Assert(!Defined(Sequential I/O))

Cheng Li[†], Philip Shilane, Fred Douglis, Darren Sawyer, and Hyong Shim
[†]*Rutgers University* *EMC Corporation – Data Protection and Availability Division*

Abstract

The term *sequential I/O* is widely used in systems research with the intuitive understanding that it means consecutive access. From a survey of the literature, though, this intuitive understanding has translated into numerous, inconsistent definitions. Since sequential I/O is such a fundamental concept in systems research, we believe that a sequentiality metric should allow us to compare access patterns in a meaningful way.

We explore access properties that could be incorporated into potential metrics for sequential I/O including: access size, gaps between accesses, multi-stream, and inter-arrival time. We then analyze hundreds of large-scale storage traces and discuss how potential metrics compare. Interestingly, we find I/O traces considered highly sequential by one metric can be highly random to another metric. We further demonstrate that many plausible metrics are weakly correlated, though metrics weighted by size have more consistency. While there may not be a single metric for sequential I/O that is best in all cases, we believe systems researchers should more carefully consider, and state, which definition they use.

1 Introduction

Sequential access is one of the most fundamental concepts in systems research. As anecdotal evidence, searching publications for variations of the term "sequential access" turned up tens of thousands of results. In particular, storage researchers have proposed numerous optimizations based on access patterns. Many storage systems leverage sequentiality to reduce the random seek overhead in hard disk drive (HDD) systems [14, 15]. As straightforward as it sounds, however, the notion of *sequential I/O* has multiple definitions and interpretations. The lack of clarity of the meaning of sequential I/O leads to biased system design and incomparable results.

A strict, though limited, definition of sequential I/O compares the number of consecutive addresses accessed to the count of all accesses, which we call the *Consecutive Access Ratio* (CAR). However, consider accesses to random addresses where the accesses in one trace are always a few hundred bytes, while a second trace always accesses hundreds of kilobytes at each randomly selected location. Intuitively, the trace with more *Consecutive Bytes Accessed* (CBA) is more sequential than the trace with smaller accesses, which a simplistic metric would fail to distinguish. A definition of sequential I/O should be able to handle the complexity of real-world access patterns and allow us to state that one pattern is more or less sequential than another. Also, while hardware characteristics such as sector reallocations ultimately determine whether an access is sequential or not, it is useful to analyze I/O patterns at higher system levels, such as the patterns received by a storage server.

From surveying the literature, we have found multiple definitions of sequential I/O (e.g., [3, 4, 5, 9, 11, 14, 15]) without much discussion of why a particular definition was used. For example, if we use a metric based on CAR [5, 6, 15], a particular trace we analyzed is ranked 4th out of 94 traces, but using CBA [9, 13] changes the rank to 93rd. See §4.3 for details and examples.

The contributions of our work include 1) pointing out the imprecise definition of sequential I/O; 2) a large-scale analysis of sequentiality properties on hundreds of storage traces; and 3) a recommendation that CBA should be incorporated into a sequentiality metric.

2 Properties of Sequential I/O

To cover the wide range of real-world access patterns, multiple properties affecting sequential access should be considered. From the literature and based on our own experiences, we present a set of properties that can be classified along spatial and temporal dimensions, with a sampling of citations included with each property. Along the spatial dimension, we focus on the Logical Block Address (LBA) of accesses and I/O size. For the temporal dimension, we focus on multi-stream interactions and inter-arrival time. In the process, we consider sequentiality over time as well as the sequentiality of reads relative to all accesses. We then present a set of metrics based on combinations of these properties. We are not aware of any previous work comparing sequentiality metrics, though several memory cache pattern metrics have been shown to be algebraically related [16].

Consecutive addresses. A common definition is that the logical block address of an access is consecutive, or nearly consecutive (within a specified *range*), with the previous access; i.e. $LBA_i - (LBA_{i-1} + size_{i-1}) \in range$. We can further classify consecutive addresses into the following cases:

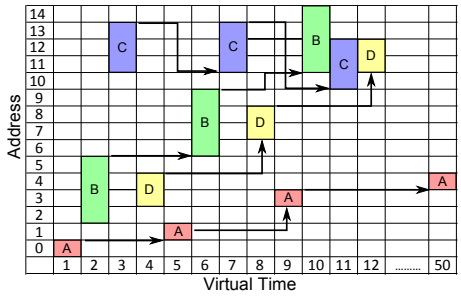


Figure 1: Four streams with different I/O sizes and access patterns highlight different interpretations of sequentiality.

- **Strictly consecutive.** The *range* is set to zero, which serves as the canonical definition of sequential access [5, 6, 15]. Because of device size limits, a large request may be split into several consecutive requests within a short time interval.
- **Strided accesses**[1]**.** The *range* includes small positive or negative values, allowing gaps, small backward seeks, and re-access of an address to still be considered sequential [7, 11]. Such patterns are common, such as database join operations inside of a nested-loop [14]. For hard disk storage, the *range* could be a track size or a file system prefetch size.

CBA. The size of I/Os affects the amount of consecutive bytes accessed, which the previous property ignores. We measure how much data are accessed, on average, after seeking to a non-consecutive address [9, 13].

We now switch to discussing temporal properties:
Interleaved streams. Multi-threaded programs leverage multi-core architectures to overlap computation with I/O accesses, and virtualization increases application consolidation. If accesses are interleaved, a multi-stream detector can help separate the accesses into potentially sequential, parallel streams. We implemented a multi-stream detector that can handle up to k streams [3].
Inter-arrival time. If the inter-arrival time is longer than a threshold, even for consecutive LBAs, we consider them as non-sequential [4, 6]. For example, as the inter-arrival time grows, there is an increased chance that background tasks will issue I/O not recorded in a trace.

We explore two further aspects of access patterns:
Sequentiality over time. Access patterns may change, so sequentiality can be calculated periodically. For example, a workload in the near past can be used to predict a workload in the near future [7, 15].
Segregate reads or writes. Since modern storage systems often use large write-buffers to service writes off of the critical path, it is worth exploring sequentiality by distinguishing reads from writes [3]. We analyzed the mixture of reads and writes and then separately analyzed the reads without writes.

[1]Strided access is sometimes defined as fixed address jumps, but we use a broader definition including uneven jumps between addresses.

Family	Metric	SR	MS	IT	Metric	Family
F1 CAR-based	M1				M9	*F2* CBA-based
	M2	x			M10	
	M3		x		M11	
	M4			x	M12	
	M5	x	x		M13	
	M6	x		x	M14	
	M7		x	x	M15	
	M8	x	x	x	M16	

Table 1: Sequentiality metrics based on CAR (*F1*) and CBA (*F2*). CAR = Consecutive Access Ratio, CBA = Consecutive Bytes Accessed, SR = stride range, MS = multi-stream, and IT = inter-arrival time.

Examples. Figure 1 shows examples to illustrate how sequentiality can be interpreted in different ways. There are four streams (A, B, C, and D) issuing I/Os of size 1, 4, 3, and 2 units respectively. Virtual time increases from left to right, and addresses increase vertically. Horizontal arrows connect consecutive accesses for a stream, while arrows with an elbow connect accesses with a stride. Consider streams A and B from time 1-12. The canonical definition of consecutive access over all accesses is the same (50% ignoring the start position). However, if we consider CBA, the sequentiality metrics are 3 and 12, respectively, highlighting that large accesses could be considered more sequential than small accesses. If we consider an inter-arrival time constraint, the access at time 50 for stream A might not be counted as sequential to the previous access at time 9, because a background task may intervene. Stream C re-accesses addresses 11-13 and seeks backward at time 11, which complicates the accounting of sequentiality. Stream D issues an I/O with a 2-unit stride distance between each access. Moreover, if all 4 streams are serviced according to their virtual time, then cycling between streams results in none of the I/Os being strictly consecutive.

Sequentiality metrics. Table 1 summarizes the main sequentiality metrics we explored. Metric 1 (M1) is the fraction of the number of consecutive accesses over all accesses, with scores in the range $[0,1]$. M2-M8 add properties to M1. Specifically, the stride range property means that non-consecutive accesses will be counted as sequential if the stride is within a threshold. The multi-stream property assigns I/Os to one of k streams when determining consecutive patterns. The inter-arrival time property causes consecutive accesses to be counted as non-consecutive if the inter-arrival time is more than a threshold. In §4, we explain how appropriate values are determined for stride range, multi-stream detection, and inter-arrival time. M9 is defined as CBA, which takes into account the size of consecutive accesses, and has scores in the range $(0,\infty)$. M10-M16 are based on M9, with additional properties. M1-M8 are called family 1 (*F1*) since M2-M8 are derived from M1. Similarly, M9-

M16 are collectively called *F2*. For purely random I/O, *F1* metrics will have scores of zero, and *F2* metrics will have scores equal to the averaged I/O size.

3 Methodology and Traces

Our methodology to understand the impact of sequentiality properties is driven by analyzing I/O traces. We first describe our approach to statistical analysis, followed by our experimental traces.

3.1 Methodology

For each metric, we calculate the sequentiality of all traces and then rank order the traces. Each metric, therefore, has an ordered list of traces, and we expect consistent metrics to have consistently ordered lists. We consider the rank order instead of the metric value itself since our metrics have different ranges.

To measure the consistency of the rank-ordered lists, we calculate Spearman's ρ [12]. Spearman's ρ ranges from 1 to -1, indicating positive to negative correlation between two ranked lists. Highly consistent metrics have values near 1, values close to 0 indicate no correlation, and negative values indicates large changes to the ordered lists. We also check the consistency of the most sequential or random traces (top and bottom of the ordered lists) by using Kendall's τ [12], which handles ranked lists with non-overlapping values. Kendall's τ has the same range and interpretation as Spearman's ρ. If metrics are strongly correlated, then it is reasonable to use the simplest metric. Otherwise, using a more complicated metric that incorporates important properties should be considered.

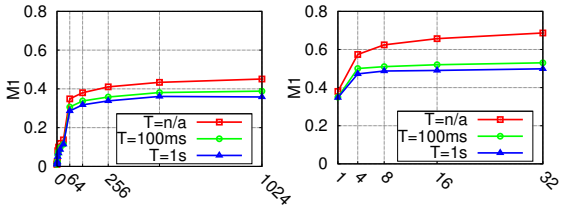
3.2 Experimental Traces

The traces we studied include block-level storage traces, both private [13] and publicly accessible [2], to enable others to reproduce our results. Our analysis focuses on block traces as compared to file system traces, since the sequentiality of block I/O is simplest to understand.

Private traces: We selected 294 traces of EMC VMAX primary storage servers that spanned at least 24 hours and had at least 1GB of both reads and writes [13].

Public traces: We created a set of 94 public traces from 4 repositories. The repositories are:

- **MS Production Server:** 50 storage traces from a diverse set of Microsoft Corporation production servers captured using event tracing for windows instrumentation with at least 700k accesses [2, 7].
- **MSR-Cambridge:** 36 block level traces for 168 hours on 13 servers, representing a typical enterprise data center [2, 10].
- **HP-IBM:** Two I/O traces from OLTP applications running at two large financial institutions and three I/O traces from a popular search engine [1].



(a) Address stride range (b) Multi-stream number

Figure 2: Determining appropriate values for stride range, multi-stream detection, and inter-arrival time properties.

- **FIU:** Florida International University collected 3 storage traces across multiple weeks for web server VMs, an email server, and home directories [2, 8].

An important question is whether our traces span sequential and random access patterns. For this analysis, we present results with M5 and M13 because they allow small strides and multiple streams. Using the default parameters (§4.1), for private traces, M5 scores are .03-.99 and M13 scores are 9KB-423MB. For public traces, M5 scores are .05-.98 and M13 scores are 5KB-2.8MB. These results suggest that our traces include highly sequential and highly random patterns for analysis.

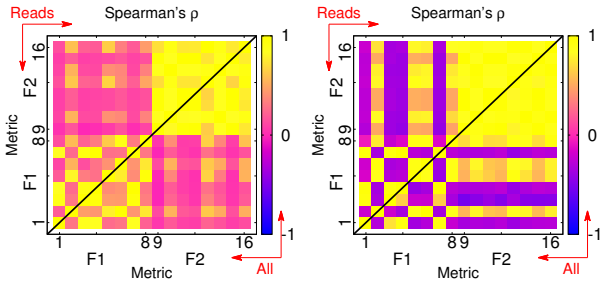
4 Evaluation

In this section, we analyze (1) the appropriate values for sequentiality properties, (2) our proposed metrics on all of the traces, (3) the consistency of the most and least sequential traces across metrics, and (4) temporal sequentiality across metrics.

4.1 Property Values

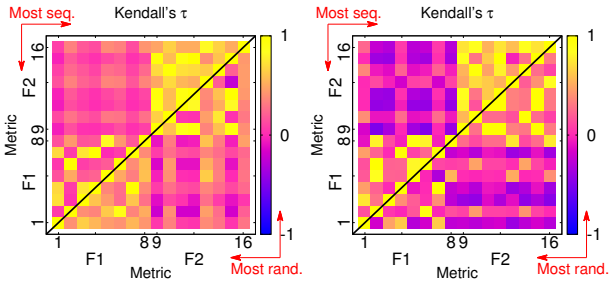
Since the parameter space is large, before presenting overall results, we determine appropriate values for relevant properties to classify an I/O as sequential. We explored a large range of possible values for each property: stride distance including back-seeks from [-1MB, 1MB] (11 values), multi-stream detector history list size from 1 to 32 (5 values), and inter-arrival time threshold from 1us to 1s (5 values). For each, we used both the CAR (M1) and CBA (M9) metrics, for both all-accesses and only reads. The cross-product of these parameters results in over 1000 different combinations that we explored.

Figure 2 shows selected experiments to determine appropriate values for stride range, multi-stream number, and inter-access time. M1 increases as the values for stride range or multi-stream detection increase. We pick a point above the area of rapid increase in the figures as the appropriate value: [-64KB, 64KB] for stride range and 16 for multi-stream detector. Note that the stride range is much larger than a sector size (usually 512B, though new disks use 4KB); this large value is likely related to the file system prefetch size or read-ahead



(a) Private block traces (b) Public block traces

Figure 3: Spearman's ρ rank correlation of sequentiality metrics for read accesses (top-left) and all-accesses (bottom-right).



(a) Private block traces (b) Public block traces

Figure 4: Kendall's τ rank correlation of sequentiality metrics for the 25% most sequential (top-left) and most random (bottom-right) traces.

size. For inter-arrival time, there is a sizable difference between 100ms and disabling the inter-arrival time threshold (n/a), however there is less difference from 100ms to 1s, so we selected 100ms. All results are consistent with M9. While these values worked well for both the public and private traces, the values should be updated when domain-specific information is available (e.g., background task schedules, degree of parallelism).

4.2 Ranking Comparison

Figure 3 shows the Spearman's ρ rank correlation coefficient for the private and public block traces. Since correlation is symmetric, we split the figure into top-left and bottom-right regions, representing results for read-only accesses and all-accesses, respectively. As an example from the public traces, <row 3, column 4> shows a correlation score of 0.63 for M3 and M4 for all-accesses, while <4, 3> shows a correlation score of 0.94 for read-only accesses using the same pair of metrics.

First, for both private and public traces, most of the metrics are positively correlated within families. The correlation of metrics across families (*F1* and *F2*) is weaker. Within metric family *F2*, the correlation is high ($\geq$ 0.65), which indicates **I/O size is a strong factor when considering sequentiality**. Specifically, for both data sets, there is a low correlation in *F1* when stride range was introduced. The multi-stream and inter-arrival time properties are largely correlated when not including the presence of stride range for *F1*. Second, the average correlation for *F1* is lower than in *F2*. Considering all-accesses, the average correlation for *F1* is 0.21 versus 0.58 for *F2* for private traces and 0.09 versus 0.92 for public traces. The read-access results have similar trends, though correlation scores are lower. Third, 96 of the 256 pairs of metrics have a negative correlation, suggesting their rankings were at least partially reversed. M3 and M9 have the most negative correlation (-0.53). This is because the MS Production traces have large I/Os per seek (sequential for M9 but not M3), and the MSR-Cambridge traces have multi-stream accesses (sequential for M3 but not M9).

Several metrics in the *F2* family, based on CBA, are highly correlated ($>$ 0.99), and we verified that the rankings were nearly identical. This shows the stability of incorporating CBA and that M9 is a good default metric unless domain-specific knowledge justifies adding specific properties such as multi-stream detection.

Besides measuring correlation between metrics, we also analyzed the sequentiality scores directly. The highest average sequentiality metric is obtained when all properties except inter-arrival time are considered, because inter-arrival time breaks sequential accesses into non-sequential accesses.

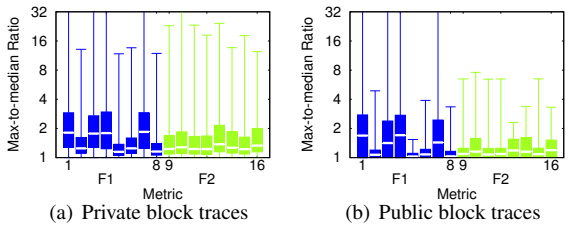
To summarize, this experiment shows that the order of ranking can vary significantly when metrics are configured with different sequential properties, but incorporating CBA leads to generally stable results.

4.3 Most/Least Sequential Comparison

Since many storage systems are optimized for highly sequential or highly random patterns (as compared to a broad range of patterns), we next focus on the top 25% most sequential and most random ranking lists to determine if they are consistent across metrics. One might hope that sequentiality metrics are at least consistent for extremely sequential or random traces.

Figure 4 plots Kendall's τ for both data sets (recall from §3.1 that Spearman's ρ cannot be used in this comparison). Correlations for the 25% most sequential and most random ranking lists are plotted in the top-left and bottom-right triangles, respectively. Although the results are not directly comparable to the Spearman's ρ experiments, the overall correlation trend is similar: the *F2* family is more internally consistent than *F1*, and there is low correlation between the metric families. Correlation scores were lower overall, indicating that sequentiality metrics are not consistent, even for the most sequential or random traces.

Consider a few examples from the 94 public traces. Comparing M1 and M2 shows the impact of stride range: only a few of the most sequential traces are in common between the metrics, "web_2" [2, 10] moved from 73rd



(a) Private block traces (b) Public block traces

Figure 5: Temporal sequentiality. *F2* is significantly more consistent than *F1*. The y-axis is logscale and truncated at 32.

to 6th, and the largest move was 69 positions. Comparing M1 and M3, "rsrch_1" [2, 10] moved from 75th to 21st when adding multi-stream detection. Comparing M1 and M9 (CBA), 0% of the most sequential traces were in common, and "Financial1" [1] moved from 4th to 93rd because there were many consecutive, small I/Os.

4.4 Temporal Sequentiality

Next we study the variation of sequentiality over time, since access patterns may vary within a trace. For this experiment, we calculate the sequentiality metrics every 10 minutes. (We also studied larger time granularities, which smoothed results.) Then we use the max-to-median ratio of the sampled data as an indication of the sequentiality variability of one trace. The larger the ratio, the greater the variability. Similar to our previous analysis, if the definition of sequential I/O is consistent, then metrics will have similar max-to-median ratios. Figure 5 shows the temporal sequentiality distribution with a box and whiskers plot including min, 1st quartile, median, 3rd quartile, and max (ordered bottom to top) of each metric for both private and public traces.

Clearly, metrics from *F1* provide an inconsistent view of sequentiality variation, because they vary significantly in terms of quartile, median and max values. For the private traces, the std. dev. of the median is 0.29 for *F1* and 0.04 for *F2*. Metrics in *F2* show a more consistent distribution, which again shows the importance of incorporating CBA into a sequentiality metric. We truncate the Y-axis at 32 for better visualization, because the maximum value for *F1* is 384. The maximum for *F2* is less than 52, and the maximum values are clustered around 20 for the private traces and 8 for the public traces.

5 Discussion and Future Work

Our work demonstrates the challenges of designing a consistent definition of sequentiality from both spatial and temporal perspectives. We further suggest systems researchers consider the following principles if possible:

I/O size matters. The canonical sequentiality definition fails to capture the impact of I/O size: M1 considers small, consecutive I/Os more sequential than large, non-consecutive I/Os, even if the CBA values are identical. Because I/Os can be split or merged at various storage layers, we feel that CBA is more indicative of sequential patterns than I/O addresses. As we demonstrated in §4, when including I/O size in sequentiality metrics (*F2*) there is greater consistency to quantify access patterns.

Stride range matters. The strictly consecutive definition of sequentiality is too restrictive in practice. Since many storage systems have internal read-ahead units (e.g. tracks), non-consecutive accesses with small strides are effectively treated as sequential.

Incorporate domain knowledge. When a metric for sequential access is based on a single formula, it cannot perfectly represent all workloads and all situations. System designers should therefore apply domain knowledge (e.g. specific prefetch scheme in a file system) and define their sequentiality metrics accordingly.

In conclusion, this paper demonstrates the controversy of sequentiality properties and its impact on sequentiality metrics. While our analysis has been on storage access patterns, sequentiality is important in other domains such as for DRAM access or synthetic I/O generation, which we leave to future work. Finally, while we have focused on sequentiality, there are other access pattern attributes that deserve a more rigorous definition.

References

[1] UMass Storage Trace Repository. `http://traces.cs.umass.edu/index.php/Storage/Storage`, 2007.

[2] Storage Networking Industry Association. `http://iotta.snia.org/traces/`, 2014.

[3] I. Ahmad. Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server. IISWC, 2007.

[4] E. Anderson. Capture, Conversion, and Analysis of an Intense NFS Workload. FAST, 2009.

[5] Y. Chen et al. Design Implications for Enterprise Storage Systems via Multi-dimensional Trace Analysis. SOSP, 2011.

[6] D. Ellard et al. Passive NFS Tracing of Email and Research Workloads. FAST, 2003.

[7] S. Kavalanekar et al. Characterization of Storage Workload Traces from Production Windows Servers. IISWC, 2008.

[8] R. Koller and R. Rangaswami. I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance. *ACM TOS*, 2010.

[9] A. W. Leung et al. Measurement and Analysis of Large-scale Network File System Workloads. USENIX ATC, 2008.

[10] D. Narayanan, A. Donnelly, and A. Rowstron. Write Off-loading: Practical Power Management for Enterprise Storage. FAST, 2008.

[11] F. Schmuck and R. Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. FAST, 2002.

[12] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. 2007.

[13] H. Shim, P. Shilane, and W. Hsu. Characterization of Incremental Data Changes for Efficient Data Protection. USENIX ATC, 2013.

[14] A. J. Smith. Sequentiality and Prefetching in Database Systems. *ACM TODS*, 1978.

[15] A. Traeger et al. A Nine Year Study of File System and Storage Benchmarking. *ACM TOS*, 2008.

[16] X. Xiang et al. HOTL: A Higher Order Theory of Locality. ASPLOS, 2013.