Analysis of the Request Patterns to the NSSDC On-line Archive[1]

Theodore Johnson
ted@cis.ufl.edu

Code 630
NASA Goddard Space Flight Center
Greenbelt MD 20771

Dept. of CIS
University of Florida
Gainesville, Fl 32611-2024

Abstract

NASA missions, both for earth science and for space science, collect huge amounts of data, and the rate at which data is being gathered is increasing. For example, the EOSDIS project is expected to collect petabytes per year. In addition, these archives are being made available to remote users over the Internet. The ability to manage the growth of the size and request activity of scientific archives depends on an understanding of the of the access patterns of scientific users.

The National Space Science Data Center (NSSDC) of NASA Goddard Space Flight Center has run their on-line mass storage archive of space data, the National Data Archive and Distribution Service (NDADS), since November 1991. A large world-wide space research community makes use of NSSDC, requesting more than 20,000 files per month. Since the initiation of their service, they have maintained log files which record all accesses the archive.

In this report, we present an analysis of the NDADS log files. We analyze the log files, and discuss several issues, including caching, reference patterns, clustering, and system loading.

1 Introduction

On-line scientific archives are an increasingly important tool for performing data-intensive research. Building a large-scale archive is an expensive proposition, however, and system resources need to be carefully managed. To date, there has been little published research that studies the performance of on-line scientific archives.

The National Space Science Data Center (NSSDC) of NASA Goddard Space Flight Center has run their on-line mass storage archive of space data, the National Data Archive and Distribution Service (NDADS), since November 1991. A large world-wide space research community makes use of NSSDC, requesting more than 350,000 files in 1994. Since the initiation of their service, they have maintained log files which record all accesses to the archive.

In this paper, we present an analysis of access patterns to the NDADS. These analyses are based on the information contained in the log files. We discuss several aspects of system performance, including the performance of several caching algorithms on the recorded request stream, and the effectiveness of the data clustering used by NDADS. We show that the request for a file are bursty, and that user requests are bursty. Finally, we present an analysis of the system load.

Several studies on the reference patterns to mass storage systems have been published. Smith [12] analyzes file migration patterns in hierarchical storage management system. This analysis was used to design several HSM caching algorithms [13]. Lawrie, Randal, and Burton [7] compare the performance of several file caching algorithms. Miller and Katz have made two studies on the I/O pattern of supercomputer applications. In [9], they find that much of the I/O activity in a supercomputer system is due to checkpointing, and thus is very bursty. They

[1] This work was performed while Theodore Johnson was an ASEE Summer Faculty Fellow at GSFC. This research is partially supported by grant from NASA through USRA, #5555-19

make the observation that much of the data that is written is never subsequently read, or is only read once. In [10], they analyze file migration activity. They find a bursty reference pattern, both in system load and in references to a file. Additional studies have been made by Jensen and Reed [5], Strange [14], Arnold and Nelson [1], Ewing and Peskin [3], Henderson and Poston [4], Tarshish and Salmon [15], and by Thanhardt and Harano [16]. However, all of these studies apply to supercomputer environments, which can be expected to have access patterns different from those of a scientific archive.

1.1 Log Files

The National Space Science Data Center is the primary archive for all space data collected by NASA. The NSSDC distributes its data using a variety of methods and media. For example, one can request photographs, CD-ROMs and tapes from the NSSDC. Manually filling orders for data is labor intensive and hence expensive. In addition, service is slow. To reduce data distribution costs and to improve service to the user community, the NSSDC created the National Data Archive and Distribution Service to store electronic images and data, and serve the data electronically.

The archive consists of a two jukeboxes storing WORM magneto-optic disks, one with a capacity of 334 GB, the other with a capacity of 858 GB. A user submits a request by naming a project, and the files of the project. Request submission is most often done by email, but can also be done using a program on the host computer, and through a new World Wide Web service. NDADS will fetch the requested files from nearline storage, place the requested files on magnetic disk, then notify the user that the files are available for transfer via ftp (alternatively, the files can be ftp'ed automatically). More information about NDADS can be found by sending an email message to archives@nssdc.gsfc.nasa.gov with a subject line of "help".

A user specifies the files of interest by naming them explicitly. In general, specifying files by predicate matching is not possible (although this capability is being developed).

NDADS is an evolving system, and log file collection is part of the evolution. Version 1 logs were recorded between November, 1991 and December, 1993. These logs record the files requested, the start and stop times of request service, and the name of the requester. Unfortunately, these log files do not include the file sizes or the name of the media from which the file was fetched. These log files were intended to aid in monitoring and debugging the system, not for performance modeling. Many of the deficiencies of the version 1 logs were fixed in version 2. The version 2.1 and 2.2 logs were collected between January, 1994 to mid-July, 1994. These logs include file size and media name information, permitting a much more detailed analysis. Version 2.3 logs start in mid-July, 1994 and are still being collected at the time of this writing (January 1995). These logs include information about ingest as well as request activity.

2 Caching

When a user requests a file, the file is fetched from tertiary storage into secondary storage and made available to the requester. The file typically has a *minimum residency requirement* (three days in NDADS) to give the requester time to access the file. The archive systems needs to have enough disk storage to satisfy the minimum residency requirement.

While the file is disk-resident, a second request for the file can be satisfied without fetching the file from tertiary storage. These cache hits can reduce the load on the tertiary storage system, and also improve response times.

A large body of caching literature exists when all cached objects are of the same size. The Least Recently Used (LRU) replacement algorithm is widely recognized as having good performance in practice, although statistical algorithms with better performance have been proposed recently [6, 11].

Caching objects of widely varying sizes is somewhat more complicated, and has not received the same amount of attention. If one wants to minimize the number of cache misses, then it is much better to choose large files than small files for replacement, because removing large files frees up more space. The optimal replacement algorithm for variable size objects, with respect to cache misses, is the GOPT algorithm [2]: Let F be the set of cached files, and for file $f \in F$, let N_f be the time until the next reference to f and let S_f be the size of f. Choose for replacement the $f' \in F$ whose product $N_{f'} * S_{f'}$ is the largest.

The GOPT algorithm cannot be implemented (because it requires knowledge of future events), but it can be approximated. The Space-Time Working Set (STWS) algorithm [13] approximates GOPT be substituting P_f, the time since the last reference to f, for N_f.

While STWS can be implemented, it also requires a great deal of computation. For this reason, STWS is often approximated by what we call the STbin algorithm [8]: A file is put into a bin based on its size. The files in a bin are sorted in a list using LRU. To choose a file for replacement, look at the file at the tail of each bin and compute its $P_f * S_f$ product. Choose for replacement the file with the largest space-time product.

In our caching analysis, we use the LRU, STWS, and STbin algorithms. We assume a disk block size of 1024 bytes, and set a limit on the number of disk blocks that are available for caching. We trigger replacement when fetching a new file will cause the space limit to be exceeded, and we remove files until the space limit will not be exceeded. For the STbin algorithm, bin i holds files that use between 2^i and $2^{i+1} - 1$ blocks.

We execute the caching algorithms on traces generated from the 1994 log files (which have size information attached). We divide the logs into three month periods, to make the logs large enough to capture the steady-state hit rates, but also indicate changes in the access patterns.

The hit rate information is summarized in Table 1. The STWS and STbin algorithms have much better performance than the LRU algorithm. The STbin algorithm usually has performance comparable to that to the STWS algorithm, and sometimes has better performance. One surprising result is the high hit rate (up to 50%) that is possible with a moderate sized (5 Gb) cache. Given the nature of the archived data, hit rates were expected to be much lower.

When a file is fetched from tertiary storage, it remains on magnetic disk for at least three days. For a comparison, we present the disk storage requirements and the hit rates if an 3-day residency is observed, in Table 2. As the table shows, considerable more than 5 Gb of disk storage is required to satisfy the minimum residency requirement.

The resources required to fetch a file depend on the size of the file. For this reason, STWS is suboptimal is practice. Most vendors allow the user to tune the caching algorithm to reduce the penalty paid by very large files. A common technique is to assign to each file a weight computed as $P_f * S_f^c$ for a constant $c \leq 1$. In Table 3, we list the number of bytes transferred by each of the caching algorithms. LRU generally transfers the fewest bytes, closely followed by STWS. In these log files, STbin requires the transfer of many bytes (STWS transfers fewer bytes than STbin because it has a lower miss rate).

2.1 File Access Pattern Analysis

The success of caching depends on the access patterns. In this section we examine some aspects of the access patterns.

Disk Blocks (1k bytes)	Hit Rate LRU	STWS	STbin	Hit Rate LRU	STWS	STbin
	January, 1994 - March 1994			April, 1994 - June 1994		
1048576	.144	.234	.195	.155	.235	.189
2097152	.243	.314	.267	.202	.313	.194
3145728	.288	.341	.337	.272	.362	.286
4194304	.309	.355	.349	.292	.423	.448
5242880	.320	.364	.360	.304	.471	.500
	July, 1994 - September 1994			October, 1994 - December 1994		
1048576	.173	.270	.205	.127	.215	.198
2097152	.248	.308	.259	.155	.243	.222
3145728	.271	.328	.309	.181	.266	.245
4194304	.302	.340	.340	.219	.291	.256
5242880	.327	.366	.377	.252	.311	.287

Table 1: Hit rates for different cache replacement algorithms.

	1/94 - 3/94	4/94 - 6/94	7/94 - 9/94	10/94 - 12/94
hit rate	.175	.183	.193	.129
Storage	7.2 Gb	8.4 Gb	14.0 Gb	11.0 Gb

Table 2: Disk storage and hit rates for 3-day residency.

Disk Blocks	Hit Rate LRU	STWS	STbin	Hit Rate LRU	STWS	STbin
	January, 1994 - March 1994			April, 1994 - June 1994		
1048576	35.0 Gb	34.9	36.7	43.2	43.4	44.8
2097152	30.1	32.1	34.9	41.2	41.1	44.6
3145728	29.1	30.8	31.8	39.1	39.6	42.6
4194304	28.4	29.8	30.7	38.0	38.1	38.8
5242880	27.8	29.1	29.9	37.2	36.6	36.6
	July, 1994 - September 1994			October, 1994 - December 1994		
1048576	56.7	57.7	61.0	35.4	35.9	37.3
2097152	52.7	55.1	59.6	32.7	34.8	36.2
3145728	50.1	53.3	57.9	31.7	33.5	35.2
4194304	47.4	52.2	55.8	29.2	31.6	34.3
5242880	46.3	50.0	52.7	28.3	29.8	32.1

Table 3: Disk blocks moved for different cache replacement algorithms.

Year	Jan. - March		April - June		July - Sept.		Oct. - Dec.	
	total	unique	total	unique	total	unique	total	unique
1992	12806	73%	28899	61%	41296	67%	53253	65%
1993	47046	66	31513	69	49058	65	26106	61
1994	77415	62	92325	45	113594	55	74606	67

Table 4: Total and unique references (in percentage) to NDADS.

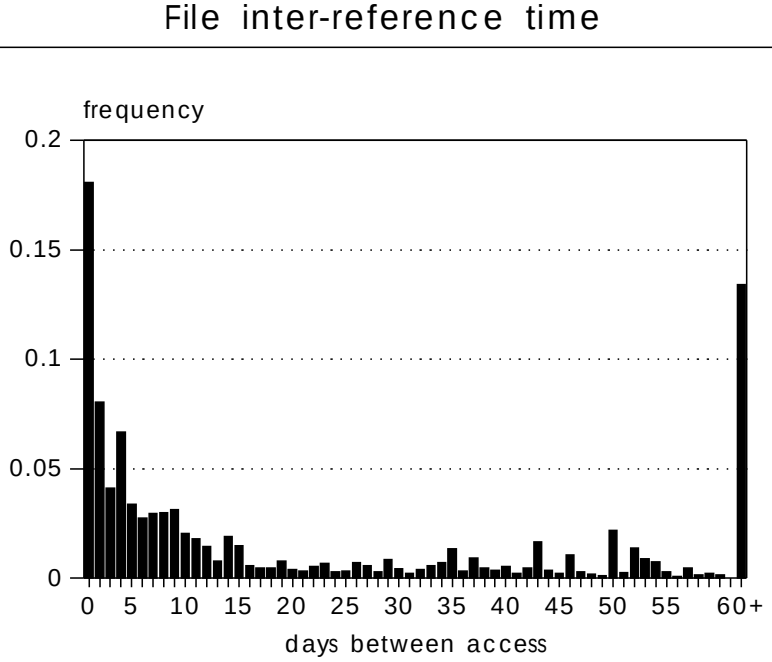

Figure 1: Distribution of file inter-reference times.

The number of possible cache hits depends on the number of duplicate references in the reference stream. In Table 4, we show the number of references and the number of unique references by three month period. The 1994 data shows that the STWS and STbin algorithm are getting close to the best possible hit rates.

The effectiveness of caching also depends on the average time between references to a file (the *inter-reference* time). In Figure 1, we plot the distribution of inter-reference times during 1994. To generate this plot, we scanned through all file accesses and searched for repeat accesses. Whenever a repeated reference was found, we incremented a histogram based on the number of days since the last reference. The plot shows that most repeat references occur shortly after an initial access, but that the inter-reference time distribution has a long tail. The average number of days between an access to a file, given that the file is accessed at least twice in 1994, is 27.5 days. There is a sharp peak at 3 days that does not fit well with the curve. We speculate that this is a side effect of the three-day residence period (users re-submit their request when they find that the files have been removed from disk storage area).

The effectiveness of STWS also depends on the distribution of file sizes. In Figure 2, we plot the distribution of sizes of the files accessed in 1994. The average size of a file accessed in 1994 is 560 Kbytes. Because of the wide range of sizes, we created the histogram by binning on the base two logarithm of the file size. Most of the files accessed in this archive are between 128

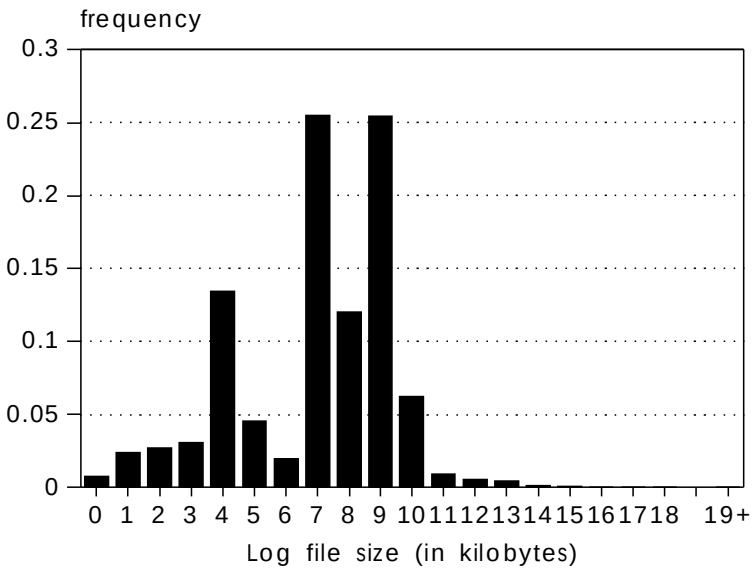

Figure 2: Distribution of file sizes.

	Number of users			
year	Jan. - March	April - June	July - Sept.	Oct. - Dec.
1992	153	185	230	300
1993	430	552	677	607
1994	678	689	692	670

Table 5: Growth in the user population.

Kbytes and 1Mbyte in size. Few files larger than 2 Mbytes are accessed, but this number does not go to zero. Figure 3 shows the file access rate weighted by the file size. When we examine the number of bytes moved, files larger than 2 Mbytes account for a significant fraction of the system activity.

Finally, we look at the rate at which files of different sizes are re-accessed. In Figure 4, we plot the percentage of file accesses which are repeat accesses, binned on file size. This plot shows that small files (except for the very smallest) have a low re-access rate, and that the very large files have a high re-access rate. If the cost of transferring large files is significant, then STWS is a distinctly suboptimal caching policy. Because STWS strongly discriminates against very large files, it will often incur the cost of their transfer.

3 User Access Analysis

A model of request to the archive depends on the users of the system. In the accumulated log files, we have notices that the user population is growing. We first note that the user population is growing, as is indicated by Table 5.

Most users make only a few requests to the archive. Figure 5 plots a histogram of the percentage of users that make different numbers of requests to the archive in a single month.

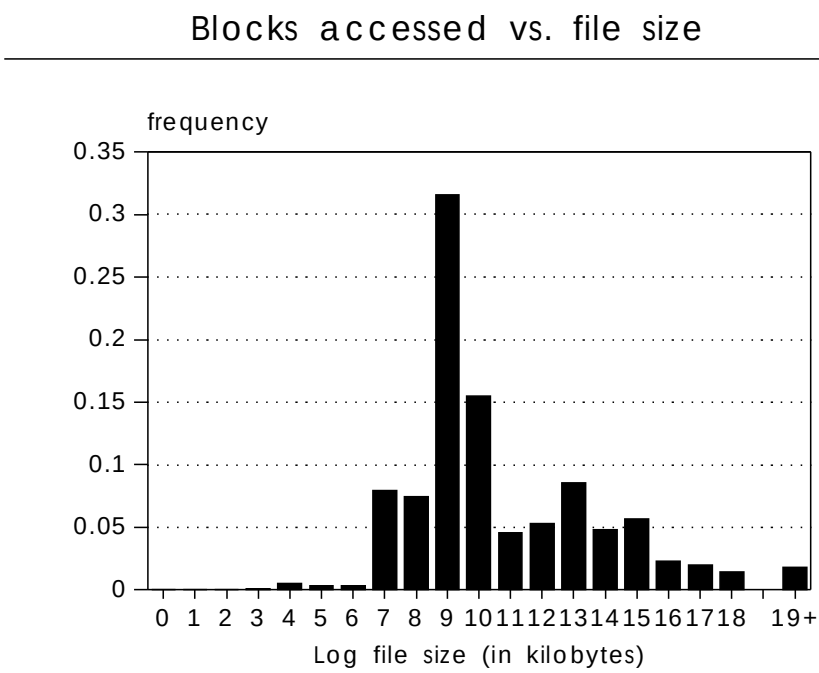

Figure 3: Distribution of file sizes, weighted by number of blocks.

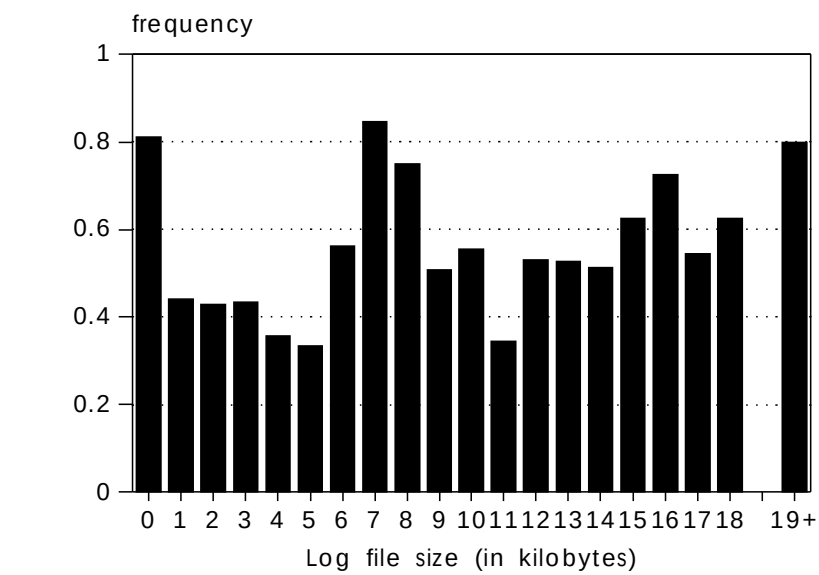

Figure 4: Probability of re-referenceing a file, by file size.

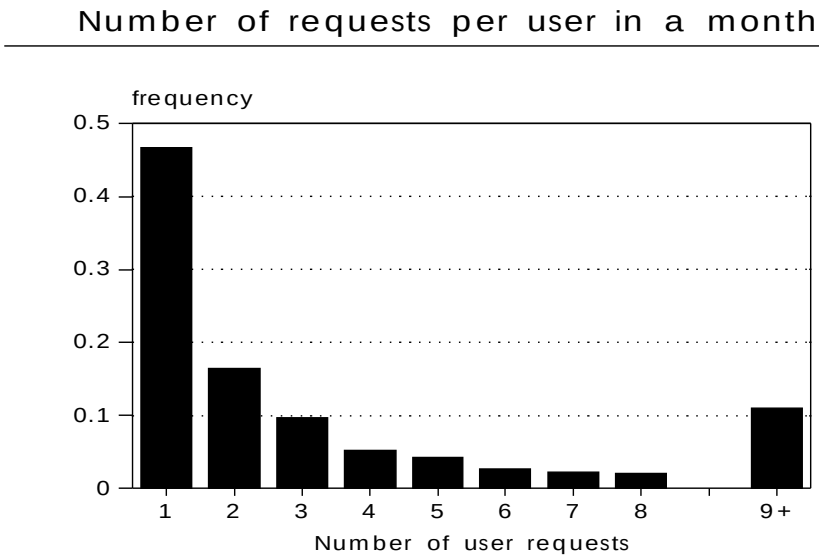

Figure 5: Requests per user.

This plot also shows that there is a moderate size core of users who make make requests. The top ten heaviest users make an average of 30 to 50 requests per month.

Finally, we plot the time between requests from a user in Figure 6. This plot shows that user activity is very bursty, as more than 80% of repeat requests occur within 3 days of the previous request.

We found that a large fraction of repeat requests for a file are due to the user that previously requested the file. The fraction of repeat request due to the same user is plotted in Figure 7, binned on the number of days since the last reference.

4 Clustering

The efficiency of a tertiary storage system depends in large part on how well the data in the archive is clustered with respect to the average request. The throughput of a drive in the tertiary storage device is zero while new platters are being loaded, or while the drive is seeking the file on the media. If the files of a single request are scattered throughout many media, and at widely varying locations in the media, the throughput of the tertiary storage device will be much lower than its potential.

The NDADS system is built over WORM storage, which has short seek times. Therefore, the most expensive overhead occurs when a new platter has to be loaded to fetch a file. Also, the 1994 log files contain the platter on which each file is stored, but not the tracks on the platter.

Files in NDADS are divided into *projects* (i.e., the satellite that generated the images contained in the file). An optical platter contains files for only one project, (but a project may be spread over many platters) to simplify the management of the platters. This policy actually aids in clustering, because all files in a request must be from the same project. If a project generates enough data to require several platters, the files are assigned to the platters in a way that is hoped to reduce the number of platters that must be accessed to satisfy a typical access. This method of placement depends on the project and the expected type of access.

For every user request, we collected the number of files requested and the number of platters needed to satisfy the request. We found that the NDADS clustering of files onto platters is effective, as the average request asks for about 27 files, spread across about 2 platters. The

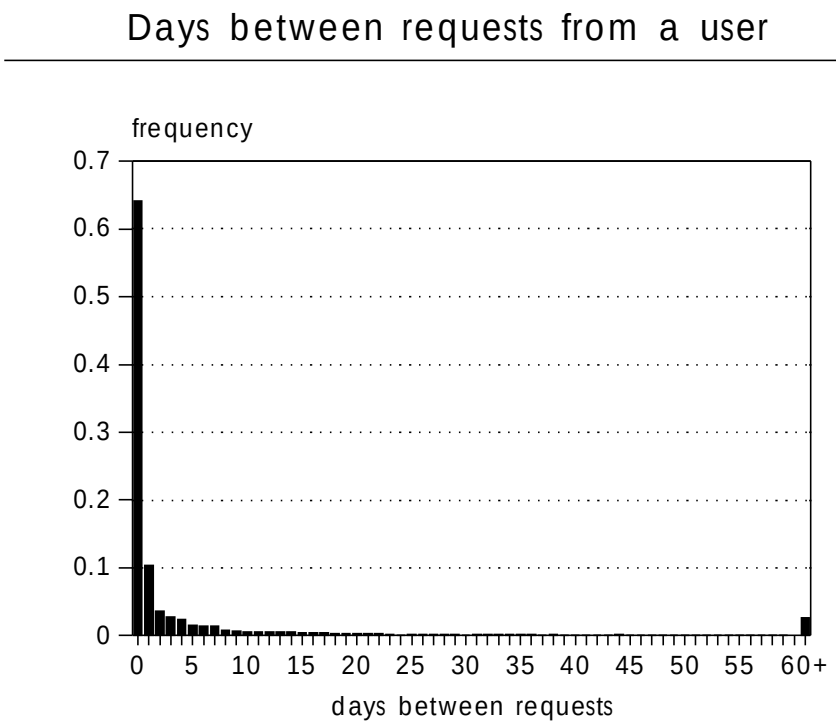

Figure 6: Time between repeat requests from a user.

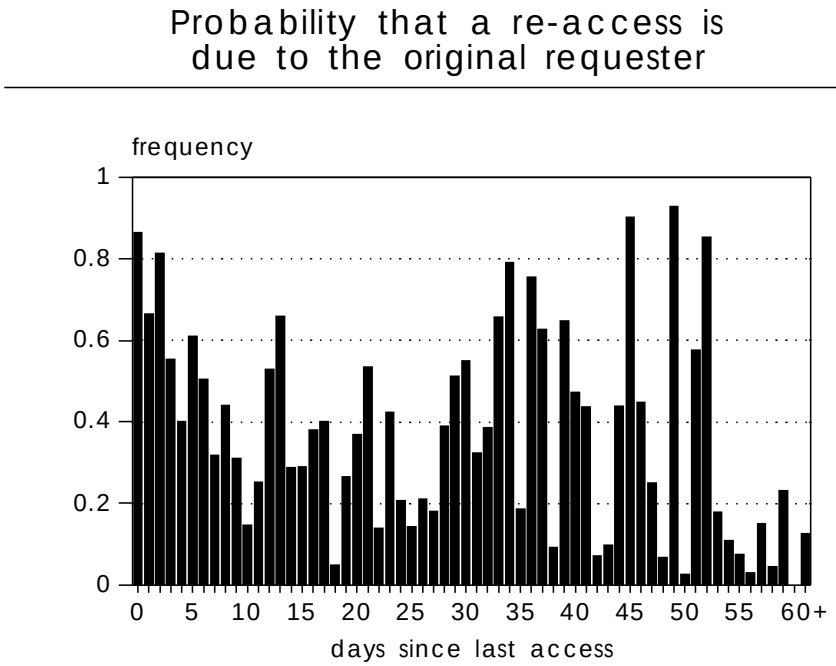

Figure 7: Fraction of repeat accesses from the original user.

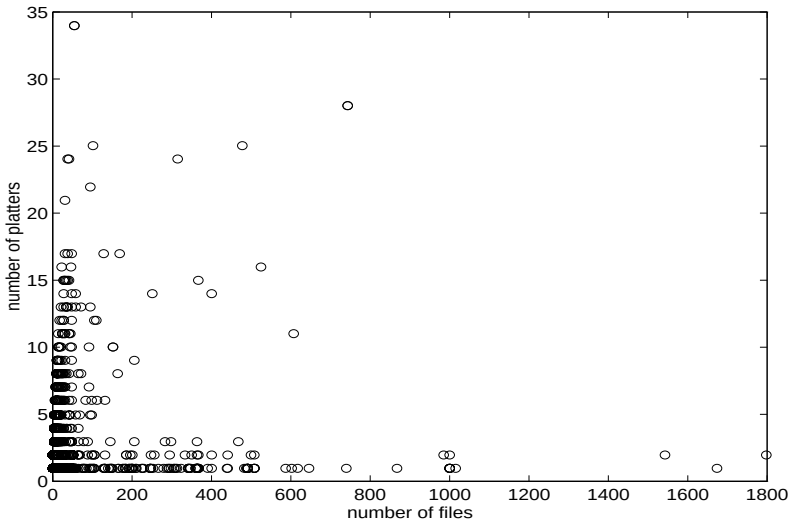

Figure 8: Scatter plot of platters per request vs. files per request. The data is was collected between July and September 1994.

number of platters required to satisfy a request is not correlated with the number of files in the request. This property is illustrated in Figure 8, which is a scatter plot of the number of platters to satisfy a request versus the number of files in the request. The data in this plot is taken from the period July 1994 to September 1994, but is typical of the total data. Most of the points in the plot are close to either the X or Y axis. The shape of the plot indicates that the clustering is appropriate for most requests, but a small fraction of the requests require a completely different clustering pattern (as is to be expected).

We also noted that some platters are accessed much more frequently than others. In Figure 9, we plot the number of platters that have different numbers of references. The plot shows that many of the platters receive only a few references, but that the distribution has a long tail. Eighty four of the platters are very hot, serving more than 500 files during 1994. The hottest platter served 112646 files.

5 System load

We computed the *system load* by summing up the number of seconds required to service all request submitted during a period of time, then dividing by the number of seconds of the observed period. We plot the system load per month for 1993, 1993, and 1994 in Figure 10. While the month to month load shows great variability, the load per month does not follow a pattern that is strongly adhered to in all three years of the observation. However, we can note that there is a usage peak in March-April, another in July-August, and low demand in January.

We next plot the system load per day of the week in Figure 11. Here, we can find a strong trend, that people tend to submit requests on weekdays instead of weekends.

Finally, we plot the system load per hour of the day. We again see that people tend to submit requests during normal working hours. The strong peak in load during (Greenbelt) working hours also indicated that most users of NDADS work in the western hemisphere. We note that

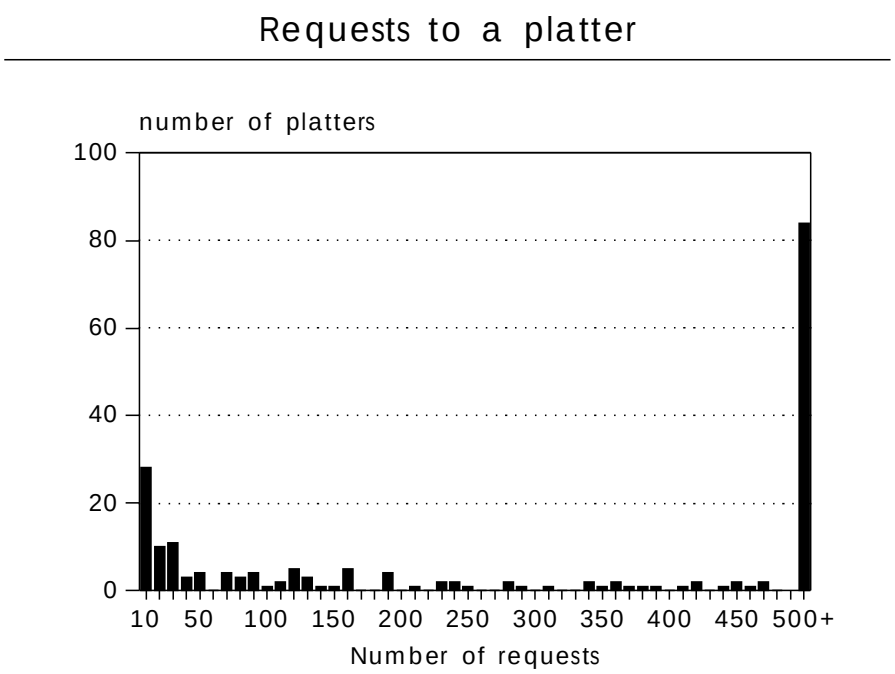

Figure 9: Number of platters receiving different numbers of references.

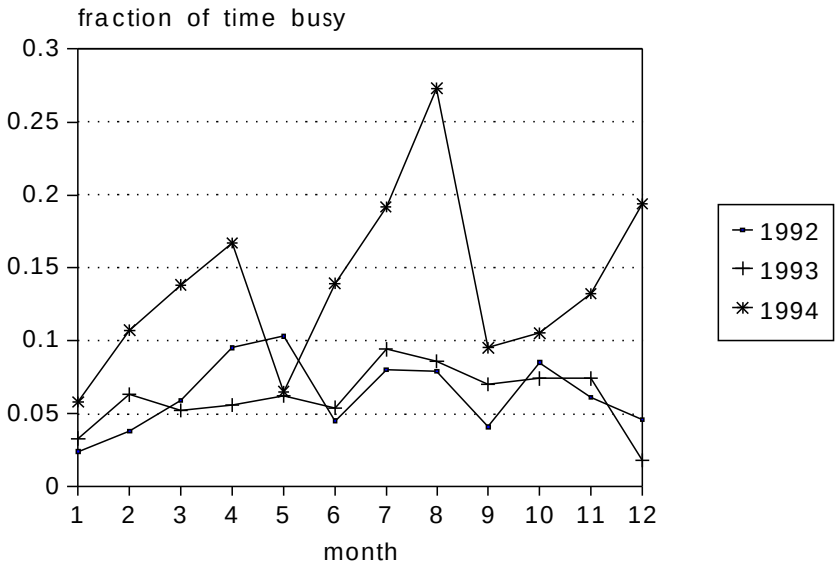

Figure 10: Average load per month. 1 is January and 12 is December.

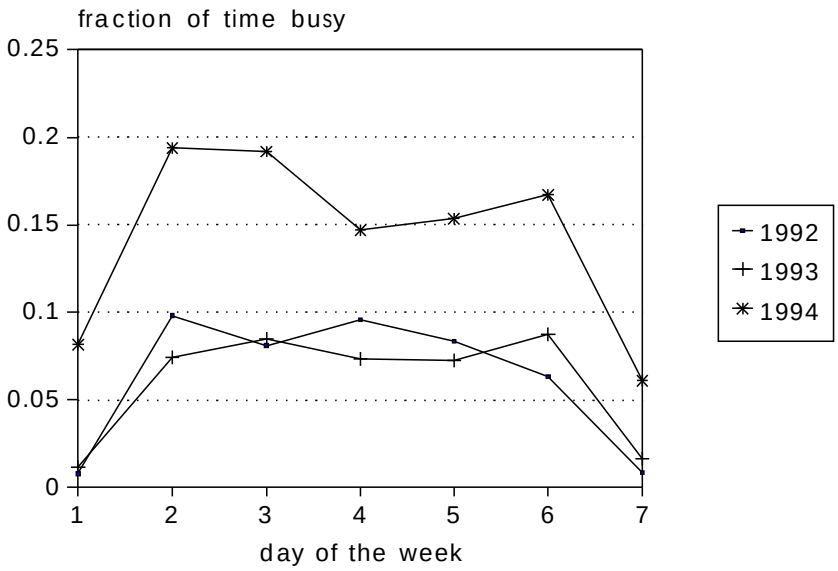

Figure 11: Average load per day of the week. 1 is Sunday and 7 is Saturday.

a survey of the email addresses of user requests shows international use of NDADS.

We have also recorded the system load due to ingest. Ingest contributes about .16 to the system load, and shows a pattern that varies in time that is similar to that of file requests.

6 Conclusions

We have studied the access characteristics of the access requests to the National Data Archive and Distribution Service (NDADS) of the National Space Science Data Center (NSSDC), of NASA's Goddard Space Flight Center. Much of NASA's electronic science data is available on through the NDADS archive. The log files present an opportunity to understand the access patters of requests to scientific archives.

We can make the following observations about the user request pattern:

- Caching can be effective. 59.4% of all files requested in 1994 had been requested previously in 1994. High hit rates (30% to 50%) can be achieved by using a space-time working set algorithm.
  - Many of the repeat requests are due to the same user. The high proportion of short-term repeat requests from the same user indicates some users are uncertain of whether their request was received. The high proportion of long-term repeat request from the same user indicates that NDADS is being used as a substitute for local storage.
  - While very large files constitute a small proportion of the total number of requests, they constitute a moderately large proportion of all bytes fetched from tertiary storage. Very large files tend to have a high repeat access rate. These two facts indicate that a caching algorithm should not discriminate too strongly against large files.
  - Access to a file is bursty. A large proportion of repeat accesses occur within 4 days of the previous access. The distribution of the time to the next access also has a

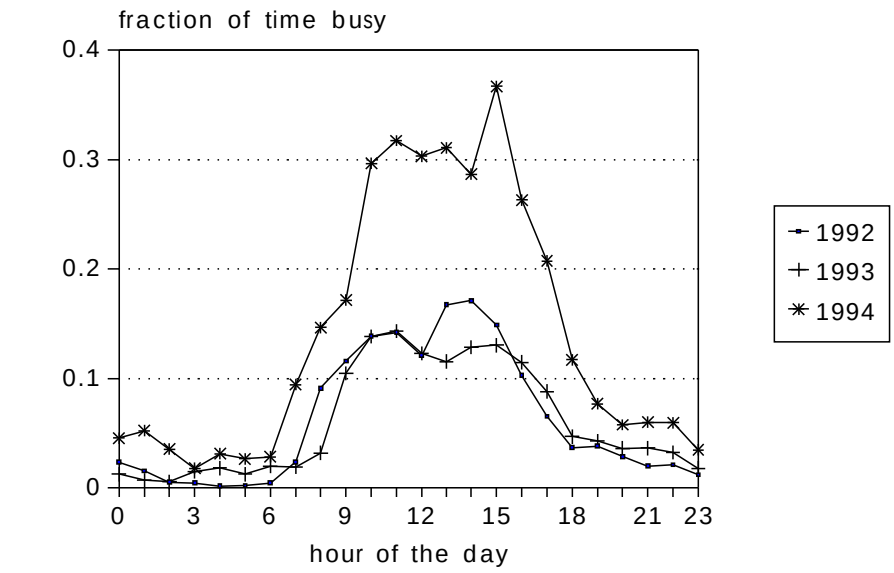

Figure 12: Average load per hour of the day.

long tail. These two facts suggest that a model of file access rates should have steady-state component and a bursty component triggered by an access.

- User request patterns tend to be very bursty. This fact combined with the high proportion of repeat requests that are due to the same user explains some of the bursty nature of file accesses.
- Access to NDADS tends to follow normal working hours. There is an increase of activity preceding important scientific events.
- The user community grew rapidly during the first year of operation, then grew at a slower pace during 1993 and 1994. The intensity of use by each user grew from 1993 to 1994.
- File access shows a great deal of clustering.
  - Most requests are satisfied by a few one or two platters. There is little correlation between the number of files requested and the number of platters required to service the request.
  - Clustering is important for performance. Although the average system utilization is low, the system load increases significantly during working hours and during certain months. If the time required to service a request doubled, NDADS would have difficulty in meeting peak demand.
  - Some data volumes are much more popular than others. During 1994, there were 84 very hot platters (i.e., served more than 500 files) and 28 very cold platters (i.e., served 1-10 files).

Acknowledgements

We'd like to thank Jeanne Behnke, Joe King, and Michael Van Steenberg for their help with this project.

References

[1] E.R. Arnold and M.E. Nelson. Automatic Unix backup in a mass storage environment. In *Usenix – Winter 1988*, pages 131–136, 1988.

[2] P.J. Denning and D.R. Sluts. Generalized working sets for segment reference strings. *Communications of the ACM*, 21(9):750–759, 1978.

[3] C.W. Ewing and A.M. Peskin. The masstor mass storage product at brookhaven national laboratory. *Computer*, pages 57–66, 1982.

[4] R.L. Henderson and A. Poston. MSS II and RASH: A mainframe unix based mass storage system with a rapid access storage heirarch file mamangement system. In *USENIX – Winter 1989*, pages 65–84, 1989.

[5] D.W. Jensen and D.A. Reed. File archive activity in a supercomputing environment. Technical Report UIUCDCS-R-91-1672, University of Illinois at Urbana-Chanpaign, 1991.

[6] T. Johnson and D. Shasha. 2Q: a low overhead high performance buffer management replacement algorithm. In *Proc. of the 20th Int'l Conf. on Very Large Databases*, pages 439–450, 1994.

[7] D.H. Lawrie, J.M. Randal, and R.R. Barton. Experiments with automatic file migration. *Computer*, pages 45–55, 1982.

[8] E. Miller, 1994. Provate communication. Thanks also to comp.arch.storage.

[9] E.L. Miller and R.H. Katz. Analyzing the i/o behavior of supercomputing applications. In *Supercomputing '91*, pages 557–577, 1991.

[10] E.L. Miller and R.H. Katz. An analysis of file migration in a unix supercomputing environment. In *USENIX – Winter 1988*, 1993.

[11] E.J. O'Neil, P.E. O'Neil, and G. Weikum. The lru-k page replacement algorithm for database disk buffering. In *Proceedings of the 1993 ACM Sigmod International Conference on Management of Data*, pages 297–306, 1993.

[12] A.J. Smith. Analysis of long-term reference patterns for application to file migration algorithms. *IEEE Trans. on Software Engineering*, SE-7(4):403–417, 1981.

[13] A.J. Smith. Long term file migration: Development and evaluation of algorithms. *Communications of the ACM*, 24(8):521–532, 1981.

[14] S. Strange. Analysis of long-term unix file access patterns for application to automatic file migration strategies. Technical Report UCB/CSD 92/700, University of California, Berkeley, 1992.

[15] A. Tarshish and E. Salmon. The growth of the UniTree mass storage system at the NASA Center for the Computational Sciences. In *Third NASA Goddard Conf. on Mass Storage Systems and Technologies*, pages 179–185, 1993.

[16] E. Thanhardt and G. Harano. File migration in the ncar mass storage system. In *Mass Storage Systems Symposium*, pages 114–121, 1988.