Analyzing Storage System Workloads

Paul G. Sikalinda*, Pieter S. Kritzinger and Lourens O. Walters

E-mail: {psk, psikalin}@cs.uct.ac.za, Lourens.Walters@s1.com

Abstract— Disk storage subsystems have not kept up the speed with processors. Processor performance has been increasing at a much higher rate than that of disk drives. Therefore, I/O subsystem has become a bottleneck in current computer systems. With this in mind the research community is looking into ways of improving the I/O subsystem. IBM and HP are among the organizations doing research and development of high performance storage systems also referred to as Enterprise Storage Systems (ESSs). Much of this effort goes into the evaluation of these systems for correctness and performance. For these evaluations, using simulations or otherwise, to be reliable, there is need to correctly understand and model the disk I/O workloads also known as I/O traffic or physical I/O workloads. Therefore in our work, we analyzed I/O workload traces to derive statistics which can be used as a guide in the (a) modelling of I/O workload and subsequent production of I/O workload representative of the actual I/O workload for evaluation and (b) optimization of the storage systems. Our results show that the distribution of inter-arrival times of I/O requests are heavy-tailed, and that the I/O request sizes are a function of the operating system.

Index Terms—Distribution, I/O Workload Trace, RAID, Synthetic I/O Workloads, Trace Generator.

I. INTRODUCTION

As processor speeds and the volumes of data stored on disks continue to increase, improving computer system performance has recently focused on improving the speed of storage systems and retrieval of data and the reliability of corresponding storage systems. RAID (Redundant Array of Independent Disks) architectures have become the architecture of choice for large storage systems since they employ striping (i.e. parallelism) and redundancy across multiple disks to increase capacity, speed, and reliability of storage systems. Analytic and simulation models for analyzing the performance and correctness of these systems have recently become popular in research literature[16].

ESS workloads are described in terms of the following parameters: logical volume, arrival time, start address, request size, type of operation (i.e. read or write) of the I/O requests. I/O workloads are represented by either a measured workload trace or a synthetically generated trace. A synthetically generated trace is one where the workload parameter values such as request sizes are generated from either analytic or empiric distributions. In the evaluation, synthetic workloads are used because they are flexible and easier to obtain than actual traces of production workload. To generate representative synthetic I/O workloads for use in the performance and correctness evaluation of storage systems, one has to analyze actual I/O workloads to derive proper models of I/O workloads.

One way of improving I/O response time is to develop better optimization techniques and algorithms in the operations of the ESSs. To achieve good performance for ESS, the I/O traffic patterns must be known, and the storage system optimized for such patterns. One can not suggest reasonable policies or optimize these policies without first understanding the I/O workload characteristics or usage patterns. Thus, our analysis of I/O workload based on I/O traces will provide information that can be used to formulate policies such as the cache replacement policy and the data placement scheme in the storage system for the workload for which the traces were collected.

The rest of the paper is organized as follows. In section II, we give the motivation of our work. In this section we explain in detail the need of understanding the I/O traffic for modelling and optimization purposes. In the following section we discuss related work. In section IV we give our methodology with a description of the traces we used. We also discuss the statistical tools used to analyze the data set from the I/O workload traces. In section V we give the results of our analysis. Finally section VI provides conclusion drawn from our work.

II. MOTIVATION

Synthetic I/O workloads are used in the design of storage system using software tools such as RAIDframe[7], which is used for design and development of RAID storage systems, and RAIDsim[8], which is used for performance and correctness modelling of RAID storage systems. Such synthetic I/O workloads require an understanding and analysis of measured I/O workloads.

Ganger in [1] points out that the largest difficulty facing

P. G. Sikalinda* is an MSc Student at the University of Cape Town (UCT) in the DNA Research Group, Department of Computer Science (phone: 650 3127, private Bag: Rhodes Gift 7707).

P. S. Kritzinger is with the University of Cape Town (UCT) in the DNA Research Group, Department of Computer Science.

L. O. Walters is with Mosaic Software, Rondebosch, Cape Town

***Corresponding Author**.

storage architects and performance analyst is identifying and obtaining I/O workloads with which to evaluate and compare designs. He says that raw I/O workload traces can be used directly to drive a simulation of storage systems, but because of the following reasons they are rarely used:

- For non-technical reasons, it can be extremely difficult to convince systems administrators to allow tracing.
- Traces, which tend to be very large, must be stored online when they are to be used for experiments.
- Each trace represents a single measure of behavior, making it difficult to establish statistical confidence in results.
- It is very difficult to isolate and/or modify specific characteristics (e.g., arrival rate) of a trace.
- Traces do not support studies of expected future workloads, since one cannot trace what does not yet exist.

On the other hand, he states that synthetic I/O workload traces are usually used in experiments because:

- Synthetic traces can be manufactured on the fly rather than stored.
- One can generate many synthetic traces with the same characteristics and thereby achieve some statistical confidence.
- Changes to the synthetic traces can be made much more readily.
- Synthetic traces representing expected future environments can be generated.

Therefore our work centers on providing statistics to aid in the selection of models to use for generating synthetic I/O workloads representative of the actual I/O workload under consideration. It involves automatic analysis of an ESS workload in form I/O traces and providing statistics for the values of the workload parameters.

There is a danger in using synthetic traces. If a synthetic trace, which is a poor match to the original, is generated the validity of the results in experiments are obviously compromised.

Ganger in [1] states that it is difficult to come up with synthetic I/O workload to use in the evaluation of storage system which is truly representative of the actual I/O workload. He points out that much research is still needed to develop representative synthetic I/O workloads. He states that because the difficulty with obtaining representative workloads is pervasive, many researchers are forced to use whatever is available (usually, ad hoc synthetic I/O workloads with many simplifying assumptions).

Ganger developed an approach to validate synthetic disk request trace generators. In this approach, statistics are collected from a single pass over a timed sequence of disk requests. These statistics are then fed into a trace synthesis tool, which produces a sequence of requests with the same characteristics. Assuming that the synthesis tool has no errors, this will be successful if the statistics capture all important information. To evaluate the synthetic requests, both request sequences (traced and synthetic) are used to exercise a simulator. Performance measurements from the resulting simulator activity are used to evaluate the closeness of the synthetic trace.

Ganger found that the *commonly applied assumptions (e.g., uniform starting address, Poisson arrivals)* are inappropriate and can produce dramatically incorrect results. For inter-arrival times, Hsu et al[2] agrees with Ganger that the Poisson distribution is not a proper model. Instead Hsu et al[2] found the lognormal distribution to be a better model of the *arrival pattern* which defines the sequence of arrival times.

An I/O *access pattern* is a sequence of accesses defined by the following parameters: logical volume, operation type (i.e., read or write), start address, and size of the I/O request. In our findings, we discovered that no one has attempted to analyze access patterns in I/O traffic with the aim of finding better models for the following parameters:

- Request size and
- Seek distance i.e. difference between consecutive start addresses, of I/O requests.

As already mentioned in the introduction, one of the keys to overcoming the I/O bottleneck is to understand how storage is actually used, so that the storage system can be optimized for the usage patterns, or if need be, new optimization techniques and algorithms can be designed[2], [9], [14].

These techniques and algorithms include:

- *Pre-fetching (read-ahead) policy.* The pre-fetching strategy simply pre-fetch blocks of data that are being accessed sequentially. Therefore understanding the access patterns of the workload is very important to decide whether this strategy will be effective for a given workload. [5] mentions that this strategy will be effective if the workload consists of large data accessed sequentially.
- *Cache management policy*. This is the policy used to decide what data blocks to keep in the cache at a given time. In this case, the cache is a faster memory than disk placed between the processor (the operating system) and the disk which holds data that has recently been read and, in some cases, adjacent data areas that are likely to be accessed next. How well the cache absorbs read requests is a very important factor. An effective cache management policy will bring about a high hit ratio and reduce seek times improving the overall performance of the storage system. To come up with an effective replacement policy workload analysis has to be done. For example, the least recently used (LRU) replacement policy will work well if the I/O workload is such that once a block of data is read, it will be used many times in given period of time.

- *Write delay*. The write delay is the time written blocks of data are kept in the write cache before they are written back to the disk. By delaying writes, blocks that are deleted or overwritten in the interval need not be written at all. This will reduce disks writes and improve the performance of storage system. To come up with the right write delay, one has to analyze I/O workloads.
- *Data placement scheme*. This is the scheme used to allocate segments of files to multiple disks. Roselli in [5] points out that to reduce disks seeks, most file systems organize their layout to optimize for either reads or writes depending on whether read traffic or write traffic dominates. In an array of disks, understanding usage patterns is important for *load balancing*.
- *Pooling of storage resources*. [2] states that storage systems managed by various entities in an enterprise are likely to be consolidated through the use of storage utilities such as Storage Area Networks or storage service providers (SSPs). To tell whether such pooling of resources is more efficient depends on the I/O characteristics of the workloads, and in particular, on whether the workloads are independent.
- *Implementation of intelligent storage systems*. [2] also mentions that the growth of processing power available in the storage systems makes it possible to build intelligent storage systems that can dynamically optimize themselves for the actual workload. Here also understanding I/O workload characteristics is important. For example we need to know the idle times of the storage system which can be used to run background tasks to do the optimization.

III. RELATED WORK

In the analysis of ESS workloads, traces are used. The traces can be classified into three depending on where they were collected:

- I/O traffic traces. These are traces collected at the physical level, between the operating system and the storage system, and represent the I/O workloads also known as I/O Traffic.
- File system traces. These are traces collected at the logical level, between the application and the operating system, and represent the file system workloads.
- Database I/O traces. These are traces collected at the logical level but from the point of view of a Database Management System (DBMS) and not the file system. These represent the Database I/O workloads [3], [4].

File system and DBMS workloads are called *logical I/O workloads*. File system level workloads have been characterized in details [5], [11], [15]. Several studies of the logical I/O characteristics of database and scientific systems have also been conducted [3], [4]. Results show that logical workload events exhibit locality of reference, are self-similar for short periods, are bursty and have more read operations than write operations.

Compared to the analysis of I/O behavior at the logical level, physical I/O behavior has received much less attention. Hsu et al[2] pointed out that part of the reason is that storage level characteristics are sensitive to the file system and buffer pool design and implementation, so that the results of any analysis are less broadly applicable. As a result there is need to analyze the physical I/O characteristics of many different environments. Hsu et al[2] analyzed I/O traffic of servers and personal computers and their results show that I/O traffic exhibits locality of reference, is bursty and self-similar, with a read/write ratio less than one.

Our findings show that they are two categories of research concerning logical and physical I/O workload analysis. The first category involves analyzing workload traces with the aim of modelling the workload and discovering some storage usage patterns. The second category covers research in the development of software tools, called workload analyzers, to perform workload analysis given the workload traces. For example, Alistair Veitch and Kim Keeton of Hewlett Packard Laboratories have developed a general tool, called Rubicon[10], for the characterization of I/O workloads. In general, Rubicon reads a sequence of disk trace records, performs some analysis on them, and outputs the result of the analysis. This tool provides a rich set of operations on I/O traces, and was designed to be easily extended through the addition of new analysis functions and reporting methods.

IV. METHODOLOGY

For our I/O workload analysis, we used a repository of web I/O traces made available to the public by the Storage Performance Council (SPC). These traces were collected from a machine hosting a web search engine.

We analyzed the start address data in terms of the *difference in the consecutive start address.* We refer to this parameter as "*seek distance*" since we believe the difference in the consecutive start address would be proportional to the seek distance between consecutive I/O requests.

From the trace data we extracted the *request sizes* and calculated the *inter-arrival times* and *seek distances.* For each of these three parameters (i.e. inter-arrival time, request size and seek distance) we calculated key data statistics such as the sample mean, five number summary (a list of the smallest value, lower quartile, median, upper quartile, and largest value), variance, standard deviation, coefficient of kurtosis, coefficient of variation and coefficient of skew. This statistics describe the distribution

of the values of the parameters. The mean gives us the average value. From the median, lower quartile and the upper quartile we can deduce the ranges over which some fraction of values lie. For example the median value tells us that half of the values are below it and half of the values are greater than it. The coefficient of kurtosis tells how symmetrical the distribution is while the coefficient of skew tells us about the skewness of the distribution of the values.

TABLE I
SUMMARY STATISTICS FOR INTER-ARRIVAL TIMES

Sample Size	1055448
Five Number Summary	(126, 242, 1695, 4487, 100100)
Sample Mean	2985.761
Sample Variance	12508927
Standard Deviation	3536.796
Coefficient of Variation	1.184554
Coefficient of Skew	2.142186
Coefficient of Kurtosis	8.884555
Upper Outlier	26142

From the mean and the median we can learn about the location of the distribution. The mean and the median are called the measures of location as they purport to locate the "middle" of the data values. We also used the histograms and empiric cumulative distribution function graphs to get the visual impression of the distribution of the data. Due to space limitations not all our figures are included in this paper.

V. RESULTS

A. Request Inter-arrival Time

Table I reports the summary statistics for the inter-arrival time data. The data were measured in microseconds and contained values in the interval (126, 100100) microseconds. The very large range over which values were distributed could be attributed to the fact that there were periods during which the applications did not access the drives very much. According to the value of the upper outlier, the bulk of data had values in the interval (126, 4487).

The five number summary indicated a severe skew to the right in the distribution of the data i.e. the range between the minimum value and the median is 1569 microseconds compared to that of the median and the maximum value, which is 98405 microseconds. The values for coefficients of skewness, kurtosis and variation confirm the large skew to the right and indicated highly variable data.

B. Request size

The second workload parameter which we analyzed is the request size from the trace data set. This parameter, combined with the inter-arrival time constitutes what is called the intensity distribution of I/O workload as opposed to the locality distribution discussed in the next section. Table II shows summary statistics for request sizes.

The histogram of this data shows that this data has a multimodal distribution. The reason for this is not difficult to understand. Under UNIX operating system, both *read()* and *write()* system calls typically make use of the concept of a Filesystem *Block*. A *Filesystem Block* is a unit of data that an operating system typically uses when reading from or writing data to physical devices. It is configurable and most operating system use a setting of 8192 bytes. Disk reads and writes are typically not multiples of 8192 bytes,

TABLE II
SUMMARY STATISTICS FOR REQUEST SIZES

Sample Size	1055449
Five Number Summary	(512, 8192, 8192, 24580, 1138000)
Sample Mean	15510
Sample Variance	102017528
Standard Deviation	10100.37
Coefficient of Variation	0.6512577
Coefficient of Skew	3.441212
Coefficient of Kurtosis	287.6503
Upper Outlier	106520

and disk space is therefore wasted when only a few bytes are written to a *Filesystem Block* of 8192 bytes. Therefore, most operating systems allow reads and writes of sizes smaller than 8192 bytes to be performed in order to conserve disk space.

After removing the few outlying data values, i.e. data with values larger than 106520, the histogram of the remainder of the data contained four peaks at values 8192, 16384, 24576 and 32768. See figure 1. 60% of the data had a value of 8192 bytes, 10% had a value of 16384 bytes, 9% had a value of 24576 bytes, and 20% had a value of 32768 bytes so that 99 % of the requests had one of these

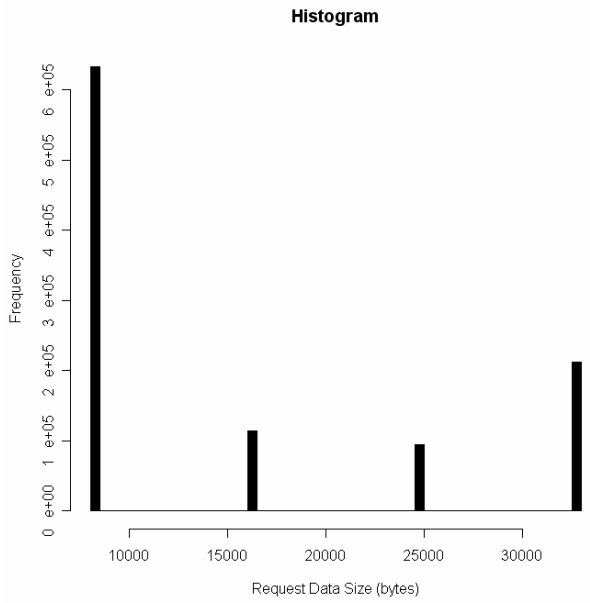

Fig. 1. The histogram of data sizes

sizes.

It would seem therefore that I/O request sizes depend almost entirely on the particulars of the operating system file management system, rather than the requirements of the application and the best way to model this parameter would be simply state the percentage of each usage of each peak value which is derived from an analysis of the relevant I/O trace and the percentage of the remaining sizes.

C. Request Locality

Table III shows summary statistics for disk *seek distances* for the web data set. There were very few *write* I/O requests in the data set from the Web search engine trace. This is understandable as a web search application does not typically write much data to the secondary storage system. The distribution of the distances for the Web search request data is symmetrical. See Figure 2. The data revealed a concentration of 57% of data in the interval (-10000000, 10000000).

TABLE III
SUMMARY STATISTICS FOR SEEK DISTANCES

Sample Size	1055448
Five Number Summary	(-34926160, -8581248, 6.4, 8580496, 34910700)
Sample Mean	27.95
Sample Variance	170691900000000
Standard Deviation	13064910
Coefficient of Variation	467398.8
Upper Outlier	51482656
Lower Outlier	-51482528

We investigated the rather curious symmetry in the data set and discovered that for every seek distance there was another seek distance value of the same magnitude but in the opposite direction. Such values did not necessarily occur in close succession and at the time of writing we have no explanation for this very interesting phenomenon. An explanation may well be found by determining how the search algorithm(s) employed by the application work. From the histogram of the data set, it was clear that none of the distribution density function would fit the read request seek distance for Web search data, even if considering all seek distances as positives and thus folding the data around the origin. It might be possible to model the data by using a mixture of two or three distributions. Clearly, knowing why the read seek distances are symmetric would be useful to optimize web searching algorithms at a system I/O or hardware level.

It should be evident from the results in this section that seek distance distribution is, not surprising, very much a function of application and general naïve assumptions of uniform seek distances are clearly not realistic.

VI. Conclusion

There is little research publicly available on the analysis of I/O requests despite the fact that storage traffic is of major importance for the performance of computer system. One reason is that real I/O traces are not easy to find. The second reason is that analysis of I/O traffic does not lend itself to make simple statements about their models as we have shown in this paper. I/O traffic patterns are sensitive to the file system design and implementation and results of any I/O traffic analysis are less broadly applicable.

Inter-arrival times show a pattern consistent with the findings of Hsu et al[2]. From our analysis the distribution of inter-arrival times looks to be heavy-tailed. Therefore, in the modelling of inter-arrival times probability density functions which are known to model such distribution should be used. These probability density functions include the lognormal, weibull, and the gamma probability distribution functions. The distribution of inter-arrival times being heavy-tailed means that the probability of having extreme inter-arrival times is not negligible. Thus the storage systems can take advantage of idle time to do background tasks such as rearranging data blocks on the disk to improve performance.

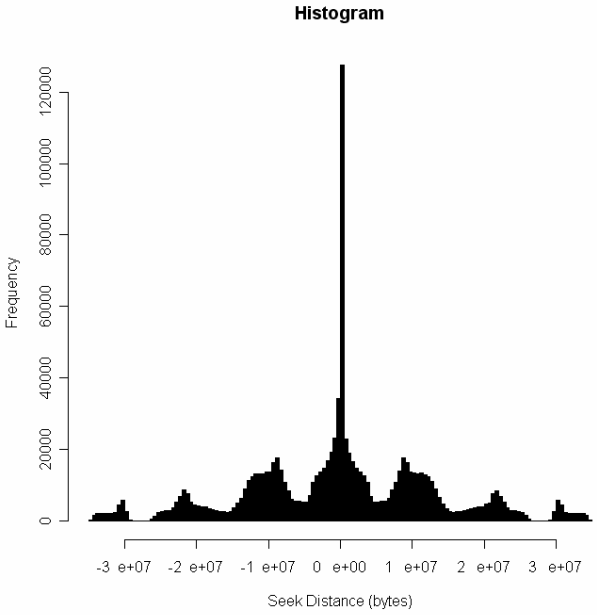

Fig. 2. The histogram of seek distances

However the size distribution of an I/O request is very much a function of the operating system behavior rather than the requirements of the application as we saw. Hence to model I/O request sizes, one has to take into consideration the configuration of the file system in use. While we have mechanical disk drives, the locality of the data on the physical disks determines the main part of the access time in the form of seek time. However, of the three workload determining parameters, inter-arrival time, request size and seek distances, locality defined by the seek distances is the most difficult to model.

A potential area of research is to rigorously fit the data sets for the parameter we have analyzed to probability function using statistical methodologies. We hope that more work will follow and soon we shall be able to know as much about I/O traffic characteristics as we do about web traffic characteristics. Not least of all we believe the information we have discovered will help storage designers to optimize aspects of the storage system which are not directly related to the physical drives, such as the cache management, both at the disk array controller and disk drive level.

REFERENCES

[1] G. R. Ganger, "Generating representative synthetic workload: An unsolved problem," In Int. CMG Conference, pp. 4-8, Nashville, TN, USA, 1995, Computer Measurement Group.

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

[3] W. Hsu et al., Characterization of Production Database Workloads and TPC Benchmarks. IBM Systems Journal, vol 40, no. 3, 2001, pp. 781-802.

[4] W. Hsu, A. Smith and H. Young, "I/O Reference Behavior of Production Database Workloads and TPC Benchmarks-An analysis at the Logical Level," ACM Transactions on Database Systems, vol 26, 2001, pp. 96-143.

[5] D. Roselli et al., "A comparison of File System Workloads," In Proceedings of USENIX Annual Technical Conference, San Diego, CA, 2000, USENIX Association, Berkeley, CA (2000), pp. 93-109.

[6] B. Pasquale et al., "A Static Analysis of I/O Characteristics of Scientific Applications in a Production Workload", Conference on High Performance Networking and Computing archive, Proceedings of the 1993 ACM/IEEE conference on Supercomputing, pp. 388-397.

[7] W. Courtright II et al., "RAIDframe: rapid prototyping for disk arrays," In Proceedings of the 1996 Conference on Measurement and Modelling of Computer Systems (SIGMETRICS), vol 24, 1996, pp. 268-269.

[8] P. Chen and D. Patterson, "Maximizing performance in a striped disk array," In Proceedings of ACM SIGARCH Conference, 1990, pages 322-331, ACM.

[9] K. Ramakrishman et al., "Trace Driven Analysis of Write Caching Policies for Disks," In Proceedings of ACM Conference on Measurement and Modelling of Computer Systems (SIGMETRICS), Santa Clara, CA, 1993, ACM, New York(1993), pp. 13-23.

[10] A. Veitch and K. Keeton,"The Rubicon workload characterization tool," SSP technical report HPL-SSP-2003-13, HP Laboratories, March 2003.

[11] K. Ramakrishnan et al. "Analysis of File Traces in Commercial Environments," Performance Evaluation Review, vol. 20, No. 1, June 1992.

[12] S. Gribble et al., Miller, "Self-Similarity in File Systems," In Proceedings of ACM Conference on Measurement and Modelling of Computer Systems (SIGMETRICS), Madison, WI, 1998, ACM, New York (1998), pp. 141-150.

[13] W. W. Hsu, "Dynamic Locality Improvement Techiques for Increasing Effective Storage Performance, Ph.D. thesis, University of California, Berkeley, CA (December 2002). Available as Technical Report CSD-03-1223, Computer Science Division University of California, Berkeley(January 2003).

[14] W. Hsu and A. Smith, "The Performance Impact of I/O Optimization on Disk Improvements," IBM Journal of Research and Development, 48(2):255-289, March 2004.

[15] M. Zhou et al, "Analysis of Personal Computer Workloads," Proceedings of the Seventh International symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 208-217, October 1999.

[16] P. Harrison and S. Zertal, "Calibration of a Queuing Model of RAID Systems," In Proceedings of Practical Application of Stochastic Models, Imperial College, London, September 2004.

Paul G. Sikalinda, born in 1977, obtained his BSc degree in computer science in 2002 in Zambia and is studying for MSc in computer science at the University of Cape Town (UCT) in South Africa. He worked as Database Administrator for the Zambia Revenue Authority before coming to UCT in 2004.