2022 IEEE International Conference on Networking, Architecture and Storage (NAS) | 978-1-6654-5408-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NAS55553.2022.9925320

SIST: A Similarity Index for Storage Traffic

Lu Pang, Krishna Kant, and Jie Wu
Temple University, Philadelphia, PA
{lpang, kkant, jiewu}@temple.edu

***Abstract*—In this paper, we address the characterization of similarity between two storage traces and define a three-part measure called SIST (Similarity Index for Storage Traffic). Such a measure is essential for identifying traces that are most appropriate for storage system evaluations. We compare SIST against several other similarity measures in the literature on both the object storage and block storage systems, and show the superiority of SIST in terms of its behavior for known perturbations.**

***Index Terms*—Storage systems, Time series, Discrete Wavelet transform, Similarity, Dynamic time warping**

I. Introduction

With applications becoming increasingly data-intensive, the performance of storage systems for various types of workloads is crucial to the viability of these applications. Storage system evaluation is normally done using storage traces, which are typically collected at the storage server and thus do not capture much semantic/application information. This is particularly true for block storage which is very common in enterprises [1]. The physical storage is usually divided into *virtual storage volumes*, possibly spread across multiple devices without explicit knowledge of the host/application. In this paper, we consider both the object and block storage, with traces taken from the server side for a specific volume. The traces capture the request timestamp, *Logical Block Address* or LBA (usually 4KB), request offset, I/O request size, and request I/O type (read, write).

In recent years, many commercial storage traces have become available to researchers. Traces for both object and block storage can be easily obtained. For example, Tencent has put out a ten-day trace, collected from thousands of cloud virtual volumes [2]. Internally, organizations often have rich traces from their applications, which continue to evolve as the applications and storage system technologies change. In working with such traces, a natural question is how to select one or more existing traces that are likely to have certain characteristics or cover a range of behaviors for the storage system. This requires a suitable notion of similarity between traces, which could then be used to cluster the traces into a set of "representative" classes.

In this paper, we design a similarity index called SIST focused on the characteristics of storage traces in general, while avoiding to tie it to a specific storage use-case such as tiering, caching, etc. We compare SIST against the prevalent similarity metrics, namely those for time-series and images, and show its advantages. In particular, SIST decreases gradually as an original trace is progressively perturbed to a greater degree, unlike other metrics. However, efficient SIST based trace clustering is beyond the scope of this paper.

This research was supported by NSF grant IIP-1439672

The remainder of the paper is organized as follows. Section II discusses the representation of the storage traces and the similarity measures. Section III then defines our similarity measure called SIST, with evaluation in Section IV. Section V discusses the related work and Section VI concludes the discussion.

II. Storage Trace Representation and Similarity

A. Trace Representation

Storage traces continue to get more voluminous due to the increasing size and IO rates of storage systems. This has led to increasing sizes of data transfers both for the purposes of tiering across storage devices and for caching of content in DRAM. Thus, an LBA level representation of traces is unnecessary and it suffices to work with much larger units, often known as *chunks*, which we assume to be 8MB for our work. When converting the raw trace to use chunk granularity, access to any LBA within a chunk will be counted as access to the chunk. It is also useful to divide time into slots, and report all accesses per time-slot as a list of unique blocks (or chunks) along with their address, operation (read/write), number of accesses, and average access size. Although this introduces another dimension to the problem (#accesses), it obviates the need for keeping the timestamps. Also, given the large chunk size, the average access size is well contained within a chunk. Thus, for each time-slot and chunk#, we have (#accessses, operation) as the key attributes in our trace representation. We consider reads and writes separately in our SIST measure and thus focus on #accesses.

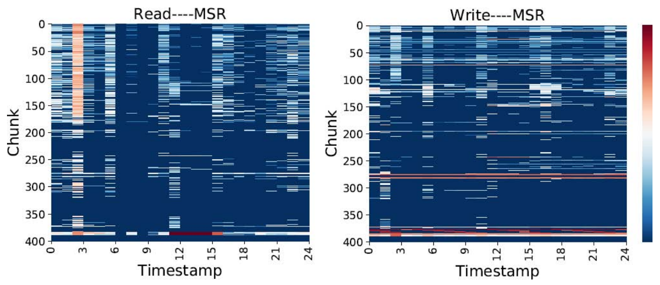


Figure 1. Number of read and write accesses of Friday for MSR usr workload

Fig. 1 shows the #accesses (or access intensity) for reads and writes for the well-known Microsoft Research Cambridge (MSR) trace [3] that come from a block storage system. We show its "usr" dataset collected on a Friday. As seen, read and write access patterns are quite different, and high access regions tend to bunch up both in time and space.

B. *Trace Similarity*

The trace representation can be considered as a 2-D time series with time-slot# and chunk# being the x and y dimensions. This allows the use of existing literature on time series similarity (TSS). One could also look at Fig. 1 as an image, which potentially allows the use of image similarity (IS) measures that have been explored extensively in the literature. However, neither view is well-suited for storage traces. In particular, TSS largely focuses on precise differences between corresponding values between two time-series, which may not be very interesting from the storage performance and hence the storage similarity perspective. The IS measures focus on how similar the images appear to the human eye, which is again different from how the storage behaves.

As can be seen from Fig. 1, the trace behavior is nonhomogeneous both temporally and spatially. That is, there are sudden changes or breaks in activity over both the chunk and time. Therefore, aggregate measures such as overall access frequency of the chunks (i.e., aggregation over time), overall accesses in successive time-slots (i.e., aggregation over space), or even the aggregate variability measure are inadequate. Instead, we need a characterization of the heterogeneity, but only to the extent that it may affect storage characteristics.

The nonhomogeneity in storage behavior in a commercial environment results from many causes. The three common cases are: (1) Applications with different characteristics starting or ending at unpredictable times, (2) Occasional unusually heavy load on some applications, and (3) Regular but distinctive activities such as backup or report generation where the behavior shows bunched up requests during a short period of time. It is likely that the two traces are roughly similar except for these special situations, and we would like to recognize them as such.

C. *Image Similarity (IS) Measures*

IS metrics have been studied extensively but focused on directly or indirectly trying to model the human visual system. The most prominent example of full image similarity is the structural similarity index (SSIM) [4]. SSIM is a simple measure computed as a product of the similarity in the mean value (or intensity), variance, and correlation between the target and reference image. The similarity between two values μ_x and μ_y is computed using the expression $s_{xy} = 2/(\mu_x/\mu_y + \mu_y/\mu_x)$, which has the important properties of $s_{xx} = 1$, $s_{xy} = s_{yx}$, and $s_{xy} \leq 1$. For correlation, it uses the standard (Pearson's) correlation coefficient, but this would be inappropriate as it emphasizes large differences. The other extreme is to use Spearman's rank correlation [5], which only correlates the trend (increasing or decreasing) across the traces. However, this is rather insensitive to large differences in accesses.

MS-SSIM [6] is a generalization of SSIM to multiple (resolution) scales, by successively applying a low pass filter (similar to using Haar Wavelet transform). Gradient similarity measure (GSM) [7] essentially replaces the intensity term of SSIM with a similar term involving gradients, thereby focusing on edges rather than the general intensity. SIST also adopts this idea.

D. *Time Series Similarity (TSS) Measures*

The TSS measures in the literature focus on scenarios where direct comparison of values is important (e.g., for comparing gnomic structure, stock price, ECG signals, etc.) The three most popular comparison types are (a) Euclidean (or the more general Minkowski) distance, (2) Edit distance (or the number of changes required to convert one series into another), and (3) Longest Common SubSequence (LCSS) [8]. The edit distance is not suitable to evaluate the attributes of storage traces because there is no natural notion of a "word" in storage access traces.

Time series from two different sources may be misaligned and thus a direct comparison can be misleading. In particular, the two series could have a relative shift, compression, or expansion. Dynamic time warping (DTW) [9], originally developed for speech comparison, is a highly popular approach for alignment. However, it may look arbitrarily far from the current point to find the closest point to match, which makes it exceedingly expensive. DTW also has another problem: if one series has repeated occurrences of the same value, they all might be matched to the same value in the other series. This problem may occur rather frequently in the traces where the "value" is the number of accesses to a chunk in a small time window. It can be lessened by matching the trend rather than the actual values [10], where the trend is often estimated from the next/previous points only. We will use a low-dimensional approach with a small band to indicate the trend.

III. Similarity Index for Storage Traces

A. *Preprocessing and Trace Grids Representation*

For each request in the raw trace, we convert the LBA offset into the corresponding chunk number. Then we collect the total number of read and write requests received within successive time-slots of duration t_s into separate quantities R_t and W_t. Let K be the maximum offset in the storage trace. Then, m, the total number of chunks, is given by $m = \frac{K}{|C|}$, where $|C|$ is the chunk size. Suppose that our data spans n time-slots. Thus, the range of the spatial dimension is $[1, \cdots, m]$ and the range of the time dimension is $[1, \cdots, n]$. For a grid point in position **(chunk_no, time-slot)** denoted as $[i][j]$, its value represents the total number of read or write requests for the specific chunk and time-slot. Fig. 2 shows an example of our data grids. The time-window size t_s and chunk-size $|C|$ are parameters, and their choice is discussed in Section IV-D.

B. Similarity Measure for Storage Trace

Consider two equal size trace grids $G_{m\times n} = g[1\ldots m][1\ldots n]$ and $H_{m\times n} = h[1\ldots m][1\ldots n]$, as illustrated in Fig. 2. Based on the discussion above, we define similarity between traces G and H as a triplet $\mathbb{S}_{G,H} = (S_M, S_A, S_D)$ where

1) $S_M \in [0,1]$ represents the overall (or Main) similarity measure between the two traces by exploiting discrete wavelet transform (DWT).
2) $S_A \in [-1,1]$ represents the similarity in terms of the number of chunk access intensity (or the Activity level).
3) $S_D \in [-1,1]$ is the similarity for high-frequency content (or Detail) that is ignored in the computation of S_M.

Given two identical traces, the similarity measure will be (1,0,0) meaning that they have the same overall characteristics, and the activity level and high-frequency content are the same.

C. Estimating Activeness Similarity S_A

Given two equal size trace grids $G_{m\times n} = g[1\ldots m][1\ldots n]$ and $H_{m\times n} = h[1\ldots m][1\ldots n]$, the activeness of G and H is computed as

$$A(G) = \sqrt{\sum_{\substack{1\le i\le m\\1\le j\le n}} g[i][j]^2}, \quad A(H) = \sqrt{\sum_{\substack{1\le i\le m\\1\le j\le n}} h[i][j]^2}, \tag{1}$$

Now we define the activeness difference between these two traces as

$$S_A(G,H) = \frac{A(G)-A(H)}{A(G)+A(H)}. \tag{2}$$

Note that $S_A(G,H) \in [-1,1]$. The negative value of $S_A(G,H)$ means H is more active than G, and the positive value means G is more active than H.

D. Estimating Main Similarity Index S_M

For this, we first compute a row-wise distance measure between G and H, henceforth denoted as $d(G,H,i)$, and then the overall measure, denoted as $D(G,H)$ by a simple summation. That is,

$$D(G,H) = \sum_{1\le i\le m} d(G,H,i).$$

Note that $d(G,H,i)$ denotes the similarity in terms of accesses to chunk i. A simple summation reflects the fact that accesses to all chunks are given equal weight.

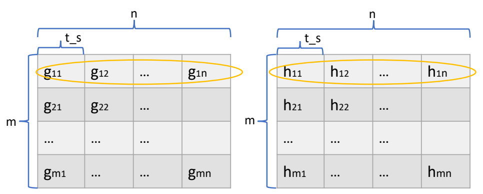


Figure 2. Computing similarity of traces G and H

Now to estimate $d(G,H,i)$, we can regard $G(i)$ and $H(i)$ as time series and use a suitable distance measure. A suitable candidate for this is Euclidean distance along with DTW. However, a full DTW has $O(N^2)$ complexity for a time series of length N since it may examine the points that are arbitrarily far. This is not necessary or meaningful for a storage trace that has been aggregated into fairly large time windows and chunks. Thus we use the Sakoe-Chiba banded comparison [11] where the ith element in one series is only compared against element j with $i-L \le j \le i+L$ for some small band size of $2L+1$. (Note that $L=0$ corresponds to the case of simple Euclidean distance.) We let $r = (2L+1)/N$ where r is a fractional parameter $0 \le r \le 1$. The choice of r is discussed in Section IV-D.

Instead of using the similarity distance $D(G,H)$ directly, we scale the distance by the activeness of the two trace grids. Storage traces normally follow the Pareto principle where a large percentage of I/O requests a small section of the storage. Thus, the similarity scaled by the activeness of the request data is the part we should focus on. We normalize the scaled distance $d_{scale} = \frac{D(G,H)}{4*(A(G)+A(H))}$ using $\frac{1}{1+d_{scale}}$ to output a value in the range of $[0,1]$. That is,

$$S_M(G,H) = \frac{1}{1+\frac{D(G,H)}{4*(A(G)+A(H))}}. \tag{3}$$

E. Wavelet Transform Based Distance Comparison

Since raw traces are likely to have quite a bit of "noise" or high-frequency variability that is not necessarily the most fundamental part from the storage perspective, direct analysis of the trace becomes less desirable. Wavelet transform provides an attractive way to handle this situation since it includes both the frequency (through the scale factor) and temporal location (through the time-shift factor). Discrete wavelet transform (DWT) represents scale factors as powers of 2 and uses a pair of high-pass and low-pass filters in each step, and the decomposition is applied recursively to the low-pass part [12], [13]. Both filters are constructed using the scaled and shifted versions of an underlying *mother wavelet*. The eventual output consists of a concatenation of all the wavelet coefficients. The simple Haar mother wavelet is used most commonly and amounts to computing the average and difference of adjacent pairs of elements. The differences become the *detail* coefficients at this level, and the process is continued for the next level using the averages recursively until we get down to the last level which is simply the overall average of all elements. With the Haar wavelet, the successive stages of DWT provide "trends" at lower frequency levels.

With DWT, we can get a low dimensional representation of the signal by ignoring the top k detail coefficients (where k is a parameter), and use this for a "less noisy" distance measure. We also compute a measure on ignored detail coefficients to capture the high-frequency content. One benefit of the high-frequency content is that it might reveal the activities of special processes (e.g. backup, report generation, etc.) or other special activities that tend to have intensive accesses over short time periods. These could be considered as "anomalies" in data [13] and DWT provides a convenient way to characterize them.

F. Estimating Similarity Measure with DWT

Given two equal size trace grids G, H of size (m, n), $\overline{G}$ is a representation of G with reduced dimensionality (m, n') and $\overline{H}$ is a representation of H with reduced dimensionality (m, n'), where $n' \ll n$. $\overline{G}$ and $\overline{H}$ extract the main features of G and H respectively.

With the low dimensional grids $\overline{G}$ and $\overline{H}$, we form the similarity degree of two traces G and H as $S_M(\overline{G}, \overline{H})$, and the activeness difference as $S_A(\overline{G}, \overline{H})$.

We then use the top k detail coefficients of DWT to compute the differences in the finer details. We output the finer detail difference between the traces.

$$S_D(G, H) = \max_l \frac{RMSD(G_{cd}^l) - RMSD(H_{cd}^l)}{RMSD(G_{cd}^l) + RMSD(H_{cd}^l)}, \quad (4)$$

where G_{cd} and H_{cd} are the detail coefficients. We scale the input two trace grids using the minmax scaling.

G. Implementation Details

In this section, we discuss the details of the implementation choices we made. We give the choice of the method that we use to measure the distance $D(G, H)$ and the method used to form the low dimensional representation $\overline{G}$ and $\overline{H}$.

1) Using DTW for Distance Measure: As previously discussed, the distance measure of two trace grids is obtained by summing up the distance per corresponding row vectors that represent the request intensity change in time of a specific chunk in these two traces. We use constrained DTW to align each pair of the corresponding row vectors.

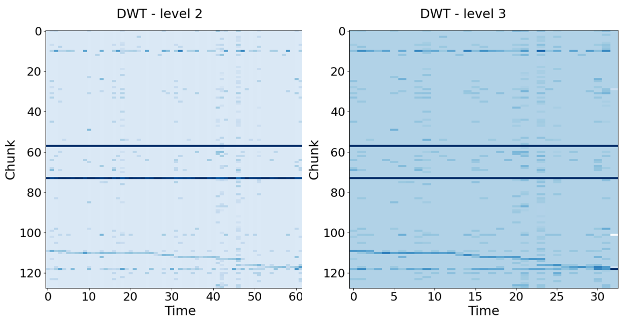


Figure 3. Sample DWT_T at levels 2 and 3

2) Using DWT for Trace Analysis: In using DWT for traces, we intend to deal with 2 dimensions – time-slot number and chunk number. One could thus start with a 2-D DWT, where each scale will be a pair (k_s, k_t) with k_s denoting the chunk-space and k_t the time. This is rather unwieldy; therefore, we consider DWT individually for each dimension. We will largely use the DWT in the time domain (DWT_T). This provides a 2-D plot at each time scale, where x-axis is the detail coefficient index of DWT_T at that scale and y-axis is the chunk number. The value at (x, y) is x_{th} wavelet coefficient value for chunk no. y. Fig. 3 shows the DWT_T plot at levels 2 and 3. Note that the amount of coefficients becomes 1/2 as we go down each level. Thus, for example, the two vertical bands around x position of 25 in Fig. 3(b) have the corresponding (rather faint) bands around position 50 in Fig. 3(a). This shows that there is significant activity around 80% time offset into the trace across the entire range of chunks, and this is prominent only at level 2 (higher frequency).

To illustrate the spatial DWT (DWT_S), imagine that in Fig. 3 the x axis is the time-slot number and the y-axis is the index of the DWT coefficient. Thus, level 2/3 activity will correspond to the variability in access behavior at the chunk granularity of 16MB and 32MB respectively. Thus, the DWT_S provides us with information about the level of sequentiality in the trace. For example, if Fig. 3 were actual DWT_S plots, we would expect many 16 MB accesses at 80% time point all across the volume, but much fewer 32 MB accesses. Thus, DWT_S can be useful in finding out the reasonable granularity for chunking.

3) Overall Algorithm: The pseudocode implementation of computing the trace similarity distance is given in Table 1

Algorithm 1 Trace Similarity Distance (TSD) algorithm

```
1:  procedure TSD($G, H, l, r$)        ▷ Input trace grid G, H, wavelet level l, and DTW constraint band $r$
2:      $G_s = scale(G, min, max)$
3:      $H_s = scale(H, min, max)$
4:      $G_{ca}, G_{cd} = DWT_T(G_s, wavelet = haar, level = l)$
5:      $H_{ca}, H_{cd} = DWT_T(H_s, wavelet = haar, level = l)$
        ▷ $G_{ca}$ and $H_{ca}$ is the approximation coefficients; and $G_{cd}$ and $H_{cd}$ are the detail coefficients
6:      while $i \leq n$ do        ▷ For each chunk
7:          $D_i = DTW(G_{ca}[i, :], H_{ca}[i, :], constraint = r)$
8:      $D(G, H) = sum(D_i)$
9:      Compute similarity degree $S_M(G, H)$ by Eq.3
10:     Compute activeness different $S_A(G, H)$ by Eq.2
11:     Compute finer detail different $S_D(G, H)$ by Eq.4
12:     return $S_M(G, H), S_A(G, H), S_D(G, H)$
```

After some experimentation, we chose $k = 3$ and $r = 0.15$. We use the minmax scaling with the range [0,1] to normalize the input grids. The minimum value of the grids is made equal to 0 and the maximum value is made equal to 1.

H. Evaluation Methodology

Although SIST is intended for use with different traces that might exist in a storage trace repository, such comparisons are not useful for evaluation purposes since we would not have any underlying ground truth to assess whether the obtained similarity measure is reasonable. Therefore, we take individual traces and apply certain perturbations to them to generate a variant to be compared against the original. In the following, we describe four basic types of perturbations that we believe cover some key differences between real traces. These could be used both individually and in combination for evaluation purposes.

- Mixing: Here we mix two different traces, say A and B using a percentage parameter p, where the mixed trace $M = (1 - p\%) * A + p\% * B$. The mixed trace can now be compared against both A and B.

- Shifting: Here we shift the original trace in the left or right direction by a certain percentage denoted as $p\%$ to get a trace variant for comparison. Any vacated places are filled with zeros.
- Salt and pepper noise addition: Here we replace $\frac{p}{2}\%$ of the accesses (in randomly chosen positions) to have the minimum access count and the remaining $\frac{p}{2}\%$ of accesses to have the maximum access count.
- Traffic Thinning: For this, we simply remove randomly chosen $p\%$ of the accesses (i.e., make them zero) without changing the trace length.

We have verified that using multiple of these perturbations has a roughly additive impact on SIST; therefore, for brevity, we only provide results with a single perturbation in each case.

To evaluate the performance of SIST, we compare it against the following four measures from the literature.

- Structural Similarity Index (SSIM): The detail of this measure is mentioned in II-C. In our experiment, we treat the trace grids as the input images to compute the similarity.
- Longest Common SubSequence (LCSS) similarity: We compute this as the length of the longest common subsequence divided by the time series length.
- Euclidean Distance (Euclid): This measure computes the Euclidean distance $d_{Euclidean}$ between the two time series, normalized using the expression $1/(1+d_{Euclidean})$.
- Dynamic Time Warping based similarity: This measure calculates the mean DTW distance d_{DTW} for the row vectors of the trace grids and normalizes it as $1/(1+d_{DTW})$.

IV. Evaluation of SIST

A. Datasets Used

In the following, we show detailed results for traces from two sources. One is the MSR block trace that we discussed earlier [3]. This trace is for an enterprise system and includes several workloads, of which we use one workday trace from one of the disks associated with "usr", "proj", and "hm" workload. We call the trace from "usr" as trace A, "proj" as trace B, and "hm" as trace C. The other traces [14] are from an HPC environment (Rennes site of Grid'5000 environment [15]) and uses the OrangeFS [16] parallel file system. The traces include 14 read workloads and 14 write workloads generated using the MPI-IO Test tool [17]. Among the many configuration parameters, an important one is the shared file for all applications vs. one file per application. We use one trace in each category here. For these traces, the object size is only 64KB, and the max access size is 4MB. We selected two traces from the HPC environment, that we call as traces A and B. The spatiality parameter of trace A is configured to be contiguous, whereas that of trace B is configured as non-contiguous. The two traces were also generated using a different number of processors, one is 64 + 64 the other is 64 + 32. For the mixing setup, we also selected another trace which we call trace C which has a different request size than A and B.

B. Evaluation of Main SIST Measure S_M

Figs. 4, 5, and 6 show the comparisons for the (object store) HPC trace. We use the four perturbations mentioned above, namely mixing of two traffics, left/right shifting, salt-n-pepper noise addition, and thinning out of the traffic.

Let us start with Fig. 4(a) which compares a mixture of traffics A and C (denoted $Mix(A,C)$ in the caption) with A. The x-axis indicates the percentage of the traffic C that is used in the mixture and it varies from 0% to 100%. As expected, at 0%, all methods yield a similarity of 1.0, and it generally decreases as the percentage of C increases. It is seen that LCSS, DTW, and Euclid go down very rapidly and thus do not represent the mixture very well. (The LCSS even rises when the traffic is 95% type C). Both SIST and SSIM show plausible behavior, but SIST has an almost linear decay until it reaches 50% B, at which point it stabilizes to the inherent similarity between A and B. Fig. 4(b) shows a symmetrical behavior except for the slight asymmetrical of SSIM between (a) and (b). Cases (c) and (d) use a mixture of traffic B and C, and the results again show similar behavior.

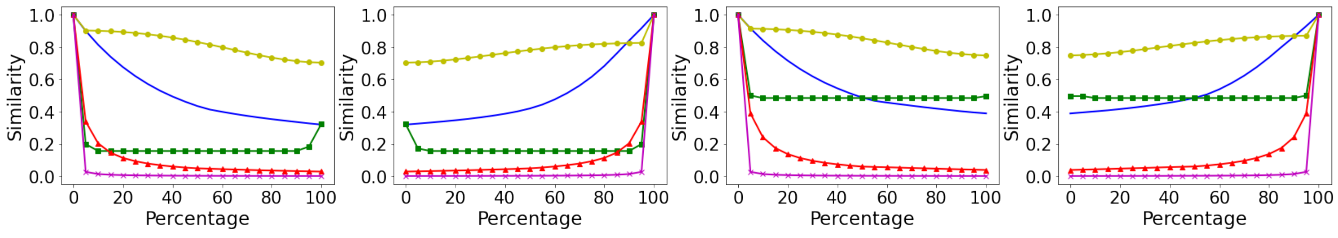
Fig. 5(a) shows the result of right shifting traffic A by the percentage on x-axis. Such a shift will zero out increasing amounts of initial trace and thus expected to lead to increasing dissimilarity until we reach the limit of similarity of traffic A with all zeros. The zeroing out creates an extremely non-homogeneous situation in time and is challenging for all measures. Nevertheless, SIST shows a roughly gradual decay up to the minimum similarity value. SSIM here shows a gradual increase after a very rapid drop, whereas DTW/Euclid shows no ability to tolerate the shift. Fig. 5(c) shows a similar behavior for trace B. Figs. 5(b) and (d) show that the left shift performs almost identical to the right shift.

Fig. 6(a) shows the impact of salt-n-pepper noise mixture on trace A. Such noise is rather extreme and all methods except SIST record a negligible similarity with only a small amount of noise. SIST may appear to show anomalous behavior of settling down at around 0.3. However, note that salt-n-pepper noise largely results in adding a high-frequency noise to the signal. SIST will catch this through the S_D part of the triplet (discussed below); however, with high-frequency noise filtered out, the trace retails a level of similarity with the unperturbed trace regardless of the percentage of noise. Fig. 6(b) shows similar behavior for traffic B.

Fig. 6(c) and (d) show the similarity with the traffic thinned out by an increasing percentage. Again SIST shows the most plausible behavior – a smooth and concave curve ending in a rather small similarity with all zeros. In contrast, SSIM decreases only slowly and is very high even for all zeros. DTW and Euclid go down to a negligible value rapidly, and LCSS drops out unexpectedly at around 45%.

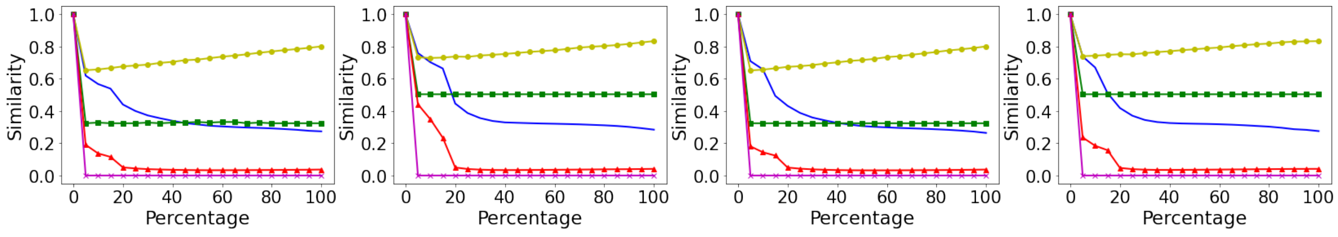
We ran all the same experiments with the "usr" and "proj" parts of MSR block traces. The results turned out to be quite similar and thus only some are depicted in Figs. 7 and 8. It can be seen that even though the detailed behavior is different, the trends are very similar to that for the HPC trace, and once

Legend for Figs. 4, 5, 6, 7, 8, 9: SIST SSIM LCSS DTW Euclid



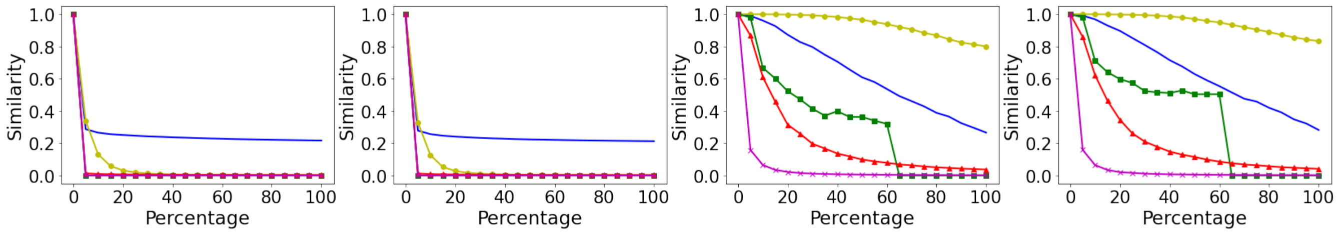
(a) Mix(A,C) vs. A (b) Mix(A,C) vs. C (c) Mix(B,C) vs. B (d) Mix(B,C) vs. C

Figure 4. Comparison of different similarity measures for mixing (HPC traces)



(a) Shift(A, right) vs. A (b) Shift(B, right) vs. B (c) Shift(A, left) vs. A (d) Shift(B, left) vs. B

Figure 5. Comparison of different similarity measures for shifting (HPC traces)



(a) Salt_pepper(A) vs. A (b) Salt_pepper(B) vs. B (c) Thinout(A) vs. A (d) Thinout(B) vs. B

Figure 6. Comparison of different similarity measures for noising and traffic-thinning (HPC traces)

again SIST shows a smooth decay in similarity as the amount of perturbation increases.

We also evaluated SIST using grids where a point in position $(chunk\#, time)$ denotes the average *stack distance* for $chunk\#$ within the specific time-slot. Fig. 9 shows the similarity comparisons for the four perturbations of the results with HPC stack distance grids. It is shown that the behavior with stack distance is quite similar to the ones with access trace grids we discussed above. Stack distance is an important measure of locality and hence performance; therefore, finding an almost similar behavior for stack distance confirms that SIST should also track the storage performance quite well.

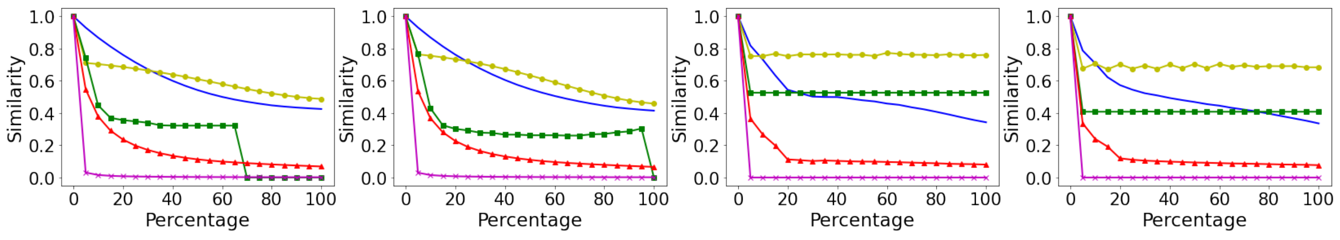
C. Evaluation of S_A and S_D SIST Measures

Since other similarity measures do not have the counterparts to S_A and S_D, we only show them for various perturbations to traces A and B. Fig. 10(a) shows the S_A metric, which ranges from -1 to +1 with zero meaning no detectable difference from the original. It is seen that the salt-n-pepper noise shows the most dramatic behavior: the S_A metric rises rather rapidly and then saturates at around 0.75. Since the salt-n-pepper noise is applied to all the chunks, it makes many new chunks active. In contrast, trace mixing does not have much influence on activity level. The shift and thinning of the traffic both reduce the activity level (by zeroing out some accesses) and the impact becomes very pronounced at a high percentage of shift/thinning. Fig. 10(b) shows very similar behavior. The behavior for MSR traces in Fig. 10(c) and (d) is also qualitatively similar to that of the HPC traces.

Fig. 11 shows the S_D metric for the same cases as for activity level. The S_D metric also takes the range -1 to +1 with zero indicating no detectable difference from the original. It is seen that the behavior S_D is generally qualitatively similar to that of S_A. This is largely a result of most perturbations affecting both the activity level and high-frequency content in a similar way; however, the two concepts are not identical and can capture the different aspects for certain situations, and thus it helps retain both measures in SIST.

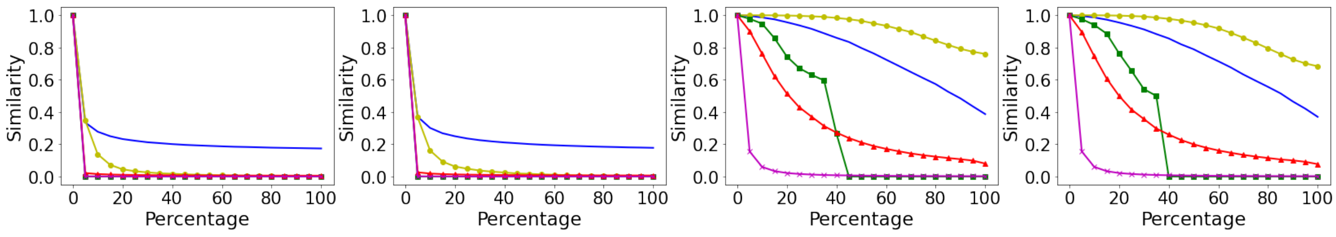
D. Justification for Parameter Choices

Because of the complexity of the storage traces, deriving a similarity measure for them necessarily involves a judicious



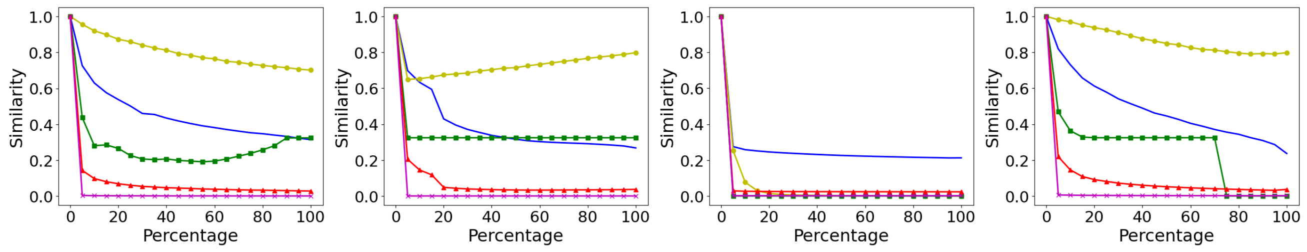
(a) Mix(A,C) vs. A (b) Mix(B,C) vs. B (c) Shift(A, right) vs. A (d) Shift(B, right) vs. B

Figure 7. Comparison of different similarity measures for mixing and shifting (MSR traces)



(a) Salt_pepper(A) vs. A (b) Salt_pepper(B) vs. B (c) Thinout(A) vs. A (d) Thinout(B) vs. B

Figure 8. Comparison of different similarity measures for noising and traffic thinning (MSR traces)



(a) Mix(A,C) vs. A (b) Shift(A, right) vs. A (c) Salt_pepper(A) vs. A (d) Thinout(A) vs. A

Figure 9. Comparison of different similarity measures for various perturbations in the stack-distance (HPC traces)

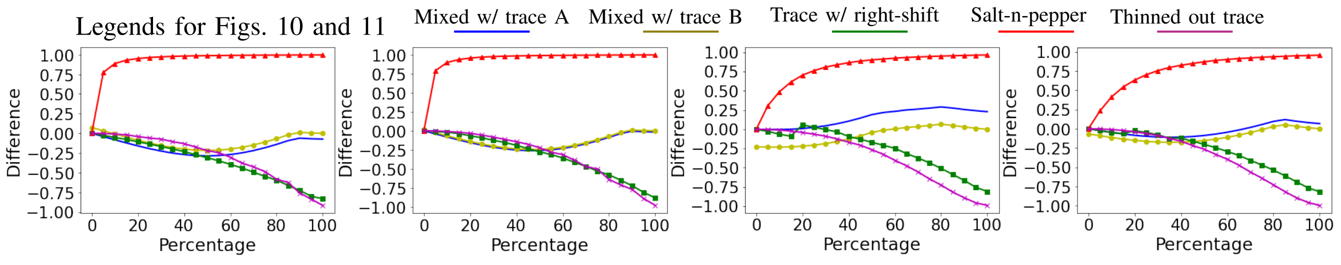
choice of several parameters. For the proposed method, there are four different parameters. In the following, we provide some justifications for their choice.

- Time-slot duration t_s: The choice of t_s depends on the intensity of the trace. Most storage operations occur in phases, with each phase accessing a rather small range of LBAs actively (and others either occasionally or none at all). We need to choose t_s large enough to contain a significant number of accesses to these active ranges (e.g., 10's to 100's), else the similarity is either not meaningful (too few accesses) or not granular enough (too many accesses).
- Chunk-size C: Chunk size needs to be chosen so that much of the sequentiality is captured within the chunk access and corresponds to data transfer sizes in contemporary storage systems that continue to go up both in total size and hence the access size to control the management overhead. Currently, a chunk size of 1-16 MB is perhaps appropriate in most cases.
- DTW band size (r): One important aspect in deciding r is the maximum span beyond which trace characteristics are considered to be sufficiently different that does not make sense to match them. Based on this and some experimentation, we chose the total band side of 15% (or approximately 7.5% on each side of the diagonal).
- DWT levels for high-frequency content (k): This is related to both t_s and C, and represents the time and space granularity that is relevant for the overall comparison vs. the high frequency "noise" or anomalies.

V. Related Work

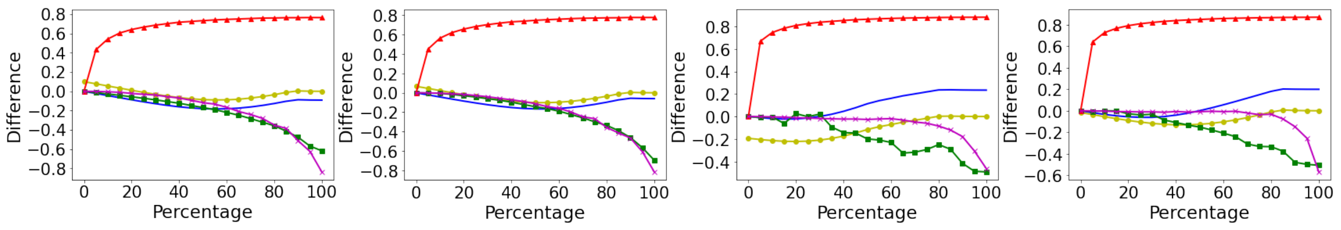
In addition to SSIM, another image similarity proposal is visual information fidelity (VIF) [18]. VIF models the wavelet coefficients as Gaussian scale mixtures and quantifies the mutual information between reference and target images. The feature similarity index (FSIM) [19] uses a combination of two feature maps for RRS. Reisenhofer [20] introduces a Haar wavelet-based perceptual similarity index (HaarPSI), which can be considered as a simplification of FSIM.

Many strategies have been explored to reduce the complexity of DTW [21]. In particular, other than the constant size band of Sakoe-Chiba [11], there is also the Itakura "parallelogram" approach, but it is generally worse than Sakoe-Chiba. Hauswirth et al. [5] analyze the performance of object-oriented programs by examining techniques for trace alignment and trace correlation. Reference [22] considers similarity for "uncertain" time series, i.e., time series where each sample can be considered to be drawn from a distribution. It computes



(a) HPC Trace A (b) HPC Trace B (c) Microsoft Trace A (d) Microsoft Trace B

Figure 10. Activeness differences for various perturbations.



(a) HPC Trace A (b) HPC Trace B (c) Microsoft Trace A (d) Microsoft Trace B

Figure 11. Finer detail differences for various perturbations

distance measure which is less than some threshold with a high probability. While this notion is useful, the lack of knowledge of the distribution makes this approach infeasible.

DWT has been explored extensively under different wavelets and filters [12]. There are also other transforms, such as Short-term Fourier Transform (STFT) [23] and stationary DWT (SDWT) [24], but we don't believe they are appropriate.

VI. Conclusions and Future Work

In this paper, we have defined a similarity index called SIST for comparing storage traces and compared it against other commonly used measures for traces from both object and block storage. We showed that SIST has a much better behavior compared to these measures and more directly relates to storage access issues. In the future, we will validate SIST against a variety of other storage traces, and attempt to correlate it with actual storage system performance for various use cases. We will also devise efficient algorithms for clustering traces based on the similarity metric and potentially other criteria relevant to storage system performance. We will explore the suitability of SIST for persistent memory and DRAM traffic as well.

References

[1] A. Alazzawe, A. Pal, and K. Kant, "Efficient big-data access: Taxonomy and a comprehensive survey," *IEEE transactions on big data*, Nov 2020.
[2] Y. Zhang *et al.*, "Osca: An online-model based cache allocation scheme in cloud block storage systems," in *USENIX ATC (2020).*
[3] D. Narayanan *et al.*, "Write off-loading: Practical power management for enterprise storage," *ACM Transactions on Storage (TOS)*, 2008.
[4] Z. Wang *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, 2004.
[5] M. Hauswirth *et al.*, "Automating vertical profiling," in *Proc. ACM conf. on Object-oriented programming, systems, languages, & applications*, 2005, pp. 281–296.
[6] Z. Wang *et al.*, "Multiscale structural similarity for image quality assessment," in *37th IEEE Asilomar conf. on Signals, Systems & Computers*, vol. 2, 2003, pp. 1398–1402.
[7] A. Liu *et al.*, "Image quality assessment based on gradient similarity," *IEEE Transactions on Image Processing*, 2011.
[8] M. A. Rahim Khan and M. Zakarya, "Longest common subsequence based algorithm for measuring similarity between time series: a new approach," *World Applied Sciences Journal*, 2013.
[9] M. Meinard, "Dynamic time warping" *Information retrieval for music and motion*, pp.69-84, 2007.
[10] M. Zhang and D. Pi, "A new time series representation model and corresponding similarity measure for fast and accurate similarity detection," *IEEE Access*, vol. 5, pp. 24 503–24 519, 2017.
[11] Z. Geler, *et al.*, "Dynamic time warping: Itakura vs sakoe-chiba," in *2019 IEEE INISTA Symp.*. IEEE, 2019, pp. 1–6.
[12] I. Popivanov and R. J. Miller, "Similarity search over time-series data using wavelets," in *Proceedings 18th international conference on data engineering*. IEEE, 2002, pp. 212–221.
[13] P. Chaovalit, *et al.*, "Discrete wavelet transform-based time series analysis and mining," *ACM Computing Surveys (CSUR)*, 2011.
[14] F. Z. Boito, *et al.*, "On server-side file access pattern matching," in *2019 (HPCS)*. IEEE, 2019, pp. 217–224.
[15] R. Bolze, *et al.*, "Grid'5000: A large scale and highly reconfigurable experimental grid testbed," *Intl. J. High Performance Computing Applications*, vol. 20, no. 4, pp. 481–494, 2006.
[16] OrangeFS, "The OrangeFS Project," http://www.orangefs.org/, 2022.
[17] MPI-IO Test, http://freshmeat.sourceforge.net/projects/mpiiotest, 2014.
[18] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on image processing*, 2006.
[19] L. Zhang, *et al.*, "Fsim: A feature similarity index for image quality assessment," *IEEE transactions on Image Processing*, 2011.
[20] R. Reisenhofer, *et al.*, "A haar wavelet-based perceptual similarity index for image quality assessment," *Signal Processing: Image Communication*, vol. 61, pp. 33–43, 2018.
[21] V. Kurbalija, *et al.*, "The influence of global constraints on dtw and lcs similarity measures for time-series databases," in *3rd Intel. conf. on Software, Services & Semantic Technologies*, Springer, 2011, pp. 67–74.
[22] Y. Zuo, *et al.*, "Similarity matching over uncertain time series," in *2011 7th Intl. IEEE conf. on Computational Intelligence and Security*, 2011.
[23] J. A. Moorer, "A note on the implementation of audio processing by short-term fourier transform," in *2017 IEEE WASPAA Workshop*. 2017.
[24] J. Lin, *et al.*, "Ssaw: A new sequence similarity analysis method based on the stationary discrete wavelet transform," *BMC bioinformatics*, 2018.