

Characterization of Storage Workload Traces from Production Windows Servers

Swaroop Kavalanekar, Bruce Worthington, Qi Zhang, Vishal Sharda

The scarcity of publicly available storage workload traces of production servers impairs characterization, modeling research, and development efforts across the storage industry. Twelve sets of storage traces from a diverse set of Microsoft Corporation production servers were captured using ETW (Event Tracing for Windows) instrumentation. Windows Server 2008 dramatically increases the breadth and depth of ETW instrumentation, and new trace capture and visualization tools are available in the Windows Performance Tools kit. Additional analytical tools were developed to analyze and visualize traces captured from Exchange, software build and release, Live Maps, MSN storage, security authentication, and display advertisement platform servers. This paper contains a first set of characterizations for these traces, including simple block-level statistics, multi-parameter distributions, rankings of file access frequencies, and more complex analyses such as temporal and spatial self-similarity measurements. Trace data visualizations enable the examination of workload parameters, subcomponents, phases, and deviations from predicted behavior.

I. INTRODUCTION

OBTAINING comprehensive traces of storage activity on a wide range of production servers is, to say the very least, a challenging undertaking. Making the traces publicly available increases the difficulty by an order of magnitude. The scarcity of such traces over the last 15+ years attests to the obstacles that must be overcome to achieve this goal.

Starting in late 2007, Microsoft began tracing the storage workloads of a variety of its corporate production servers specifically to support internal and external researchers and developers of storage hardware and software. Windows Server can selectively enable extensive instrumentation via Windows command-line utilities or WMI (Windows Management Infrastructure) calls.

Initial samples of production server storage traces are being publicly distributed via the online trace repository provided by SNIA (Storage Networking Industry Association) [10]. While the initial production traces analyzed in this report are 5-24 hours in duration and include primarily file and disk I/O events, the end goal is to make longer and more comprehensive traces publicly available as the process for automated capture, verification, analysis, confidential string obfuscation, and distribution of such traces stabilizes.

A variety of statistical parameters are extracted and tabulated for twelve unique trace sets. These basic

parameters are useful for identifying a few general workload behaviors and for selecting specific traces for deeper analysis. Several visualizations of individual and combinations of characteristics are presented. Complex interrelationships such as self-similarity are explored as well.

Storage traces provide insights on design and implementation tradeoffs of file systems, file and block caches, storage drivers, storage controller firmware, and the various hardware components of a modern server storage subsystem. For example, the use of nonvolatile memory (NVM) in the server storage stack is receiving significant attention now that enterprise-quality Flash memory components have dropped in price, increased in per-unit capacity, improved in reliability, and addressed the random write performance penalty traditionally associated with the technology. Comprehensive storage traces of production servers will be crucial in the determination of what forms and configurations of NVM are appropriate for different workloads. Even application developers can take advantage of storage traces of specific workload scenarios in order to improve the manner in which I/Os are issued.

The remaining sections in this paper are as follows: Section II provides background and related work. Section III describes the new traces analyzed in this report. Section IV lists a wide variety of parameters and metrics associated with a server storage workload. It also includes a tabular overview of the new traces' basic characteristics. Section V provides a deeper look into some of the behavior captured in the traces. Section VI extends this analysis by demonstrating how some of these storage workloads exhibit self-similarity both in time and in space. Section VII provides a summary and some directions for future work.

II. BACKGROUND AND RELATED WORK

A. Traces vs. Synthetic Workloads

Typical uses of server storage traces can be divided into three broad areas: analysis of the characteristics and behaviors of the specific systems or workloads traced; input for trace-driven simulation or modeling; and extraction of parameters and heuristics to configure synthetic workload models. In the latter two usages, there are purposes beyond the context of the traced environment.

While trace-driven simulation of storage subsystems has a number of limitations, the design and use of synthetic workload models is still largely dependent on the availability of traces from which characteristics can be selected,

Authors are with Microsoft Corporation, Redmond, WA 98052 USA (e-mail: swaroopk, bworth, qizha, vsharda @microsoft.com).

extracted, and subsequently generalized. Unfortunately, storage traces are often difficult to characterize due to the complexity of interactions between multiple request streams and the uncertainty of whether or not they are truly representative of environments similar to the ones traced. It is also difficult to prove that even the most detailed synthetic models capture all of the important properties of a given set of traces. Nevertheless, there are a number of advantages of utilizing synthetic models for analysis instead of the server storage workload traces themselves:

- Synthetic models have the flexibility to observe behaviors with alternative parameters or heuristics
- Large sets of comprehensive traces are difficult to obtain due to security and performance impact concerns
- Even when compressed, lengthy storage traces require significant on-line or archival storage

So, while a trace is an accurate representation of a specific set of activities on a specific server and thus can be used to accurately drive trace-driven models, the ideal purpose is the transformation of a set of representative traces into a set of characteristics that enable modeling and simulation across a wider spectrum of system environments.

B. Capturing and sharing server storage workload traces

The ideal storage workload trace methodology includes all of the features given below.

- Administrative access to production servers
- Minimum overhead imposed during trace capture and zero overhead when tracing is disabled
- Coverage of all phases of activity and all data sets across multiple equivalent platforms for months/years
- Sufficient trace event types captured to supply all future analyses (impossible, so capture as much as feasible without adversely impacting other features)
- Correlated file block and disk block events
- Full disclosure of software, hardware, and firmware configuration information during the trace period
- Self-describing portable trace event formats
- High compression factor for trace files
- Selective obfuscation of trace event elements (e.g., file or process names)
- Selective trace event element decryption to maintain a single trace format [2]

Since this combination of features cannot be simultaneously realized, the inherent tradeoffs must be carefully evaluated.

C. Characterizing server storage workload traces

As is demonstrated in Section IV, a long list of characteristics can be extracted from a storage workload trace. The interaction between different basic characteristics must also be identified and prioritized by overall impact [4] [12] [13] [14] [19] [27]. If the trace characterization is driving an open model, the observed response times of the traced system may or may not be utilized. Similarly, if the model is closed, then arrival times may or may not be

utilized in favor of maintaining a certain level of I/O concurrency [30].

The possible existence of self-similarity in real-world disk traffic was first proposed in [4]. Using two disk-block IO traces, self-similarity analyses in disk arrival patterns and disk access patterns were provided in [5]-[7]. A combination of the ON/OFF source model and Cox's model described self-similar behavior in these two dimensions. Binomial multifractals have been used to generate synthesized IO workloads that display self-similarity [8]. Entropy plots can quantify the spatial and temporal correlation of disk requests, and an efficient entropy-based model was proposed to capture all the characteristics of real spatio-temporal traffic [29]. All these related works demonstrate the existence of self-similarity in storage IO traces, which must therefore be considered in analytic modeling to capture accurate storage performance.

D. Storage workload tracing facilities

Event Tracing for Windows (ETW) has been the core tracing component built in to Windows operating systems since Windows 2000. ETW provides a high performance, low overhead, and highly scalable tracing framework. It uses efficient buffering and non-blocking logging mechanisms with per-CPU buffers written to stable storage by a separate thread. Windows Server 2003 and 2008 releases allow tracing to be enabled and disabled dynamically without requiring system reboots or application restarts. Many Windows components, including the kernel, produce numerous events describing their behavior. Typical events are discrete time-stamped trace points, but sampling and statistical data captures are also possible. Storage related instrumentation includes, but is not limited to: initiation and completion disk events for reads, writes, and flushes; and file events for creates, deletes, reads, writes, and attribute queries and updates.

One predecessor of ETW is *VTrace*, which provides kernel event instrumentation through patching and hooking Windows NT and Windows 2000 by undocumented and unsupported means [15]. Other operating systems also provide embedded support for storage workload tracing, such as the "io" provider in Solaris' *DTrace* [9], the Veritas *vxtace* utility in HP-UX [28], the *Tracefs* file system filter for Linux [2], and the disk I/O workload characterization tool in VMware [1].

E. Published server storage block-level traces

Various server storage block-level traces gathered and published in the last 15 years are listed below.

- 1992: 4 months of disk block traces from the *Snake* file server at U. C. Berkeley serving nine clients without local storage [20]
- 1992: 2 months of disk block traces from the *Cello* shared compute/mail server used by a HP Labs research team [20]
- 1996: 3 months of disk block traces from the same *Cello* shared compute/mail server

- 1999: 1 year of disk block traces from the same *Cello* shared compute/mail server [31]
- 2000: 1 hour of disk block traces from an *OpenMail* mail server for 1400 users at HP Labs [12]
- 2000: 1 month of file block traces from a web server (*WEB*) for an online library at U. C. Berkeley [19]

Note that this list does not include storage traces of scientific workloads or traces without block-level events, both of which are beyond the scope of this paper.

Storage traces of well-understood benchmarks are useful to verify tracing, postprocessing, modeling, and simulation infrastructures. Since benchmarks are predefined models of steady-state workload behavior, extracting workload characteristics from traces of benchmarks may be easier than extraction of production workload characteristics, albeit of less value. Some benchmarks used for server storage analysis are given below.

- TPC-C [14] [21] [29]
- TPC-D [11] [12] [22]
- TPC-H [14] [17] [24]
- FileBench [1] [16]
- DBT-2 [1] [3]
- Postmark [2]
- AM-Utils [2] [18]

Conclusions about real-world behavior drawn from analyses of benchmark traces are only as accurate as the benchmarks' accuracy in representing real-world environments.

III. SERVER STORAGE WORKLOAD TRACES

This section describes twelve of the initial production server traces and two database benchmark traces. The traces referenced in this report consist of ETW events enabled in the Windows kernel. A post-processing script library was developed to extract the reported workload characteristics and metrics.

The Windows Performance Tools kit (WPT) is an extensible performance analysis toolset that provides high-level control and decoding of ETW events. It includes a comprehensive visualization tool for detailed analyses of a wide range of system activities. This tool provides powerful interactive summary tables and graphs with dynamic grouping, sorting, and aggregation capabilities.

The traces analyzed in this paper are broken into intervals to reduce the size of individual traces and make analysis and visualization easier. The duration of the interval is adjusted based on the storage activity of the workload to keep the size of the individual traces manageable.

A. Live Maps front-end and back-end servers (*LM*)

Virtual Earth is a feature of Live Maps that displays satellite images and photographs of locations. The tile front-end server (*TFE*) takes a user request for a location and passes it to the tile back-end server (*TBE*). The TBE hosts a portion of the map imagery. It accesses the image tiles from a disk and sends them to the TFE, which then mashes up the image by adding routes, markers and other relevant

information and sends it back to the user. The traces from the TFE and TBE cover a 24-hour period and are broken into 1-hour intervals.

B. Display Ads Platform data and payload servers (*DAP*)

The purpose of the data server (*DS*) is to be a caching tier between the front-end server and the payload server (*PS*). A front-end server makes an advertisement request with a user id to the DS. The DS looks up the user id in the cache, appends any information available for that user to the request, and passes the request to the PS. The PS is responsible for ad selection. The traces from the DS and PS cover a 24-hour period and are broken into 30-minute intervals.

C. Exchange server (*Exch*)

The Microsoft Exchange 2007 SP1 server is a mail server for 5000 corporate users. It is a 4-socket, dual-core system with 4 GB of memory. The storage consists of two 146 GB SAS drives in a RAID-1 configuration, six data arrays of fourteen 146 GB SAS drives, and two log arrays of eight 146 GB SAS drives configured as RAID-10. One trace covers a **5-hour** peak load period on a weekday afternoon. Another trace covers a **24-hour** weekday period. The traces are broken into 15-minute intervals.

D. MSN storage metadata and file servers (*MSN*)

The *CFS* server stores metadata information and blobs correlating users to files stored on the back-end file server (*BEFS*). The BEFS provides the files requested by CFS. The servers are used by several Live data services. The traces from the CFS and BEFS cover a 6-hour period and are broken into 10-minute intervals.

E. Windows build server (*WBS*)

The WBS produces a complete build each day for the 32-bit version of the Windows Server operating system. It is a 2-socket quad-core system with 8 GB of memory. The storage consists of eight 146 GB disks configured as RAID-0. To capture the complete build process as well as any disk activity during idle periods, the trace covers a 24-hour period and is broken into 15-minute intervals.

F. Developer tools release server (*DTRS*)

The DTRS is a file server accessed by more than 3000 users to download various daily builds of Microsoft Visual Studio (copied from dedicated build servers). It is a 2-socket single-core system with 2 GB of memory. The storage consists of a single Vdisk of 40 GB configured as RAID-10 storage. The Vdisk is part of a 219-disk SAN. The traces from the DTRS cover a 24-hour period and are broken into 1-hour intervals.

G. RADIUS authentication and back-end servers (*RAD*)

The RADIUS authentication server (*AS*) is responsible for worldwide corporate remote access and wireless authentication. It runs the IPSec NAP scenario. Data comes in via SQL replication on the back-end SQL server (*BE*).

The traces from the AS and BE cover an 18-hour period and are broken into 1-hour intervals.

H. Database benchmarks: *TPC-C* and *TPC-E*

TPC-C is an online transaction processing (OLTP) benchmark simulating an order-entry environment [21]. It is a mix of five concurrent transactions of different complexities. TPC-E is the successor of the TPC-C benchmark and simulates the workload of a brokerage firm [23]. TPC-E transactions are more complex than those of TPC-C, and they more closely resemble modern OLTP transactions. TPC-E has lower storage throughput requirements than TPC-C. The TPC-C trace covers 5 minutes of a steady state, fully scaled workload running on a 4-socket, dual-core system with 64 GB of memory. The storage consists of 14 RAID-0 disk arrays of 28 disks each. The TPC-E trace covers 10 minutes of a steady state, fully scaled workload running on a 4-socket quad-core system with 128 GB of memory. The storage consists of 12 RAID-0 disk arrays of 28 disks each.

IV. STORAGE WORKLOAD METRICS AND PARAMETERS

For the majority of storage metrics and parameters, computing the means and variances is insufficient to characterize a trace accurately. Histograms typically provide a good representation as long as they are sufficiently fine-grained. Analysis of specific data points may reveal standardized distributions (e.g., normal, gamma, and exponential) that provide a good match for standard storage workload parameters such as: read/write ratio, request size and alignment, interarrival rate, response time, concurrency (e.g., queue depth), disk number, and file, partition, or disk offset (Logical Block Number). Previous traces indicate heavy-tailed distributions for a range of spatial and temporal parameters [6] [7].

Spatial locality and sequentiality refer to the “distance” between blocks referenced in a specific window of time or in a specific window of requests. Spatially adjacent requests in the trace can be extracted from the stream of IOs targeting each disk, so the percent of sequential requests and the run lengths of sequential bursts are easily computed along with the logical distances (jumps) between sequential streams. Multiple concurrent streams of activity or streams with “holes” complicate the extraction and representation of sequentiality [11]. One way to detect mixed or fragmented streams is by comparing each request offset not only against the immediately previous request offset and length, but also to a window of N previous requests [1]. More complex algorithms may reconstruct individual streams and allow the formulation of per-stream models of behavior with their own basic storage workload parameter distributions and even cross-stream interactions and dependencies [30].

Temporal locality refers to length of time between accesses to the same blocks of data. Average block lifetimes are visible as overwrites at the disk level and as deletions, truncations, and overwrites at the file level [19].

The vast majority of server storage workloads are not steady state but rather fluctuate in intensity and behavior throughout the day, week, month, and year. Accurate workload characterization means generating different parameter sets and values for different workload phases. Burstiness metrics for smaller time scales reflect fluctuating interarrival times between requests, thereby creating bursts of activity interspersed with idle periods.

Analysis of file block usage can identify correlations (or the lack of correlation) between all of the above metrics and file-specific data such as file size, file lifetime, file attributes (e.g., write-through, sequential, or temporary) and file type (e.g., based on filename suffixes or the names of the containing directories). Metadata traffic can also be separated out at the file event level.















While the disk block events in the initial traces do contain filenames, the relationships between file blocks and disk blocks are not easily determined except for the case of hard page faults. Still, the event data is sufficient to extract workload information about the “heat” of various file sets, the identification of files that are read-only/mostly or write-only/mostly, the mixture of file types, and any other correlations that do not require knowledge of file offsets.

Table 1 gives a set of basic metrics extracted from the new traces. This table provides a high-level overview to aid in trace selection for specific analysis. As mentioned at the beginning of this section, statistical averages are insufficient to characterize most real-world traces, but they can be helpful in selecting traces of potential interest.

Explanations for some of the metrics are given below.

- *IO Rate* shows a miniature graph (Sparkline [25]) for each workload’s IO throughput as it varies over the duration of the trace
- *Average queue length on initiation* is taken using samples at each request arrival time – not as an average over time
- *Average system interarrival* refers to the interarrivals of all requests regardless of disk
- *Average disk interarrival* refers to interarrival deltas between requests to the same disk
- *% sequential IOs initiated* includes only those requests that are exactly sequential (no “holes”) and uninterrupted by nonsequential requests
- *Sequential run length* metrics describe the uninterrupted streams of exactly sequential requests
- *Modes* are the most common values in a set (as opposed to mean or median values)
- *80th percentile* metric modifier indicates how many unique (“hot”) values constitute 80% of the total set
- *N/A* indicates that data was unavailable at the time this paper was written

TABLE I
WORKLOAD STATISTICS FOR PRODUCTION AND BENCHMARK STORAGE SERVER TRACES

Trace Statistics		LM-TFE	LM-TBE	DAP-DS	DAP-PS	Exch-5	Exch-24	MSN-CFS	MSN-BEFS	WBS	DTRS	RAD-AS	RAD-BE	TPC-C	TPC-E
Duration (Hours)		24	24	24	24	5	24	6	6	24	24	18	18	0:05	0:10
IO Rate (typically midnight to midnight)															
Avg IO/s		3.956	517.4	17.7	12.6	133.4	628.2	207.3	1351	145.9	205.9	40.2	84.4	50255	25432
	R	0.015	408.1	16.0	7.1	36.6	234.1	153	908	74.0	137.5	4.1	15.1	32337	23358
	W	3.941	109.3	1.7	5.5	96.8	394.1	54.3	443	71.8	68.4	36.1	69.3	17918	2074
Total IOs (Millions)		0.342	44.77	1.53	1.09	11.54	54.16	4.48	29.35	12.63	17.16	1.99	4.19	15.08	15.26
	R	0.001	35.31	1.39	0.61	3.17	20.18	3.30	19.73	6.41	11.46	0.21	0.75	9.70	14.01
	W	0.341	9.46	0.15	0.48	8.38	33.98	1.17	9.62	6.22	5.70	1.78	3.44	5.38	1.24
Total GB Transferred		8.92	2344	42.63	80.3	159.3	696.9	41.4	303.2	330.6	409.9	16.1	113.7	122.6	122.0
	R	0.01	1786	41.65	36.2	53.6	250.4	27.3	200.9	155.8	235.4	2.0	75.7	74.8	107.0
	W	8.90	558	0.98	44.1	105.7	446.5	14.1	102.3	174.8	174.5	14.1	38.0	47.9	15.0
Avg Req Size (KB)		27.32	54.91	29.17	77.47	14.47	13.49	9.71	10.83	27.44	25.05	8.47	28.5	8.53	8.38
	R	11.55	53.04	31.50	62.13	17.75	13.01	8.67	10.68	25.49	21.54	10.29	106.0	8.08	8.01
	W	27.37	61.90	7.04	97.13	13.23	13.78	12.62	11.15	29.46	32.12	8.26	11.6	9.34	12.63
Req Size Modes (KB)		4, 64	64	32	64	8, 4	8	4	8	4, 32	4, 64	4, 4.5	8, 120	8	8
	R	4	64	32	64	8	8	4	8	4, 32	4, 64	4, 5	120	8	8
	W	4, 64	64	4	4, 64	4, 8	4, 8	4	8	4, 32	64, 4	4, 4.5	8, 4	8	8
Avg Q Length on Initiation		7.123	1.94	0.082	0.043	5.86	97.61	3.38	128.0	3.09	28.58	0.13	3.12	225.4	140.6
	R	0.075	0.51	0.074	0.005	2.73	42.59	2.48	94.6	3.60	15.15	0.15	0.41	213.4	138.5
	W	7.149	5.87	0.065	0.083	5.86	66.14	3.51	152.0	0.43	18.47	0.09	3.48	14.6	3.1
Avg Response Time (ms)		11.32	1.96	3.42	0.77	1.79	3.17	12.09	15.04	3.21	10.12	0.95	6.01	4.38	5.35
	R	2.93	1.73	3.76	0.84	5.31	6.20	15.62	17.11	6.11	4.64	6.99	5.09	6.47	5.77
	W	11.36	2.85	0.18	0.68	0.45	1.37	2.14	10.78	0.22	21.13	0.26	6.22	0.61	0.52
Avg System InterArriv (ms)		253	1.93	56.5	79.6	15.69	1.59	4.86	0.74	6.88	N/A	24.84	11.85	0.02	0.04
	R	66757	2.45	62.4	141.5	51.37	4.27	6.58	1.10	13.55	7.32	240.8	66.26	0.03	0.04
	W	254	9.15	594.1	181.8	21.66	2.54	18.53	2.26	13.96	14.72	27.70	14.42	0.06	0.49
Avg Disk InterArriv (ms)		505	3.87	56.5	79.6	N/A	35.1	43.7	4.31	6.88	N/A	24.8	79.0	0.38	0.56
	R	66757	4.90	62.4	141.5	N/A	43.8	59.2	5.40	13.55	109.9	240.8	437.5	0.55	0.56
	W	507	18.30	594.1	181.8	N/A	54.5	166.8	13.16	13.96	N/A	27.7	96.1	0.98	7.34
% Seq IOs Initiated		32.62	70.43	0.77	61.65	N/A	16.12	8.30	4.35	7.79	21.80	47.78	58.19	2.10	1.55
	R	13.99	66.47	0.51	97.41	N/A	4.76	7.81	1.93	13.09	20.13	3.63	11.77	0.001	0.034
	W	32.70	95.25	3.35	16.52	N/A	23.21	10.66	9.69	3.87	32.86	55.82	70.72	6.08	18.75
Mean Seq Run Length (IOs)		4.62	11.2	2.69	17.09	3.35	3.28	2.76	3.00	3.53	4.01	7.91	21.47	46.04	32.99
	R	3.93	11.4	3.03	131.6	4.52	3.38	2.46	2.79	3.64	3.60	2.46	3.24	2.35	2.19
	W	4.63	182.8	2.37	3.23	3.24	3.30	4.35	3.17	3.42	6.78	12.83	43.91	1235	84.06
Run Length Modes (IOs)		2, 3, 5	2, 3, 8	2	2, 3, 4	2, 4, 3	2, 4	2	2	2	2	4, 2, 8	2	2, 3	2
	R	2, 3, 4	2, 3, 4	2	2	2	2	2	2	2	2	2	2	2	2
	W	2, 3, 5	2, 3, 8	2	2, 3, 4	2, 4, 3	2, 4	2	2	2	2, 8	4, 2, 8	2	2, 3, 4	2
Mean Seq Run Length (KB)		241.4	648.6	61.9	1056	54.7	53.6	87.5	68.4	172.2	377.4	41.6	409.3	1083	862
	R	92.8	635.8	81.6	8294	186.5	133.9	60.5	78.9	334.9	253.8	74.1	628.2	55.6	19.5
	W	241.7	11723	42.8	178	40.0	45.5	219.0	65.9	474.1	837.7	64.4	446.9	29312	2313
Run Length Modes (KB)		8, 128	128	8	128, 8	8, 1	8	68, 8	16, 1	8, 128	256	15	240, 1	1024	16
	R	8, 12	128	80, 12	80, 12	16, 24	16	68	16	8, 128	16	8, 128	240	16, 64	16
	W	8, 128	256	8	128, 8	8, 1	8	8, 68	1, 120	8	1024	15	1, 4, 8	1024	120
Unique Files Accessed		818	15097	3647	4338	40428	138K	958K	331	1.1M	N/A	20586	3818	220	199
	R	99	11938	3450	3962	34275	124K	654K	165	726K	N/A	20190	2694	178	169
	W	733	4450	252	617	7020	16456	438K	270	793K	N/A	16949	2661	45	37
Hot Files (80% of total reqs)		16	219	1	191	87	72	180K	14	83463	N/A	18	2	1	1
	R	19	184	1	156	38	34	104K	9	62914	N/A	5311	3	1	1
	W	15	164	8	38	91	89	203K	21	64502	N/A	9	2	1	1

V. TRACE ANALYSIS

This section provides an analysis of selected workload characteristics for a subset of the described traces in order to give a flavor of some useful ways to visualize the data and identify interesting storage workload characteristics. Due to space limitations, only the most interesting figures for each workload are presented in this paper.

A. Build and release servers (WBS and DTRS)

Fig. 1 shows the sorted distribution of IO request sizes observed on the WBS while building Windows Server. Due to the large number of small files accessed, the most common request size is 4 KB, the default cluster size of the NTFS file system. Over 3.5 million requests (29% of the total) are for 4 KB, split evenly between reads and writes. Similarly, the DTRS has 42% of its requests accessing 4 KB (not shown).

Spatial locality is a good indicator of how much disk actuator movement is required between servicing individual requests. It can be represented as the *offset* of the first block of a request relative to the last block of the most previous request to the same disk. Purely sequential requests have an offset of zero in the context of this analysis.

Fig. 2 shows the short-distance spatial locality distribution while building Windows. 4% of write requests and 13% of read requests (value not graphed) are exactly sequential. Short distance “jumps” between temporally adjacent requests occur most often for 4 KB multiples up to 40 KB, both forward and backward. Small backward jumps have unfortunate performance effects on rotating media, but small forward jumps can take advantage of hardware read prefetching and write buffering. In comparison, the DTRS trace contains 20% read and 32% write sequentiality.

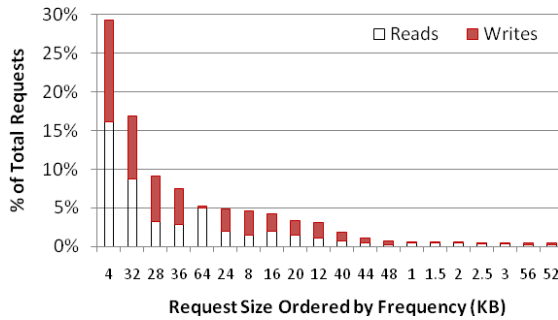


Fig. 1. Read and write request size distribution in WBS trace.

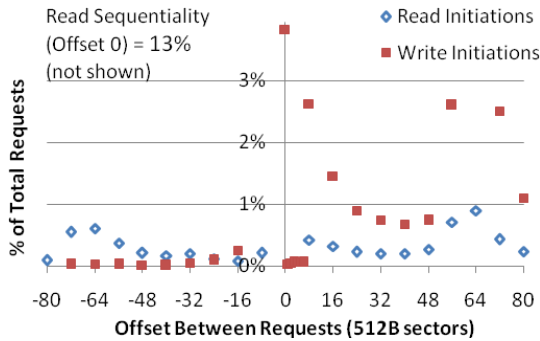


Fig. 2. Short distance spatial locality of adjacent requests in WBS trace.

B. Display Ads Platform servers (DAP-DS, DAP-PS)

Fig. 3 shows the distribution of the interarrival times in 100 microsecond buckets for the DAP-PS. While the vast majority of requests arrive within one second of the immediately previous request, the figure clearly shows periodic interarrival modes at multiples of one second (along the x-axis). For example, the overlaid CDF function shows that 5% of all requests arrive exactly one second after the most previous request. However, the log scale on the y-axis indicates that the magnitude of these interarrival time modes tails off sharply. The regularity of this behavior suggests a background service or timer that wakes up once per second during idle periods, or after a certain number of requests are issued, and issues IOs depending on system state. By

identifying the set of files accessed after these one-second delays, along with the specific processes issuing the IOs, those familiar with the workload can isolate the source of the behavior. If the server environment is power-sensitive, such periodic activity may drag the system out of low power states and thus be inadvisable.

Fig. 4 shows a similar distribution of request interarrival times for the DAP-DS. The interarrival histograms for these two workloads are quite distinct. For example, while there are many histogram buckets for interarrival times less than 250 milliseconds that are essentially empty for the DAP-PS trace, every corresponding bucket for the DAP-DS trace has at least 100 data points.

Fig. 5 shows the distribution of the “hot file” read accesses for DAP-PS. The y-axis shows the percentage of read requests for the sorted list of files. 6% of all files accessed during the trace (roughly 240 files) receive more than 97% of the total read accesses, significantly exceeding the Pareto principle (also known as the “80/20” rule). These files do not appear on the list of hot files for disk writes since they are read-mostly or read-only files.

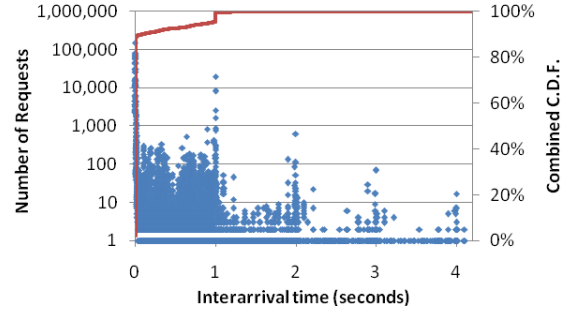


Fig. 3. Request interarrival times (100 μsecond buckets) for DAP-PS trace.

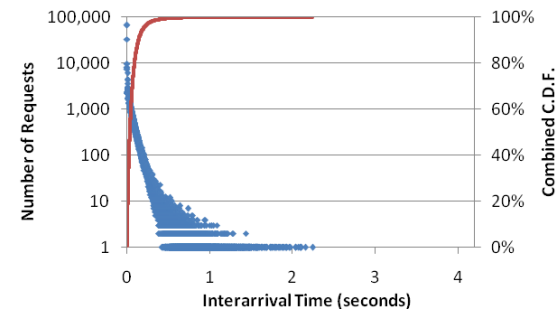


Fig. 4. Request interarrival times (100 μsecond buckets) for DAP-DS trace.

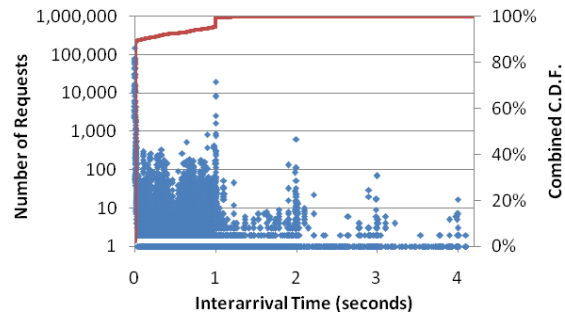


Fig. 5. Read access distribution for hottest 241 files in DAP-PS.

C. Exchange server (Exch-5, Exch -24)

Fig. 6 shows the distributions of read IO accesses across all Exch-24 disk arrays. This particular Exchange server has excess storage resources, so the disks have no problem providing the requisite throughput and responsiveness throughout the day. However, the graph shows one massive spike of read activity across the six data disk arrays at approximately 3:30 AM. Digging deeper into the event traces clearly indicates which processes and files are active during this 15-minute window. The cause of the spike is the Exchange replication task that occurs early each morning.

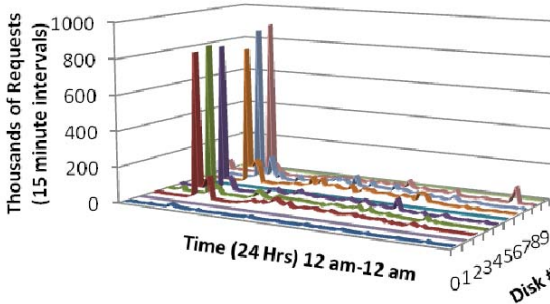


Fig. 6. IO rates in 15-minute increments for Exch-24 disk arrays.

Fig. 7 shows the distribution of hot files for Exch-5. The y-axis shows the percentage of total accesses for the sorted list of files. Enumeration of the “hottest” files in the workload can reveal interesting interactions between the application and the system. In this case, more than 20% of all IOs are NTFS metadata updates (i.e., writes to on-disk NTFS metadata structures). While the top 40+ hottest files receiving read requests are all Exchange data files, the hottest write files constitute not only writes to the Exchange database log files but also NTFS log writes. They are visible on the figure as the set of files receiving about 3% of the total writes each. These metadata writes skew the overall read:write ratio from the 1:1 value expected by the Microsoft Exchange developers to a 1:2 value in favor of write accesses. This discovery led to a more detailed investigation of potential causes for this behavior and possible “best practices” solutions. One tradeoff would be to set the system wide *DisableLastAccess* NTFS registry key if applications on the system do not need per-file last access times.

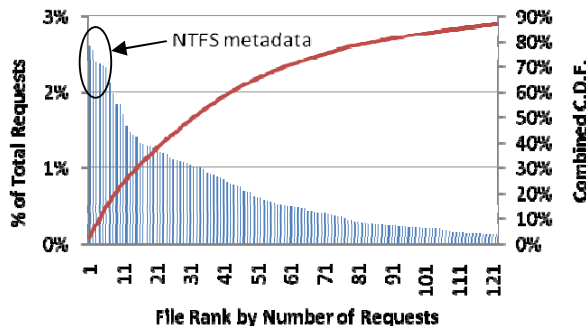


Fig. 7. Disk access distribution for hottest 125 files in Exch-5.

Fig. 8 shows the distribution of idle period durations (i.e., when there are zero disk requests outstanding) for Exch-5. When designing intelligent storage hardware components, there is often the need to perform background activities (e.g., bit scrubbing, defragmentation, page cleaning). As the best time to do this is during idle periods, determining the frequency and length of idle periods experienced throughout the phases of a given workload trace provides invaluable guidelines on what types of background activities are feasible and how finely the work has to be sliced in order to fit into the typical intervals. This particular graph shows that 45% of the idle periods in the trace last less than 15 milliseconds. The tail on this distribution extends out beyond a millisecond of idle time; there are some intervals where an Exchange disk array is idle for more than a second.

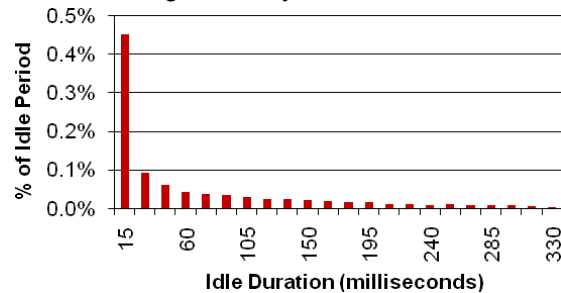


Fig. 8. Idle duration distribution for Exch-5 disk arrays.

Analysis of the traces indicates an excess of IO requests split across array stripe unit boundaries (i.e., across multiple physical disks). Microsoft Exchange Server best practices include forcibly aligning disk partition boundaries to prevent this behavior. While Vista and Windows Server 2008 automatically align partitions to 1 MB boundaries on typical disks, the traced server was running Windows Server 2003 and the primary mailbox partition was in fact misaligned. This led to the discovery of misaligned mailbox partitions on other corporate Exchange servers.

Misalignment of data within a file system can also degrade performance. NTFS defaults to 4 KB clusters (allocation units) to conserve disk space, but this can result in misalignment for larger data objects such as the 8 KB data items found in an Exchange database. Assuming 4 KB clusters and a hardware array stripe unit size of 256 KB (a reasonable choice for this workload), one might expect up to a 3% performance hit due to the extra disk accesses required for 8 KB requests that straddle stripe unit boundaries. The instrumented system appears to have 4 KB NTFS clusters, resulting in 8 KB requests to the database files (the most common size) being occasionally misaligned. Reformatting the partition with 8KB or larger clusters could solve this.

Feeding back this type of information provides real value-add to system administrators willing to take and share traces, enabling them to make informed decisions on optimizing their systems.

D. Radius back-end SQL server (RAD-BE)

Fig. 9 shows the distribution of disk read accesses (y-axis) across the LBN ranges (x-axis) of individual disks (z-axis). While this graph gives a nice overview of how the LBN space is utilized, it becomes far more useful if some basic domain knowledge is applied to filter the trace for specific workload phases or sub-workloads or file sets. Knowing how each portion of the disk space is used enables sophisticated data placement or reorganization to improve performance characteristics for the most important components of the storage workload. For example, placing hot files and directories on partitions near the “outer” tracks of a modern disk drive can improve sequential throughput by as much as 2X depending on hardware bottlenecks in the storage paths and specific disk drive characteristics. Another example would be to take advantage of temporal locality across data sets by placing them on a disk in close proximity (based on LBN), thereby potentially reducing seek delays.

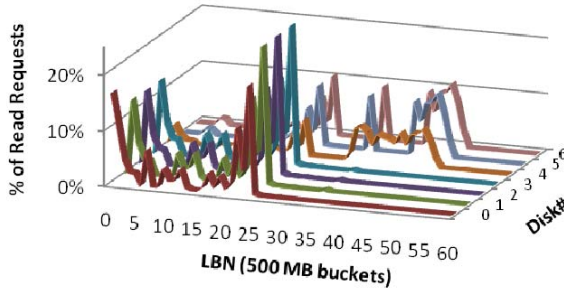


Fig. 9. Read access distribution across 500 MB buckets for RAD-BE.

E. MSN storage file server (MSN-BEFS)

Fig. 10 shows the distribution of accesses to the hottest files for the MSN storage file server. The top 15 files receive both read and write traffic, in contrast to most of the other traces where hot files are read-dominated or write-dominated. This information can be used to configure caches appropriately. The top 10% of MSN-BEFS files receive more than 99% of the total disk accesses.

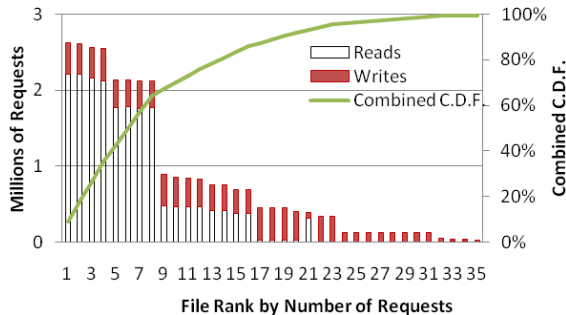


Fig. 10. Disk access distribution for hottest 35 files in MSN-BEFS.

F. Database workloads: TPC-C and TPC-E

Compared to real world traces, the TPC traces are much more steady state and have higher interarrival rates. Fig. 11

and Fig. 12 show the system-wide (not per-disk) request interarrival time distributions in 100-microsecond buckets for TPC-C and TPC-E, respectively. Request counts are plotted along the y-axis on a log scale. The majority of TPC-C requests arrive within one millisecond of the immediately previous request. However, there is a tail of requests arriving up to 10 milliseconds after the previous request. In contrast, almost all IO requests for TPC-E arrive within one millisecond of the immediately previous request.

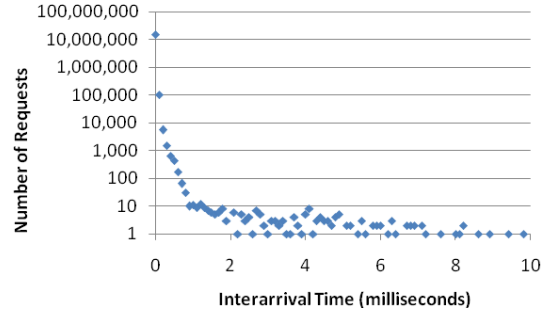


Fig. 11. Request interarrival times (100 μ second buckets) for TPC-C.

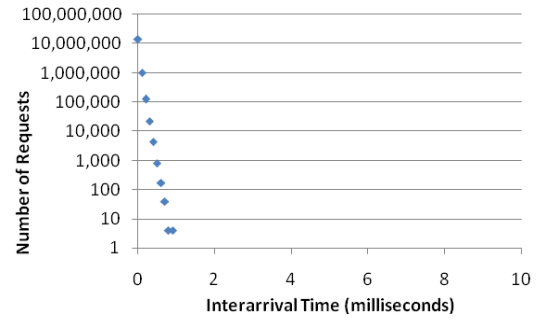


Fig. 12. Request interarrival times (100 μ second buckets) for TPC-E.

VI. SELF-SIMILARITY IN TIME AND SPACE

Self-similarity can help characterize and model the burstiness of storage workloads.

A. Theory of self-similarity

This section gives a brief introduction of the definitions of self-similarity and its estimation methodologies [5] [6] [7].

Informally, self-similarity means that a stochastic process looks “roughly” the same on any scale. Consider a stochastic process $X = \{X_1, X_2, X_3, \dots\}$ with mean μ and variance σ^2 . Its autocorrelation function $r(k)$ for $(k \geq 1)$ is defined as

$$r(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}.$$

Let $X^{(m)} = (X_k^{(m)}, k = 1, 2, 3, \dots)$ denote a process obtained by averaging X over non-overlapping blocks of size m :

$$X_k^{(m)} = (X_{(k-1)m} + \dots + X_{km-1})/m, \text{ where } m = 1, 2, \dots$$

Then the process X is called second-order self-similar with self-similarity Hurst parameter H if, for all k large enough,

$$r^{(m)}(k) \rightarrow r(k) \sim C_r |k|^{-2(1-H)}, \text{ as } m \rightarrow \infty,$$

where $r^{(m)}(k)$ is the autocorrelation function of $X^{(m)}$, C_r is a positive constant, and $0.5 < H < 1$.

This paper uses variance-time plot to get the Hurst parameter of the disk IO processes. The method of variance-time plot is based on the fact that the variances of the sample mean are decaying more slowly than the reciprocal of the sample size:

$$\text{var}(X^{(m)}) \sim a_1 m^{2H-2},$$

where a_1 is a finite positive constant. Therefore, the log-log plot of the variance of $X^{(m)}$ over m has a slope approximated as $(2H-2)$.

B. Temporal self-similarity

The self-similarity of the arrival processes of IO requests are examined for four of the traces described in Section III: DAP-DS (Display Ads Platform data server), DAP-PS (Display Ads Platform payload server), WBS (Windows build server), and MSN-BEFS (MSN back-end file server). Fig. 13 illustrates the variance-time plot for the overall arrival process of each trace. Higher H values are indicated for workloads with data points that maintain fairly constant y-values. All the arrival processes display strong self-similarity except that of DAP-PS.

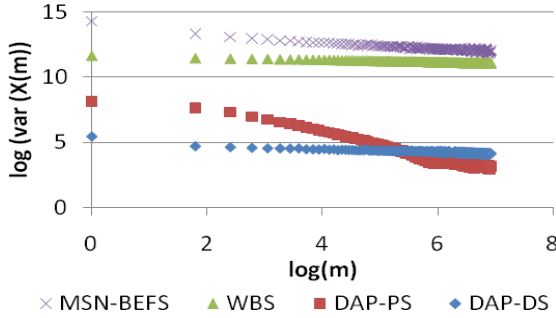


Fig. 13. Variance-time plot for the total arrival rate of four IO traces.

TABLE II
HURST PARAMETER OF IO REQUEST ARRIVAL RATES ESTIMATED BY VARIANCE-TIME METHOD

Trace	H		
	R	W	Total
DAP-DS	0.966	0.832	0.938
DAP-PS	0.461	0.705	0.548
WBS	0.945	0.939	0.958
MSN-BEFS	0.878	0.744	0.872
Disk0	N/A	0.686	0.676
Disk1	0.844	0.883	0.893
Disk4	0.875	0.688	0.865
Disk5	0.878	0.688	0.869

Table II lists the estimated H values for different arrival processes. In addition to the total arrivals to each system, the Hurst parameters of arrival processes for read and write requests to each disk are also estimated. Note that due to the small number of requests to Disk 0 in the MSN-BEFS trace, its H value is not available. Although DAP-PS does not show self-similarity in the total arrival process, its write arrival process has an H value of over 0.7. MSN-BEFS is aggregated by the requests for mainly four disks. Each disk has similar H values with the total arrival process except Disk 0, which has fewer requests than the other disks.

Alternatively, the total arrival process can be viewed as an aggregation of the arrival processes of each file. Fig. 14 displays the estimated H of the arrival processes to the top 40 “hottest” files in MSN-BEFS, which service close to 100% of the total request load (as shown in Fig. 10). Most of these processes give H values of approximately 0.7. The self-similarity does not have a strong relationship with the relative heat of the files.

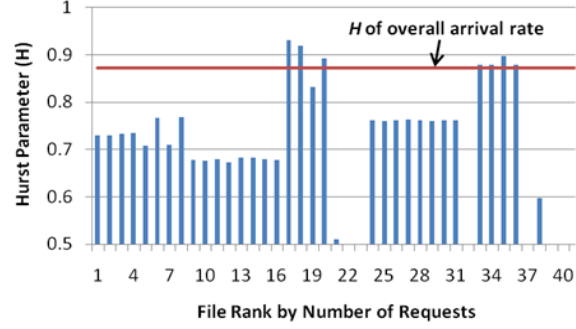


Fig. 14. Hurst parameters of arrival rates for hottest 40 files in MSN-BEFS.

C. Spatial self-similarity

Similar analysis is given for the spatial behavior of the disk access locations. Let Y_t denote the number of requests with starting location on block bucket t for $t \geq 0$. Then H of Y_t tells the self-similarity degree of the request starting locations. Table III summarizes the estimated H values for the four traces using the variance-time plot method. Most of the traces display very strong self-similarity with H larger than 0.9. In general, the write processes have lower H values than the read processes.

DAP-PS still has a low H , which indicates weak self-similarity for the total request set. However, its read request pattern gives an H value as high as 0.963 for the starting locations, while its write request pattern does not show self-similarity. This is opposite to the temporal behavior of DAP-PS reads and writes as shown in Table II. WBS has a similar request arrival process H value to that of DAP-DS, but a significantly lower H value when computing spatial self-similarity.

TABLE III
HURST PARAMETER OF LBN ESTIMATED BY VARIANCE-TIME METHOD

Trace	H		
	R	W	Total
DAP-DS	0.983	0.616	0.940
DAP-PS	0.963	0.564	0.581
WBS	0.827	0.700	0.792
MSN-BEFS			
Disk0	N/A	0.938	0.940
Disk1	0.967	0.931	0.934
Disk4	0.931	0.847	0.919
Disk5	0.936	0.857	0.925

These types of self-similarity analyses improve the accuracy of analytic modeling for different applications. They represent a promising avenue for future investigation.

VII. SUMMARY AND FUTURE WORK

The availability of comprehensive long-term traces of storage activity from a variety of production servers should enable more accurate modeling, simulation, research, development, and implementation of storage subsystem software, firmware, and hardware components. Virtually anyone having access with appropriate privileges to a production server running Windows Server 2008 can use its highly tunable in-box tracing capability to capture such traces. Ideally, a wide range of companies and individuals will provide traces to the broader storage community and publish associated analyses.

A set of twelve initial production server traces spanning a variety of workloads are characterized in this report. A lengthy set of extracted statistics as well as some sample visualizations are provided as a first step in the analysis of these traces. Future publications will undoubtedly expand on this analysis and provide deeper insights into the particulars of specific workloads. As more workloads are added to the list of publicly available traces, they will be analyzed using the ever-growing set of extraction scripts and visualization techniques. As tracing technology continues to improve, it may be possible to collect correlated end-to-end traces across multiple systems and networks.

In the near future, the authors hope to develop a Windows host model to enable closed-loop event-based simulation by emulating the Windows scheduler. ETW process, thread, context switch, and other scheduler events could be fed into the host piece of a joint software/hardware storage model to allow a workload to be realistically scaled as the underlying hardware model's performance characteristics are changed.

ACKNOWLEDGMENT

The authors wish to thank Seagate for providing the initial set of disks for storing the traces.

REFERENCES

- [1] I. Ahmad, "Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server," *Proceedings of the 2007 IEEE International Symposium on Workload Characterization (IISWC)*, Sept. 2007, pp. 149-158.
- [2] A. Aranya, C. Wright, and E. Zadok, "Tracefs: A File System to Trace Them All," *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST)*, May 2004, pp. 129-143.
- [3] Database Test Suite, Database Test 2 (DBT-2). Available: <http://osdl.dbt.sourceforge.net/#dbt2>.
- [4] G. Ganger, "Generating Representative Synthetic Workloads," *Proceedings of the Computer Measurement Group Conference (CMG)*, Dec. 1995, pp. 1263-1269.
- [5] M. E. Gomez, and V. Santonja, "Analysis of Self-Similarity in I/O Workload Using Structural Modeling," *Proceedings of the 7th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Oct. 1999, pp. 234-242.
- [6] M. E. Gomez, and V. Santonja, "A New Approach in the Analysis and Modeling of Disk Access Patterns," *Proceedings of the 2000 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2000, pp. 172-177.
- [7] M. E. Gomez, and V. Santonja, "A New Approach in the Modeling and Generation of Synthetic Disk Workload," *Proceedings of the 8th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Aug. 2000, pp. 199-206.
- [8] B. Hong, and T. M. Madhyastha, "The Relevance of Long-Range Dependence in Disk Traffic and Implications for Trace Synthesis," *22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST)*, Apr. 2005, pp. 316-326.
- [9] "io Provider," Solaris Dynamic Tracing Guide, Chapter 27. Available: <http://docs.sun.com/app/docs/doc/817-6223/chp-io?a=view>.
- [10] IOTTA Repository, Storage Networking Industry Association, <http://iota.snia.org/>.
- [11] K. Keeton, G. Alvarez, E. Riedel, and M. Uysal, "Characterizing I/O-Intensive Workload Sequentiality on Modern Disk Arrays," *Proceedings of the 4th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Jan. 2001.
- [12] K. Keeton, A. Veitch, D. Obal, and J. Wilkes, "I/O Characterization of Commercial Workloads," *Proceedings of the 3rd Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Jan. 2000.
- [13] Z. Kurmas, K. Keeton, and R. Becker-Szendy, "I/O Workload Characterization," *Proceedings of the 4th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Jan. 2001.
- [14] Z. Kurmas, K. Keeton, and K. Mackenzie, "Synthesizing Representative I/O Workloads Using Iterative Distillation," *Proceedings of the 11th International Symposium on Modeling, Analysis, and Simulation of Computer Telecommunications Systems (MASCOTS)*, Oct. 2003, pp. 6-15.
- [15] J. Lorch and A. J. Smith, "Building VTrace, a Tracer for Windows NT," *MSDN Magazine*, Sept.-Oct. 2000.
- [16] R. McDougall, "FileBench: A Prototype Model Based Workload for File Systems, Work in Progress." Available: http://www.solarisinternals.com/si/tools/filebench/filebench_nasconf.pdf.
- [17] O. Ozmen, K. Salem, M. Uysal, and M. Attar, "Storage Workload Estimation for Database Management Systems," *Proceedings of 2007 ACM SIGMOD International Conference on Management of Data*, June 2007, pp. 377-388.
- [18] J. S. Pendry, N. Williams, and E. Zadok. *Am-utils User Manual*, 6.1b3 edition, July 2003. Available: <http://www.am-utils.org>.
- [19] D. Roselli, J. Lorch, and T. Anderson, "A Comparison of File System Workloads," *Proceedings of the 2000 USENIX Annual Technical Conference*, June 2000, pp. 41-54.
- [20] C. Ruemmler and J. Wilkes, "UNIX Disk Access Patterns," *Proceedings of 1990 SIGMETRICS*, Jan. 1990, pp. 405-420.
- [21] "TPC Benchmark C, Standard Specification," June 2007. Available: http://tpc.org/tpcc/spec/tpcc_current.pdf.
- [22] "TPC Benchmark D (Decision Support), Standard Specification," Feb. 1998. Available: http://tpc.org/tpcd/spec/tpcd_current.pdf.
- [23] "TPC Benchmark E, Standard Specification," Feb. 2008. Available: <http://tpc.org/tpce/spec/TPCE-v1.5.0.pdf>.
- [24] "TPC Benchmark H (Decision Support), Standard Specification," Feb. 2008. Available: http://tpc.org/tpch/spec/tpch_262.pdf.
- [25] E. Tufte, "Sparklines: Theory and Practice." Available: http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001OR.
- [26] A. Veitch and K. Keeton, "The Rubicon Workload Characterization Tool," Tech. Rep. HPL-SSP-2003-13, HP Laboratories, Palo Alto, CA, Apr. 2001.
- [27] W. Vogels, "File System Usage in Windows NT 4.0," *Proceedings of the 17th Symposium on Operating System Principles (SOSP)*, Dec. 1999, pp. 93-109.
- [28] "vxtrace," HP-UX Reference. Available: <http://docs.hp.com/en/B3921-90010/vxtrace.1M.html>.
- [29] M. Wang, A. Ailamaki, and C. Faloutsos, "Capturing the Spatio-Temporal Behavior of Real Traffic Data," *IFIP Intl. Symp. on Computer Performance Modeling, Measurement, and Evaluation (Performance)*, Sep. 2002, pp. 147-163.
- [30] J. Wilkes, "Traveling to Rome: QoS Specifications for Automated Storage System Management," *Proceedings of the International Workshop on Quality of Service (IWQoS)*, June 2001, pp. 75-91.
- [31] T. Wong and J. Wilkes, "My Cache or Yours? Making Storage More Exclusive," *Proceedings of the USENIX Annual Technical Conference (USENIX)*, June 2002, pp. 161-175.