A Principled Approach for Selecting Block I/O Traces

Omkar Desai
Syracuse University
odesai@syr.edu

Seungmin Shin
Soongsil University
seungminshin2@gmail.com

Eunji Lee
Soongsil University
ejlee@ssu.ac.kr

Bryan S. Kim
Syracuse University
bkim01@syr.edu

ABSTRACT

We present IOTap, a tool that analyzes and profiles block I/O traces. IOTap computes the (dis)similarities among a set of workloads and sets a guideline for selecting a subset of traces for benchmarking. By doing so, we avoid experimentally running all workloads or, even worse, arbitrarily selecting a subset that skews the results. We demonstrate the usefulness of IOTap by comparing its results with experiments on real SSDs, achieving a high correlation of 0.92 for an NVMe SSD.

CCS CONCEPTS

• **Mathematics of computing** → Dimensionality reduction; • **Computing methodologies** → *Principal component analysis*; • **Information systems** → **Flash memory**.

KEYWORDS

Workload analysis

ACM Reference Format:
Omkar Desai, Seungmin Shin, Eunji Lee, and Bryan S. Kim. 2022. A Principled Approach for Selecting Block I/O Traces. In *14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '22), June 27–28, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538643.3539754

1 INTRODUCTION

Selecting I/O traces to benchmark a storage is not an easy task. Although the Storage Networking Industry Association (SNIA) hosts a wide variety of workload traces [19], it is often unclear which workloads to run for evaluation. Table 1 shows a summary of the most recent set of block I/O traces available from IOTTA SNIA [19]. I/O traces are replayed faithfully to their timestamps if performance characteristics such as I/O latency need to be measured accurately, and it would take over a year to replay all of them, making it both intractable and wasteful. On the other hand, arbitrarily choosing a small set of workloads for testing may result in a bias and lack the coverage of the full spectrum of I/O.

Table 1: Size and length of recent SNIA workload traces.

Suite	# of files	# of I/Os	Total time
YCSB+RocksDB [21]	27	352 M	0.4 Days
Virtual desktop [11]	2694	4.3 B	103.3 days
Slacker [7]	57	274.2 k	13.9 mins
Nexus [22]	31	410.2 k	23.3 mins
MS Production [10]	297	1 B	120 Days
MS Enterprise [9]	116	2.6 B	120 Days
MSR Cambridge [14]	36	434 M	8 Days
Total	3258	8.7 B	441 Days

This arbitrary selection of workloads from a huge set can lead to a benchmarking crime if it does not provide a justified reason for the selection [8]. Providing this justification is often difficult when there are too many files to select from (in the case of VDI trace [11]), or when they are too old but there is no a better alternative for that domain (in the case of MS Production traces [10]). Moreover, although we can rely on expert knowledge to select proper workload traces fit for the specific target system, we believe that evaluating a storage system under a diverse set of workloads will become increasingly relevant due to storage virtualization, workload colocation, and workload heterogeneity. Thus, we, the storage research community, need an analytical toolchain that selects a subset of I/O traces based on a principled approach.

We present IOTap, a tool that extracts important features from traces and computes (dis)similarities among them to provide a guideline for selecting traces when benchmarking storage systems[1]. The three main advantages of using our tool are as follows. First, IOTap is unbiased, analytically

[1]Our tool is available at https://github.com/swiftomkar/IOTap

HotStorage '22, June 27–28, 2022, Virtual Event, USA

ACM ISBN 978-1-4503-9399-7/22/06…$15.00
https://doi.org/10.1145/3538643.3539754

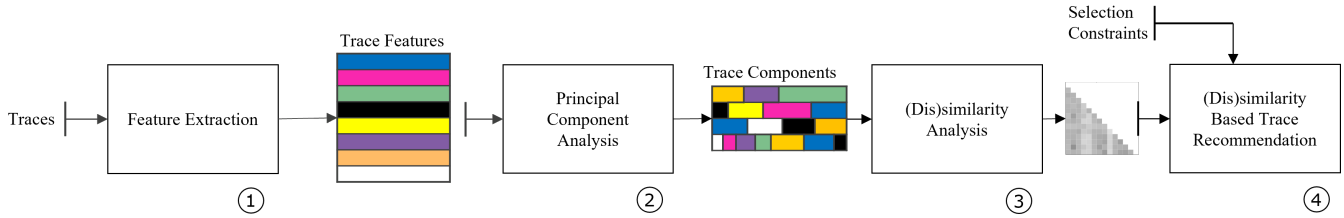


Figure 1: Workflow of IOTap. It (1) extracts 253 features from each trace, (2) reduces its dimensionality through PCA (Principal Component Analysis), (3) computes the (dis)similarity between traces based on their distance in the PC dimension, and (4) sets forth a guideline for selecting traces.

computing the trace characteristics rather than empirically measuring how the trace performs, and using this information to sample a subset of workloads with the broadest coverage. Second, it is fast, only needing a single scan of the trace file to extract all of its features, and using a fast and efficient principal component analysis (PCA) [6] to understand the key differences among traces. Third, it is accurate, achieving a high correlation of 0.92 when compared to experimental results on real storage devices.

To validate our tool, we replay and measure 16 trace files among the 3258 trace files available on SNIA [19]. (We cannot replay and measure all traces as this would take over a year to complete.) We hypothesize that two analytically similar traces would exhibit similar behavior when exercised on a real storage device. We use I/O latencies as measurable behavior, and demonstrate that our hypothesis holds true. Thus, if IOTap analyzes two traces to be similar, benchmarking them would not only be redundant, but also accentuate the particular characteristics of the traces, skewing the overall evaluation. We hope that IOTap can be used for unbiasedly selecting I/O traces for a fair evaluation of storage systems.

2 RELATED WORKS

Table 2 shows a high-level overview of recent related works that analyze, classify, or characterize traces. In particular, we briefly outline the works by Tarasov et al. [20] and Basak et al. [4], as they have been validated empirically. Tarasov et al. [20] bin each I/O into a multi-dimensional *feature matrix* based on its operation type and I/O size, and use these feature matrices to build a flexible replayable model for generating representative synthetic counterparts. On the other hand, the work done by Basak et al. [4] measures latencies and workload parameters in fixed intervals and uses both CART (classification and regression tree) and hierarchical clustering to extract the workload signature; these workload signatures are used to determine if workloads can be colocated.

Table 2: Summary of Related Works.

	Statistical analysis	Empirical validation	Guideline for selecting traces
Chen et al. [5]	✓	✗	✗
Tarasov et al. [20]	✗	✓	✗
Li et al. [12]	✓	✗	✗
Basak et al. [4]	✓	✓	✗
Zhou et al. [23]	✓	✗	✗
This work (IOTap)	✓	✓	✓

We faithfully implement the two designs and examine if they can be effective in identifying (dis)similar traces. However, we identify the following three deficiencies. First, the prior approach can be very slow. In particular, we find that the signature extraction method is too slow, making it infeasible to analyze large traces. Second, if the trace does not include latencies, the signature extraction method does not work. Lastly, they are not accurate in matching analytical results with experimental measurements. We show more details to these results in Section § 5.

3 IOTAP: I/O TRACE ANALYSIS AND PROFILING

We consider the following criteria when designing IOTap.

- The tool should neither require performance measurement to classify its characteristics nor depend on the system hardware on which the traces were collected.
- The tool should be fast, only requiring a single pass for each trace, and its analysis time should scale at most linearly with the number of I/O traces analyzed.
- The tool should holistically consider all aspects of the trace as a continuous spectrum of values, rather than binning and discretizing them according to arbitrary thresholds.

In meeting these design criteria, our work processes all traces according to the workflow described in Figure 1. First, it extracts 253 features that capture the distributions of I/O type, inter-arrival distance, size, and skew. Second, it uses PCA (Principal Component Analysis) [6] to reduce the number of dimensions that characterize the workload, from 253 features to 40 PCs (Principal Components). Third, based on

Table 3: Trace attributes and features.

Attribute group	Description	Number of attributes	Number of features
I/O type	Read-write ratio, I/O change probabilities	5	55
I/O size	I/O size, data transfer rate	6	66
Inter-arrival distance	Root-mean-square of distances	3	33
Skew	Portion of data transferred in top most accessed blocks	9	99
Total		23	253

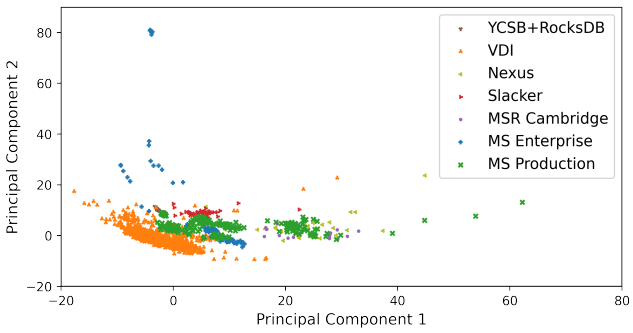


Figure 2: Two-dimensional projection on the PCA of all 3258 traces from Table 1, capturing the top 32.31% of the variance. Trace suites such as MS Production [10] and MS Enterprise [9] consists of a more diverse set of workloads compared to those such as Slacker [7] and YCSB+RocksDB [21].

the PC coordinates of each trace, it computes the distance between two traces, which represents how similar (if close) or dissimilar (if far) the points are. Lastly, it provides a guideline for selecting a subset of traces for benchmarking based on the location and inter-distance among traces.

Table 3 outlines the 23 attributes and 253 features extracted from a trace. Each attribute consists of 11 features that describe its distribution and dynamics. The first feature of an attribute is the average value across the entire trace. For example, the $I/Osize_{avg}^{entire}$ is a feature, representing the average I/O size across the entire trace. The next 5 features are the minimum, first quartile, median, third quartile, and the maximum values when the trace is chunked into 1-minute intervals. For example, $I/Osize_{max}^{1min}$ is the maximum value of the average I/O size among the 1-minute intervals in a trace. The last 5 features are the same 5 distributions but for 1-second intervals. By including distribution information from multiple chunk sizes, we describe both the second-scale and the minute-scale dynamics of the workload. In addition, extracting these features uses a moving window when scanning the trace file and does not require latency measurements.

Inspired by the approach in the architecture community to analyze CPU workloads [15, 16], we use PCA to distill the most important characteristics among the 253 features. In essence, PCA reduces the dimensionality of a dataset, transforming the 253 features into 40 PCs where each PC is some combination of the original features. Each PC maximizes the variance from the data, and 40 PCs collectively capture 93.79% of the original data's variance. This process accentuates the differences among the traces, and similar features are made less important. PCA is relatively fast and efficient, only taking several seconds to analyze a matrix of 3258 (number of traces) by 253 (number of features).

Figure 2 projects the location of all 3258 traces from Table 1 onto the two most important PC dimensions, capturing the top 32.31% of the variance. Projection onto the other 38 dimensions is not shown for brevity. All the YCSB+RocksDB traces [21] are tightly clustered together as they represent the same workload chopped into different files. Similarly, the VDI traces [11], all from the same virtual desktop workload captured in the span of 28 days, are concentrated on a relatively smaller corner of the map, despite having nearly 2700 traces (82.7% of all trace files). In contrast, the MS Production traces [10] cover a wider range of I/O characteristics, spread across the 2D plot. These analytical results are consistent with our expectation as YCSB+RocksDB traces and VDI traces are trace segments while MS Production traces are independent traces.

Table 4 lists the top 5 important features analyzed through PCA. A single feature may be part of multiple PCs, thus we show the total contribution across all PCs. The feature that discerns traces the most is the root-mean-square (RMS) of distances between two consecutive read I/Os. This feature reflects upon the sequentiality of a workload; sequential workloads have small RMS distances while random ones have large distances. The second important feature is the first quarter (Q1) for bytes read per second in 1-minute intervals. This captures the burstiness of a workload; even if two workloads have similar data transfer rates on average, a bursty workload would have a smaller Q1, while uniform workloads would have Q1 values similar to the median.

We consider the distance between two points on the 40-PC dimension to be the degree of dissimilarity. That is, if two points are close, they are similar, while if far, dissimilar. We use the Manhattan distance (L1 norm) for this purpose, and use the variance captured for each PC as the weight. We show in Section §5 that these distances correlate well with latency measurements.

Table 4: Top 5 important features according to PCA

Attribute	Features	Contribution (%)
RMS of distance between consecutive reads (*RRMS*)	$RRMS_{avg}^{entire}$	5.3
Bytes read per second (*BRPS*)	$BRPS_{q1}^{1min}$	4.7
Probability of write after read I/O (*WAR*)	WAR_{q2}^{1sec}	3.35
Portion of data transferred in top 10% hot blocks (*10HOT*)	$10HOT_{q1}^{1min}$	2.76
Probability of read after write I/O (*RAW*)	RAW_{max}^{1sec}	2.54

4 EXPERIMENTAL METHODOLOGY

To test our hypothesis that two analytically similar traces would also be similar empirically, we replay the traces and measure the latencies on real SSDs. We use a primary performance metric such as I/O latencies for validation, but other metrics such as throughput and SSD-internal write amplification can also be considered. We leave the investigation of correlating our analysis with these other metrics as future work. In addition, we do not run all 3258 traces (which would take over a year), but arbitrarily choose 16 trace files. By doing so, we, unfortunately, commit a benchmarking crime of subsetting workloads. However, we cannot use our own tool to select traces for validation as it creates a cyclic dependency. We plan to continue experimentally replaying the remainder of the traces and make the validation results available at our tool repository (https://github.com/swiftomkar/IOTap).

Figure 3 illustrates the workflow for the experiments. For replaying traces, we use `btreplay` [3] which takes in a binary `blktrace` file as input. To create this binary file, we develop `blkunparse` which converts I/O trace in text format into a `btreplay`-compatible input. As the traces are replayed, the I/O latencies are measured using the `blktrace` [2] tool.

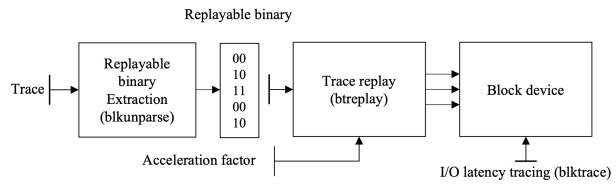


Figure 3: Replaying and measuring latencies of I/O traces for empirical validation: (1) `blkunparse` converts the block I/O in text format into a replayable binary format, (2) `btreplay` replays the trace binary with an acceleration factor on a test device, (3) and `blktrace` measures the latencies.

Table 5: Traces replayed.

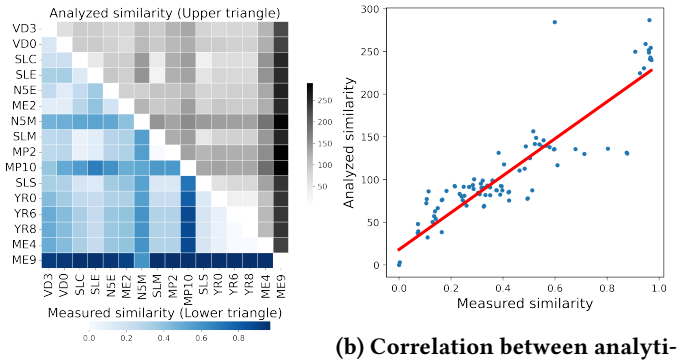
Trace file	Trace suite	Acceleration	Label
2016022212-LUN3	VDI	13	VD3
2016022314-LUN0	VDI	10	VD0
Cassandra	Slacker	51	SLC
Elasticsearch	Slacker	95	SLE
Email	Nexus	189	N5E
Exchange 2:39 PM	MS Enterprise	12	ME2
Messaging	Nexus	262	N5M
Mysql	Slacker	116	SLM
Radius 2:43 PM	MS Production	80	MP2
Radius SQL 10:05 AM	MS Production	15	MP10
Sonarqube	Slacker	60	SLS
SSDTrace-00	YCSB+RocksDB	1	YR0
SSDTrace-06	YCSB+RocksDB	1	YR6
SSDTrace-08	YCSB+RocksDB	1	YR8
TPCC 4:01PM	MS Enterprise	0.2	ME4
TPCC 9:43 AM	MS Enterprise	0.2	ME9

However, there is a fundamental issue of hardware difference between the one where the trace was collected and where the trace is replayed. For example, replaying a data-intensive workload from a storage array onto a hard disk drive would overwhelm the single drive. The other case of replaying a trace from a mobile environment is also problematic. Although there have been studies for downscaling traces [17], there is no general-purpose solution for matching hardware differences. For our work, we extrapolated the target IOPS based on the read-write ratio and average I/O size of the trace, and the performance specification of the storage drives used. Thus, we use different trace acceleration factors depending on the replayed trace and the SSD used, as shown in Table 5. The labels abbreviate the long trace names for Figure 4a.

5 EVALUATION RESULTS

In this section, we present the accuracy of our analytical method by comparing it against the measured performance of traces in Table 5. We measure the latency with `blktrace` and use the Kolmogorov-Smirnov test to quantify the similarity between two one-dimensional cumulative distributions on latencies. The empirically measured similarity is compared with the analytical similarity computed using the Manhattan distance of two traces on the PCA dimensions. We use Manhattan distance from among several distance functions as it is effective at computing (dis)similarities between high dimensional data points [1]. Prior to each experiment, SSDs are pre-conditioned [18] through 2 full-drive sequential writes followed by 2 full-drive random writes.

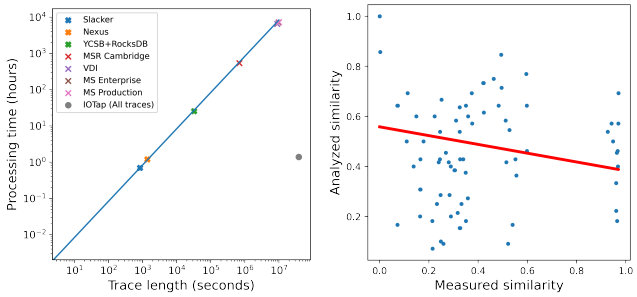
Figure 4 illustrates the similarity matrix (Figure 4a) and correlation (Figure 4b) between the analytical and empirical



(a) Similarity matrix.

(b) Correlation between analytical and empirical results.

Figure 4: Figure 4a shows the similarity matrices between analytical (upper-right triangle in gray) and empirical (lower-left triangle in blue). Lighter shades indicate higher similarity, while the darker the more dissimilar. A greater degree of diagonal mirroring means a higher correlation between the analytical and empirical results. Figure 4b plots the correlation between analytical and empirical similarity. Each point represents a pairwise comparison between two distinct traces in Table 5. The correlation coefficient is 0.92.



(a) Runtime of the signature extraction method

(b) Accuracy of the feature matrix method

Figure 5: Performance and accuracy of prior methods. Figure 5a plots the trace processing time for the signature extraction method [4]. The measured processing time (in bold ×) for the Slacker, Nexus, and YCSB+RocksDB traces are 0.6, 1, and 24 hours, respectively. On the other hand, IOTap only takes 1.4 hours to extract features from all 3258 traces together. Figure 5b shows the correlation between the empirically measured similarity and the analytical similarity using the feature matrix method [20]. The two do not correlate well, with a correlation coefficient of -0.19.

results. In Figure 4a, the degree of similarity is represented by the shade of the blues and grays. The lighter the shade, the more similar the two corresponding traces are. The analytical similarity computed using the Manhattan distance of two points from PCA is on the upper-right, while the empirical similarity quantified by the Kolmogorov-Smirnov test of the latency distributions of the two traces is on the lower-left. The similarity matrix shows a high degree of mirroring along the diagonal, indicating that our analytical approach matches with the measured results, confirming our hypothesis.

For Figure 4b, each point on the graph is a pair of two distinct traces, and its x-coordinate is the empirical similarity between their latency distributions, and the y-coordinate is its analytical similarity computed by the distance in the PCA. We observe a high correlation coefficient of 0.92. The outliers are comparisons between MS Enterprise (collected on a storage server) and Nexus (collected on a mobile phone). The locations of these outliers relative to the regression line indicate that the analytical method perceives the difference between the two trace suites to be greater than the measured latency distributions. We interpret this to be caused by the performance ceiling of the drive, limiting the measured difference between the two trace suites.

Compared with prior methods in analyzing I/O traces, IOTap is both faster and more accurate, as shown in Figure 5. Figure 5a plots the runtime of the signature extraction method based on CART [4] that shows the infeasibility of using this method on long traces. However, our approach only takes 1.4 hours to analyze all the traces together, even faster than the 24 hours for the signature extraction method to analyze only the YCSB+RocksDB traces. We are unable to compare the accuracy of this approach due to it requiring measured latencies for analysis. On the other hand, Figure 5b shows the accuracy of the feature matrix method [20][2]. In this approach, we (1) extract feature matrices for every 10-second interval in the traces, (2) cluster all the feature matrices to group similar ones into a single signature, and (3) compute the Jaccard similarity between two traces that are expressed as a set of signatures. The analyzed similarity using this feature matrix approach does not correlate well with the empirical measurements.

6 TRACE SAMPLING

Based on the analysis of all the traces, IOTap provides a guideline for selecting I/O traces for evaluation. Once the traces are placed in the principal component dimensions through PCA, we can sample a subset of traces from all available traces. For this, we use k-means clustering [13] on the traces from the PCA, where k is the number of traces to sample, and select a trace that is centrally located within each cluster. k-means is chosen after exploring several clustering techniques as it allows us to specify k and sample traces without ignoring the outliers while also scaling to large data sets in PC dimensions [13].

[2]The feature matrix is an intermediary byproduct for Tarasov et al. [20] that generates synthetic traces based on analysis. Similarity analysis is not the main purpose of that work, but we use their approach for evaluation.

Table 6: Example of sampling traces using k-means clustering with k=5.

Trace file	Read Ratio	Bandwidth	Avg. I/O size
VDI 2016031413-LUN3	0.86	27.4MB/s	32.9KB
VDI 2016031415-LUN2	0.59	5.5MB/s	18.6KB
MS Production Display Ads 6:11 AM	0.53	835KB/s	75.4KB
MS Production Display Ads 7:06 AM	0.92	600KB/s	30.5KB
MS Enterprise TPCC 10:02 AM	0.62	1.3GB/s	8.7KB
Coverage (1-KS)	0.85	0.80	0.80

Table 7: Example of sampling traces using k-means clustering with k=12.

Trace file	Read Ratio	Bandwidth	Avg. I/O size
VDI 2016022508-LUN6	0.82	9.6MB/s	17.1KB
VDI 2016030821-LUN4	0.99	361.6KB/s	5.6KB
VDI 2016031516-LUN1	0.52	3.6MB/s	17.8KB
VDI 2016031611-LUN2	0.66	21.2MB/s	25.6KB
VDI 2016031807-LUN3	0.90	23MB/s	31.2KB
MS Production Build Server 12:13 AM	0.66	18MB/s	32.7KB
MS Production Build Server 1:44 AM	0.60	21MB/s	52.6KB
Slacker Crate	0.95	4.1MB/s	18.9KB
MS Production Display Ads 12:43 PM	0.66	1.5MB/s	92.5KB
MS Enterprise Exchange 4:25 PM	0.77	10.6MB/s	15.7KB
Nexus Music FaceBook	0.87	224.3KB/s	12.9KB
MS Enterprise TPCC 9:37 AM	0.62	1.3GB/s	8.7KB
Coverage (1-KS)	0.89	0.89	0.88

Table 6 and Table 7 show two examples of sampling, using $k = 5$ and $k = 12$ respectively. To understand how representative our sampling is, we compute the coverage of the sampled subset in relation to the entire set with respect to the read ratio, bandwidth, and average I/O size. More specifically, we use the Kolmogorov-Smirnov (K-S) test to compare how similar the two distributions (sampled subset and total set) are. Even only with 5, the sample covers at least 80% of the total traces across the three attributes. The samples in Table 6 and Table 7 only serve as illustrative examples, and the coverage for other workload characteristics such as skewness that may be of interest for understanding cache hit rates may be different.

7 CONCLUSION

We propose a principled approach to benchmarking with block I/O traces through IOTap, a tool that computes the (dis)similarities among traces and provides an unbiased guideline for selecting a subset. Our analytical approach correlates well with latency measurements on real SSDs, validating the usefulness of our approach. We leave the following two directions for future work. First, we plan to further investigate trace replay methods that address hardware disparity between trace collection and trace replay. Second, we plan to correlate our analytical results with other empirical measurements such as throughput and SSD write amplification to validate that our methods extend beyond I/O latencies.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and Gala Yadgar, our shepherd, for all their constructive and helpful comments. This work was supported in part by the Samsung Memory Solutions Lab, the National Science Foundation award CNS-2008453, and the National Research Foundation of Korea award 2019R1A2C1090337.

REFERENCES

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. 2001. On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In *International Conference Database Theory (ICDT)*. Springer, 420–434. https://doi.org/10.1007/3-540-44503-X_27.

[2] Jens Axboe, Alan D. Brunelle, and Nathan Scott. 2021. blktrace. https://git.kernel.org/pub/scm/linux/kernel/git/axboe/blktrace.git/.

[3] Jens Axboe, Alan D. Brunelle, and Nathan Scott. 2021. btreplay. https://git.kernel.org/pub/scm/linux/kernel/git/axboe/blktrace.git/tree/btreplay.

[4] Jayanta Basak, Kushal Wadhwani, and Kaladhar Voruganti. 2016. Storage Workload Identification. *ACM Transactions on Storage* 12, 3 (2016), 14:1–14:30. https://doi.org/10.1145/2818716.

[5] Yanpei Chen, Kiran Srinivasan, Garth R. Goodson, and Randy H. Katz. 2011. Design implications for enterprise storage systems via multi-dimensional trace analysis. In *ACM Symposium on Operating Systems Principles (SOSP)*. 43–56. https://doi.org/10.1145/2043556.2043562.

[6] Karl Pearson F.R.S. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 11 (1901), 559–572.

[7] Tyler Harter, Brandon Salmon, Rose Liu, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2016. Slacker: Fast Distribution with Lazy Docker Containers. In *USENIX Conference on File and Storage Technologies (FAST)*. 181–195. https://www.usenix.org/conference/fast16/technical-sessions/presentation/harter.

[8] Gernot Heiser. 2021. Systems Benchmarking Crimes. https://www.cse.unsw.edu.au/~gernot/benchmarking-crimes.html.

[9] Swaroop Kavalanekar and Bruce Worthington. 2008. Microsoft Enterprise Traces (SNIA IOTTA Trace Set 130). In *SNIA IOTTA Trace Repository*. Storage Networking Industry Association. http://iotta.snia.org/traces/block-io?only=130.

[10] Swaroop Kavalanekar, Bruce L. Worthington, Qi Zhang, and Vishal Sharda. 2008. Characterization of storage workload traces from production Windows Servers. In *IEEE International Symposium on Workload Characterization (IISWC)*. 119–128. https://doi.org/10.1109/IISWC.2008.4636097.

[11] Chunghan Lee, Tatsuo Kumano, Tatsuma Matsuki, Hiroshi Endo, Naoto Fukumoto, and Mariko Sugawara. 2017. Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. In *ACM International Systems and Storage Conference (SYSTOR)*. 13:1–13:11. https://doi.org/10.1145/3078468.3078479.

[12] Cheng Li, Philip Shilane, Fred Douglis, Darren Sawyer, and Hyong Shim. 2014. Assert(!Defined(Sequential I/O)). In *USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)*. https://www.usenix.org/conference/hotstorage14/workshop-program/presentation/li_cheng.

[13] S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

[14] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. 2007. MSR Cambridge Traces (SNIA IOTTA Trace Set 388). In *SNIA IOTTA Trace Repository*. Storage Networking Industry Association. http://iotta.snia.org/traces/block-io?only=388.

[15] Reena Panda and Lizy Kurian John. 2014. Data analytics workloads: Characterization and similarity analysis. In *IEEE International Performance Computing and Communications Conference (IPCCC)*. 1–9. https://doi.org/10.1109/PCCC.2014.7017065.

[16] Reena Panda, Shuang Song, Joseph Dean, and Lizy K. John. 2018. Wait of a Decade: Did SPEC CPU 2017 Broaden the Performance Horizon?. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 271–282. https://doi.org/10.1109/HPCA.2018.00032.

[17] Sultan Mahmud Sajal, Rubaba Hasan, Timothy Zhu, Bhuvan Urgaonkar, and Siddhartha Sen. 2021. TraceSplitter: A New Paradigm for Downscaling Traces. In *European Conference on Computer Systems (EuroSys)*. https://doi.org/10.1145/3447786.3456262.

[18] SNIA. 2020. Solid State Storage (SSS) Performance Test Specification (PTS) Enterprise. https://www.snia.org/tech_activities/standards/curr_standards/pts.

[19] SNIA. 2021. Block I/O Traces. http://iotta.snia.org/traces/block-io.

[20] Vasily Tarasov, Santhosh Kumar, Jack Ma, Dean Hildebrand, Anna Povzner, Geoff Kuenning, and Erez Zadok. 2012. Extracting flexible, replayable models from large block traces. In *USENIX conference on File and Storage Technologies (FAST)*. 22:1–22:9. https://www.usenix.org/conference/fast12/extracting-flexible-replayable-models-large-block-traces-0.

[21] Gala Yadgar, Moshe Gabel, Shehbaz Jaffer, and Bianca Schroeder. 2021. SSD-based Workload Characteristics and Their Performance Implications. *ACM Transactions on Storage* 17, 1 (2021), 8:1–8:26. https://doi.org/10.1145/3423137.

[22] Deng Zhou, Wen Pan, Wei Wang, and Tao Xie. 2015. I/O Characteristics of Smartphone Applications and Their Implications for eMMC Design. In *IEEE International Symposium on Workload Characterization (IISWC)*. 12–21. https://doi.org/10.1109/IISWC.2015.8.

[23] Jiang Zhou, Dong Dai, Yu Mao, Xin Chen, Yu Zhuang, and Yong Chen. 2018. I/O Characteristics Discovery in Cloud Storage Systems. In *IEEE International Conference on Cloud Computing (CLOUD)*. 170–177. https://doi.org/10.1109/CLOUD.2018.00029.