Appeared in the Proceedings of the Computer Measurement Group (CMG) Conference, December 1995, pp. 1263-1269.

Generating Representative Synthetic Workloads

An Unsolved Problem

Gregory R. Ganger*
M.I.T. Laboratory for Computer Science
http://www.pdos.lcs.mit.edu/~ganger
ganger@lcs.mit.edu

Abstract

Synthetic disk request traces are convenient and popular workloads for performance evaluation of storage subsystem designs and implementations. This paper develops an approach for validating synthetic disk request generators. Using this approach, commonly-used simplifying assumptions about workload characteristics (e.g., uniformly-distributed starting addresses and Poisson arrivals) are shown to be inappropriate, often leading to inaccurate performance predictions. Also, workload characteristics that require additional study are identified.

1 Introduction

Perhaps the largest difficulty facing storage architects and performance analysts is identifying and obtaining workloads with which to evaluate and compare designs. Many groups and organizations have a small set of traces/benchmarks, but few have a large set and fewer still share these all-important tools. The difficulty with obtaining representative workloads is pervasive and many researchers are forced to use whatever is available (commonly, ad hoc synthetic workloads with many simplifying assumptions). As a consequence, the open literature is riddled with studies that have no solid basis in reality, many of which have later (with better workloads) been shown to be incorrect.

An approach preferred by many storage subsystem researchers employs a record of real systems' storage activity in the form of **disk request traces**. A disk request trace represents an accurate account of what really happened in a particular storage subsystem during a given period of time. However, disk request traces have several limitations: (1) For non-technical reasons, it can be extremely difficult to convince system administrators to allow tracing. (2) Traces, which tend to be very large (e.g., the six traces used in this work require over 1.3 GB of compressed, on-line storage), must be stored on-line when they are to be used for experiments. (3) Each trace represents a single measure of behavior, making it difficult to establish statistical confidence in results. It can sometimes be difficult to distinguish between real performance characteristics of the system under test and anomalous behavior of the trace. (4) It is very difficult to isolate and/or modify specific workload characteristics (e.g., arrival rate or total accessed storage capacity) of a trace. (5) Traces do not support studies of expected future workloads, since one cannot trace what does not yet exist.

An alternative approach is to use a synthetic workload. Fundamentally, disk workload synthesis is a simple task. Based on values for a set of workload-characterizing statistics, one generates a series of disk requests. The values can be derived from expected behavior or measured from a sequence (e.g., a trace) of disk requests – hopefully in one pass, which then enables characterization based on observation rather than tracing. The difficult part of the task is to identify a set of statistics that accurately and uniquely characterizes a workload. Generally speaking, a disk request is defined by five values: device, operation (e.g., read or write), start location, size and arrival time. The first four values identify the access and the fifth identifies when it is presented to the disk subsystem. We refer to the sequence of accesses as the **access pattern** and the sequence of arrival times as the **arrival pattern**.

Disk workload synthesis has the potential to eliminate the limitations of disk request traces. For example, capturing workload characteristics rather than traces should be more palatable to system administrators (who are probably interested in such characteristics as well, for capacity planning and system tuning purposes). The synthetic traces can be manufactured on the fly rather than stored. By applying different seeds to the random number generator, one can generate many request streams with the same characteristics and thereby achieve some statistical confidence. Changes to the workload can be made much more readily and work-

*This work was partially funded by a CMG graduate fellowship. It was done jointly with Yale Patt at the University of Michigan and Richard Golding and John Wilkes at Hewlett-Packard Laboratories. While any credit should go to the collaborative, all errors are mine.

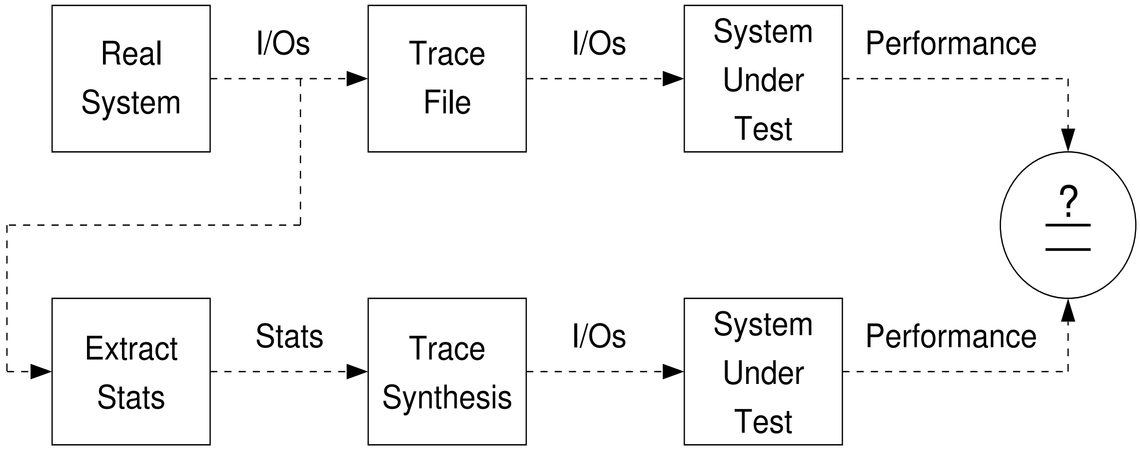

Figure 1: Calibration of disk request trace synthesis.

loads representing expected future environments can be generated.

The main disadvantage of trace synthesis is the danger of generating a trace that is a poor match to the original in some important way, thereby compromising the validity of the results. Borrowing from recent work in disk model calibration [Ruemmler94], we develop an approach to validating synthetic disk request trace generators. Using this approach, we show that commonly applied assumptions (e.g., uniform starting addresses, Poisson arrivals) are inappropriate and can produce dramatically incorrect results. We also identify specific areas where additional understanding of workload characteristics is needed.

The remainder of the paper is organized as follows. Section 2 describes our methodology for evaluating synthetic workload generators. To isolate specific issues with trace synthesis, we evaluate access pattern generation (Section 3) and arrival pattern generation (Section 4) separately. Section 5 concludes with a discussion of problems with standalone storage subsystem models and a possible source of representative synthetic workloads.

2 Evaluation Methodology

2.1 Calibration of Synthetic Workloads

To calibrate a synthetic workload generator, one can compare performance measurements of the system(s) of interest operating under a synthetically-generated workload and the real workload that it attempts to recreate. While performance-based comparison more appropriate than other approaches [Ferrari84], it limits the applicability of calibration results to those systems-under-test (SUTs) used. As a result, workload characteristics that are important in other SUTs can be overlooked. For example, this work examines disk workloads on an individual disk-by-disk basis, ignoring correlations among the workloads observed by the different disks. Such correlations might be important for evaluating management approaches (e.g., load balancing and striping) for multi-disk subsystems.

Our specific methodology is illustrated in figure 1. Statistics are collected from a single pass over a timed sequence of disk requests, as generated by the system software above the device driver. These statistics are fed into a trace synthesis tool, which produces a sequence of requests with the same characteristics (including length in time). Assuming that the synthesis tool has no errors, this will be successful if the statistics capture all important information. Both request sequences (traced and synthetic) are used, in turn, to exercise a SUT (the disk simulator described below, for this paper). Performance measurements from the resulting SUT activity are used to evaluate the closeness of the synthetic trace.

While many metrics can be used, we try to encapsulate a number of them in a single value. As in [Ruemmler94], we plot the response time distributions for the two request sequences (traced and synthetic) and use the root-mean-squared (RMS) horizontal distance between the two curves as a metric, which we call the **total error**. Lower values indicate better matches. Many synthetic traces can be generated from a single set of statistic values by simply changing the seed value for the random number generator. So, we also measure the **randomness error**, which is the average RMS distance between response time distributions for the "mean" of the synthetic traces and any one of them. The "mean" is computed by combining multiple distributions (i.e., summing the bin values). Our main metric, the **synthesis error** for a trace generation approach is the amount by which the total error exceeds the randomness error (i.e., max(0, total – randomness)). We view this as reasonable, because if the total error is less than the randomness error, one could claim that they are statistically indistinguishable.

Trace Name	Length (hours)	# of Disks	# of Requests	Average Size	Percent Reads	Percent Seq. Reads
Cello	168	8	3,262,824	6.3 KB	46.0%	2.5%
Snake	168	3	1,133,663	6.7 KB	52.4%	18.6%
Air-Rsv	9	16	2,106,704	5.1 KB	79.3%	7.8%
Sci-TS	19.6	43	5,187,693	2.4 KB	81.5%	13.8%
Order	12	22	12,236,433	3.1 KB	86.2%	7.5%
Report	8	22	8,679,057	3.9 KB	88.6%	3.8%

Table 1: Basic characteristics of the disk request traces.

2.2 Disk Request Traces

For our experiments, we use six traces of disk activity collected from systems in use at various industrial and research installations. We describe the traces only briefly as they have been described elsewhere in more detail [Ramakrishnan92, Ruemmler93]. The traced workloads span a range of user environments, and each is at least a full workshift (8 hours) in length. Some basic characteristics of the traces are given in table 1. They vary widely in read/write ratios, access sizes, arrival rates, locality and burstiness.

Two of the traces come from Hewlett-Packard systems running HP-UX^{TM}, a version of the UNIX^{TM} operating system [Ruemmler93]. *Cello* comes from a server at HP Labs used for program development, simulation, mail, and news. *Snake* is from a file server at the University of California at Berkeley used primarily for compilation and editing. While these traces are actually two months in length, we report data for a single week-long snapshot (5/30/92 to 6/6/92).

The other four traces are from commercial VAX^{TM} systems running the VMS^{TM} operating system [Ramakrishnan92]. *Air-Rsv* is from a transaction processing environment in which approximately 500 travel agents made airline and hotel reservations. *Sci-TS* is from a scientific time-sharing environment in which analytic modeling software and graphical and statistical packages were used. *Order* and *Report* are from a machine parts distribution company. *Order* was collected during daytime hours, representing an order entry and processing workload. *Report* was collected at night, capturing a batch environment in which reports of the day's activities were generated.

2.3 System Under Test

We use a detailed, validated disk simulator for this work. The simulator accurately models zoned recording, spare regions, defect slipping and reallocation, disk buffers and caches, various prefetch algorithms, fast write, bus delays, and control and communication overheads. For this work, the simulator was configured to model the HP C2240 series of disk drives [HP92] using the parameters described in the appendix of [Worthington94]. Some default specifications for the HP C2247 drive (the 7-platter member of the C2240 line) are listed in table 2.

HP C2247 Disk Drive	
Formatted Capacity	1.05 GB
Rotation Speed	5400 RPM
Data Surfaces	13
Cylinders	2051
512-Byte Sectors	2054864
Zones	8
Sectors/Track	56-96
Interface	SCSI-2
256 KB Cache, 2 Segments	
Track Sparing/Reallocation	

Table 2: Default characteristics of the HP C2247.

The simulator was validated by exercising an actual HP C2247 and capturing traces of all SCSI activity. Each traced request stream was run through the simulator, using the observed request inter-arrival delays. This process was repeated for several synthetic workloads with varying read/write ratios, arrival rates, request sizes, and degrees of sequentiality and locality. The average response times of the actual disk and the simulator match to within 0.8% in all cases. Unpredictable (from the disk's view) host delays partially account for the difference. Greater insight can be achieved by comparing the response time distributions [Ruemmler94]. Figure 2 shows distributions of measured and simulated response times for a sample validation workload of 10,000 requests. As with most of our validation results, one can barely see that two curves are present. The root mean square horizontal distance between the two distribution curves for the validation run shown in figure 2 is 0.07 ms, or less than 0.5% of the average response time. The largest value observed over all validation runs was only 1.9% of the corresponding average response time.

We simulate members of the HP C2240 series of drives because our simulator correctly models them and because we do not have the necessary information for the

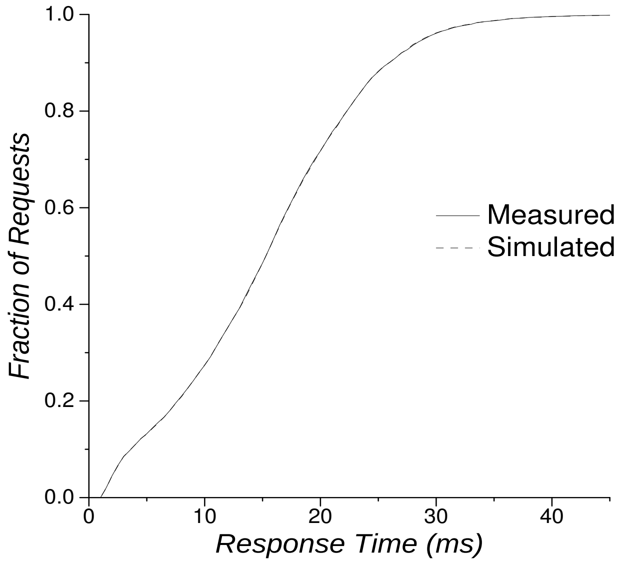

Figure 2: Measured and simulated response time distributions for an HP C2247A disk drive. The validation workload parameters are 50% reads, 30% sequential, 30% local [normal with 10000 sector variance], 8KB mean request size [exponential], interarrival time [uniform 0–22 ms].

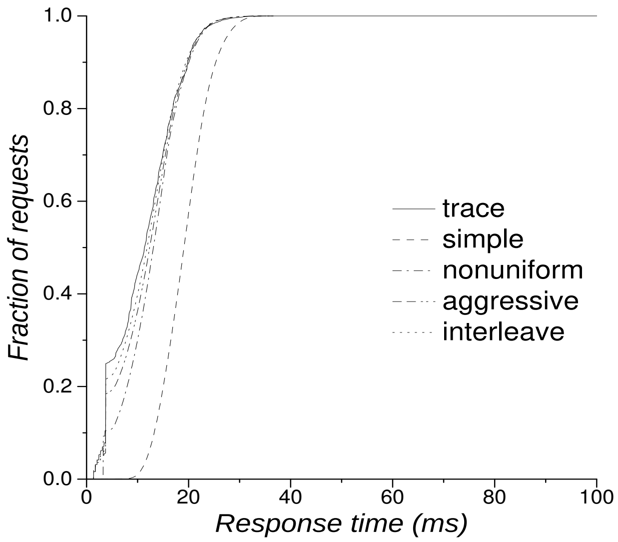

Figure 3: Access pattern synthesis for *Snake*.

Access Generation	Average Resp. time	Total error	Rand. error	Synth. Error
simple	19.2 (4.6)	8.2	0.1	8.2
nonuniform	12.9 (5.5)	2.2	0.5	1.7
aggressive	12.0 (6.0)	1.3	0.9	0.4
interleave	11.5 (6.1)	0.9	0.9	0.1

Table 3: Access pattern synthesis for *Snake*. The average response time for the trace is 11.3 ms (6.4).

disks used on the traced systems. This disk substitution raises a difficulty because our base disk (the HP C2247) has a different storage capacity than the disks used by the traced systems. To better match the size of the simulated disks to those in the traced systems, we modify the number of platters (5 for *Sci-TS*, *Order*, and *Report*; and 9 for *Cello*, *Snake*, and *Air-Rsv*) to create disks large enough to contain the active data without excessive unused media. We feel that this is a reasonable approach, as a production line of disks often differs only in the number of physical platters.

3 Access Patterns

To isolate the effects of the access pattern from those of the arrival pattern, we replace all inter-arrival times with a constant value of 10 seconds. This eliminates all queueing and re-ordering effects. One impact of this approach is that the observed response times (and therefore the error values) are much smaller than when queueing occurs. So, the impact of small error values will, with more realistic arrival patterns, be magnified.

A single disk access is defined by four values: the operation type (e.g., read or write), the device number (obviated for single disk subsystems), the starting location, and the size. Various statistics can be used to characterize each of these values independently as well as inter-dependencies among them. This section evaluates several options, chosen to highlight specific issues in workload characterization/synthesis.

Figures 3–4 and tables 3–4 compare synthetic generation schemes, indicating how well they recreate observed access patterns. All response times are in milliseconds and values in parens are variances. The schemes shown are:

1. *trace*: the observed sequence of accesses.
2. *simple*: uses the measured read/write percentage and a constant (the measured average) access size. Starting locations are independent and uniformly distributed across device storage capacity.
3. *nonuniform*: same as *simple*, except that starting locations are generated from the measured distribution of distances from the last block accessed in the previous request to the start of the next.
4. *aggressive*: same as *nonuniform*, except that it uses a 2-state Markov model to decide whether read or write, with measured values for the four transitions. Also, uses a fraction of requests that have the same

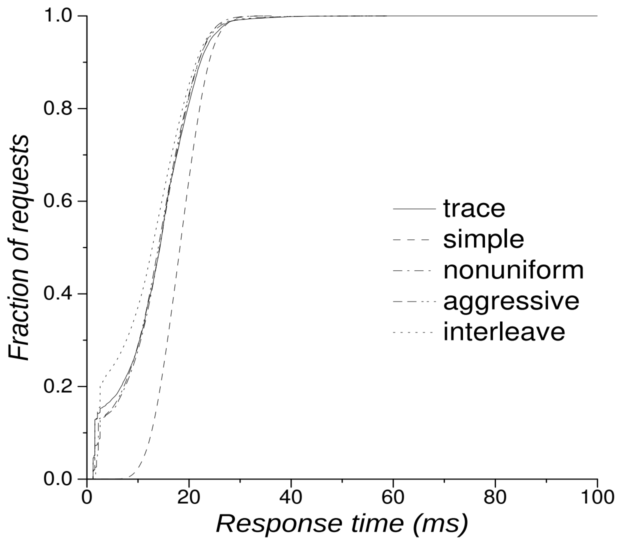

Figure 4: Access pattern synthesis for *Order*.

Access Generation	Average Resp. Time	Total error	Rand. error	Synth. Error
simple	18.3 (4.4)	5.9	0.03	5.9
nonuniform	13.6 (6.7)	1.9	0.04	1.9
aggressive	13.5 (6.9)	1.5	0.08	1.4
interleave	12.2 (7.4)	2.2	0.12	2.1

Table 4: Access pattern synthesis for *Order*. The average response time for the trace is 13.5 ms (7.3).

size as the previous request and a measured distribution for those that do not.

5. *interleave*: same as *aggressive*, except that it uses measured distributions of inter-access distances for eight "streams" to capture interleaved streams of localized accesses. The depth of these "interleave" distributions is 64 sectors (32 KB) in either direction from the prev-last-block. There is also a measured distribution for the requests that are not caught by the "interleaves".

For all six traces, the *simple* model results in large synthesis error values. As seen by comparing *simple* to *nonuniform*, much of this error is caused by the common assumption that request starting locations are independent and uniformly distributed across the disk. More aggressive statistics for the operation type and request size, as represented by *aggressive*, improve accuracy slightly. The addition of interleave locality statistics improves accuracy for several of the workloads (including *Snake*), indicating that access streams with locality are interleaved in these workloads. For *Order*, however, the use of interleave locality statistics reduces accuracy.

We were unable to achieve close matches for several of the workloads (e.g., *Order* and *Sci-TS*). We tried a number of approaches involving dependencies between statistics (e.g., separate locality statistics for reads and writes, separate locality statistics for reads following reads and reads following writes). Even with such complexity, the synthesis errors remain large. Much remains to be understood in the area of disk access patterns.

4 Arrival patterns

To isolate the effects of the arrival pattern from those of the access pattern, we use a constant per-request access time (12 ms). This eliminates all locality and caching effects. One impact of this is that access times during bursts are over-estimated in cases where they might otherwise be reduced (e.g., due to disk scheduling). As a result, queue times tend to be larger than is the case with real access patterns.

Request arrival patterns are affected by a number of system-specific details. Two important characteristics of arrival patterns are feedback and burstiness. Feedback between individual request response times and subsequent request arrivals can have a significant impact on performance [Ganger95]. However, it is not possible to extract feedback information from the disk request traces available for this work, so we do not pursue it. An arrival pattern is **bursty** if it exhibits interspersed periods of high activity (small inter-arrival times) and low activity (large inter-arrival time). More bursty workloads exhibit longer and/or more intense periods of high activity relative to the low activity periods. More quantifiable definitions and metrics for burstiness remain an important area for future workload characterization research.

Figures 5–6 and tables 5–6 compare synthetic generation schemes, indicating how well they recreate observed arrival patterns. All response times are in milliseconds and values in parentheses are variances. The schemes shown are:

1. *trace*: the observed arrival times (from logical zero).

2. *expon*: uses independent and exponentially distributed inter-arrival times, with the mean set to the measured average.

3. *actdist*: uses independent inter-arrival times taken from the measured distribution.

4. *2-dists*: uses inter-arrival times taken from one of two measured distributions, depending on the immediately prior inter-arrival time. The first is used when the previous inter-arrival time is less than 60 ms and the other when it is greater than 60 ms. We arrived at these values by examining the inter-arrival distribution and selecting a knee point.

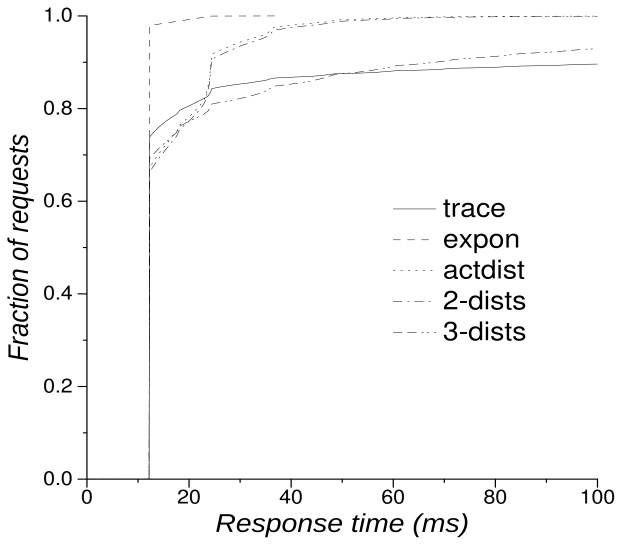

Figure 5: Arrival pattern synthesis for *Cello*.

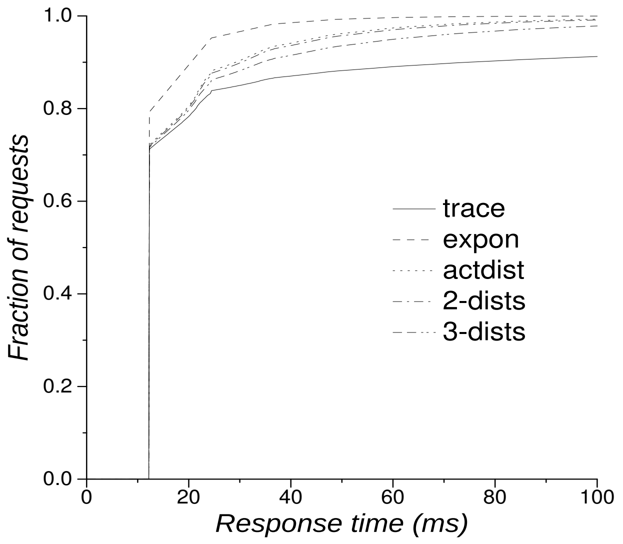

Figure 6: Arrival pattern synthesis for *Order*.

Arrival Generation	Average Resp. Time	Total Error	Rand. Error	Synth. Error
expon	12.3 (1.1)	455	0.11	455
actdist	16.1 (8.1)	449	1.1	447
2-dists	16.5 (8.9)	448	2.9	445
3-dists	34.5 (77.3)	378	10.7	367

Table 5: Arrival pattern synthesis for *Cello*. The average response time for the trace is 114 ms (436).

5. *3-dists*: uses inter-arrival times taken from one of three measured distributions, depending on the immediately prior inter-arrival time. One is used when the last inter-arrival time is less than 5 ms, one when between 5 ms and 60 ms, and one when greater than 60 ms.

None of the arrival pattern synthesis schemes explored recreate the observed behavior. In particular, they all fail to replicate the very long response times that characterize intense bursts of activity. *expon* results in the highest error, as an exponential distribution is a poor match for the observed inter-arrival times. The largest problem, however, appears to be the assumption of independence among arrival times. *2-dists* and *3-dists* reduce the synthesis error, but still do not come close to the observed behavior. Much work is needed before an understanding of disk request arrival patterns will be achieved. One possible solution, self-similarity, has been applied successfully to the characterization of network traffic [Leland93].

Arrival Generation	Average Resp. Time	Total Error	Rand. Error	Synth. Error
expon	14.4 (6.8)	548	0.9	547
actdist	17.7 (15.5)	542	1.8	539
2-dists	18.1 (16.8)	540	1.8	538
3-dists	20.4 (25.4)	534	3.0	530

Table 6: Arrival pattern synthesis for *Order*. The average response time for the trace is 77.6 ms (519).

5 Conclusions and Future Work

With this work, we have made a first step toward understanding how to characterize a disk workload and how to demonstrate the validity of a characterization. However, the large synthesis error values indicate that much work remains. We have been unable to accurately recreate the access and arrival patterns of the traces.

These results do not argue that accurate synthesis is impossible. However, they do show that commonly-used simplifying assumptions about workload characteristics are inappropriate and can lead to erroneous conclusions. In particular, request starting locations are not (in general) independent and uniformly distributed. Also, request arrival patterns do not match those generated by a Poisson process. In fact, they are neither independent nor exponentially distributed.

An accurate synthetic trace generator may not be the appropriate solution, because standalone I/O subsystem models are too narrow in scope [Ganger95]. They tend to treat all requests equally, ignoring differences in how individual requests affect system behavior. As a result, they ignore or over-simplify feedback effects between individual request response times and subsequent request arrivals. Also, they predict performance changes with I/O subsystem performance metrics, which do not (in general) correlate with changes in overall system performance. [Ganger95] demonstrates these problems and proposes a solution, system-level modeling.

A system-level approach may also be appropriate for workload synthesis. Rather than attempting to recreate the disk activity resulting from particular system activity, one could use the system activity itself to drive models (or instances) of the system. Two promising approaches to synthetic system benchmarks are the SPEC S-det benchmark [Gaede81, Gaede82] and SynRGen [Ebling94], a synthetic file reference generator.

Acknowledgements

I thank Richard Golding, Yale Patt, John Wilkes and Bruce Worthington for their direct involvement in the development of this work. This work was partially funded by a CMG graduate fellowship. Our storage subsystem research activity at the University of Michigan benefited greatly from the financial and technical support of several industrial partners, including AT&T/GIS, DEC, Hewlett-Packard, MTI and SES.

References

[Ebling94] M. Ebling, M. Satyanarayanan, "SynRGen: An Extensible File Reference Generator", *ACM SIGMETRICS Conference*, May 1994, pp. 108–117.

[Ferrari84] D. Ferrari, "On the Foundation of Artificial Workload Design", *ACM SIGMETRICS Conference*, May 1984, pp. 8-14.

[Gaede81] S. Gaede, "Tools for Research in Computer Workload Characterization", *Experimental Computer Performance and Evaluation*, 1981, ed. by D. Ferrari and M. Spadoni.

[Gaede82] S. Gaede, "A Scaling Technique for Comparing Interactive System Capacities", *13th International Conference on Management and Performance Evaluation of Computer Systems*, 1982, pp. 62–67.

[Ganger95] G. Ganger, "System-Orented Evaluation of I/O Subsystem Performance", Ph.D. Dissertation, University of Michigan, Ann Arbor, 1995.

[HP92] Hewlett-Packard Company, "HP C2244/45/46/47 3.5-inch SCSI-2 Disk Drive Technical Reference Manual", Part Number 5960-8346, Edition 3, September 1992.

[Leland93] W. Leland, M. Taqqu, W. Willinger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *ACM SIGCOMM Conference*, 1993, pp. 183–193.

[Ramakrishnan92] K. Ramakrishnan, P. Biswas, R. Karelda, "Analysis of File I/O Traces in Commercial Computing Environments", *ACM SIGMETRICS Conference*, 1992, pp. 78–90.

[Ruemmler93] C. Ruemmler, J. Wilkes, "UNIX Disk Access Patterns", *Winter USENIX Conference*, January 1993, pp. 405–420.

[Ruemmler94] C. Ruemmler, J. Wilkes, "An Introduction to Disk Drive Modeling", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 17–28.

[Worthington94] B. Worthington, G. Ganger, Y. Patt, "Scheduling Algorithms for Modern Disk Drives", *ACM SIGMETRICS Conference*, May 1994, pp. 241–251. Full version: "Scheduling for Modern Disk Drives and Non-Random Workloads", Report CSE-TR-194-94, University of Michigan, Ann Arbor, March 1994.