J.G. Thompson, "A trace-driven analysis of the UNIX 4.2 BSD File System," *Proceedings of the ACM Symposium on Operating System Principles*, pp. 15-24, Dec. 1985.

[9] D. Roselli, J. Lorch, and T. Anderson, "A Comparison of File System Workloads," *Proceedings of USENIX Technical Conference*, pp. 41-54, 2000.

[10] W.W. Hsu and A.J. Smith, "Characteristics of I/O traffic in personal computer and server workloads," *IBM Systems Journal*, vol. 42, no. 2, 2003, pp. 347-372.

[11] E.L. Miller and R.H. Katz, "Analyzing the I/O Behavior of Supercomputers Applications," *Digest of Papers, 11th IEEE Symposium on Mass Storage Systems*, Monterey, CA, pp. 51-55, Oct. 1991

[12] M. Satyanarayanan, "A Study of file sizes and functional lifetimes," *Proceedings of 8th ACM Symposium on Operating Systems Principles*, pp. 96-108, Dec. 1981

[13] G.A. Gibson, D.F. Nagle, K. Amiri, F.W. Chang, E.M. Feinberg, H. Gobioff, C. Lee, B. Ozceri, E. Riedel, D. Rochberg, and J. Zelenka, "File Server Scaling with Network-Attached Secure Disks," *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Seattle, WA, June 1997.

[14] R. Card, T. Ts'o, and S. Tweedie, "Design and Implementation of the Second Extended Filesystem," *Proceedings of the Dutch International Symposium on Linux*, 1994.

[15] T. Ts'o and S. Tweedie, "Future Directions for the Ext2/3 Filesystem," *Proceedings of the USENIX Annual Technical Conference (FREENIX Track)*, Monterey, CA, June 2002.

[16] S. Best, "JFS overview: How the Journaled File System cuts system restart time to the quick," IBM Whitepaper, http://www.ibm-.com/developerworks/library/l-jfs.html, Jan. 2000

[17] VeriTest, Inc., "NetBench 7.0.2," http://www.netbench.com, 2001.

[18] J. Ousterhout and F. Douglis, "A Case for Log-Structured File Systems," *Operating Systems Review,* vol. 23, no. 1, pp 11-28, January 1989.

[19] R.Y. Wang, T.E. Anderson, D.A. Patterson, "Virtual Log Based File Systems for a Programmable Disk," in *Proceedings of Symposium on Operating System Principles,* pp. 29-43, Feb. 1999.

[20] Open Source Development Labs (OSDL) "OSDL-DBT2", http://www.osdl.org, 2003

[21] Transaction Processing Performance Council (TPC), http://www.tpc.org