GraphLens: Mining Enterprise Storage Workloads Using Graph Analytics

Yang Zhou
Georgia Institute of Technology
yzhou@gatech.edu

Sangeetha Seshadri, Lawrence Chiu
IBM Almaden Research Center
{seshadrs, lchiu}@us.ibm.com

Ling Liu
Georgia Institute of Technology
lingliu@cc.gatech.edu

***Abstract*— Conventional methods used to analyze storage workloads have been centered on relational database technology combined with attributes-based classification algorithms. This paper presents a novel analytic architecture, GraphLens, for mining and analyzing real world storage traces. The design of our GraphLens system embodies three unique features. First, we model storage traces as heterogeneous trace graphs in order to capture diverse spatial correlations and storage access patterns using a unified analytic framework. Second, we employ and develop an innovative graph clustering method to discover interesting spatial access patterns. This enables us to better characterize important hotspots of storage access and understand hotspot movement patterns. Third, we design a unified weighted similarity measure through an iterative learning and dynamic weight refinement algorithm. With an optimal weight assignment scheme, we can efficiently combine the correlation information for each type of storage access patterns, such as random v.s. sequential, read v.s. write, to identify interesting spatial correlations hidden in the traces. Extensive evaluation on real storage traces shows GraphLens can provide scalable and reliable data analytics for better storage strategy planning and efficient data placement guidance.**

I. Introduction

Performance optimization in enterprise storage systems has primarily relied on the ability to isolate and control workloads that were relatively well understood [1], [2], [3], [4]. With virtualized environments and cloud implementations, enterprise storage systems have to support a mixture of a large number of disparate workloads from a variety of applications [5], [6], [7]. Thus, storage systems not only need to deal with changes within a single workload, but also have to respond to changes to the workload mix, enabling more efficient sharing of the underlying infrastructure. Although Flash, DRAM and newer high performance storage hardware hold the promise to alleviate the performance problem, to fully capitalize on the potential of these new storage devices, intelligence and automation are required to identify the right data to be placed on the right storage devices or storage tiers at the right time.

Trace analysis is recognized as a viable model to assist with characterizing workloads and gaining deeper insights into workload behavior. Conventional storage trace analysis is primarily carried out by the per-column based statistical analysis (single attribute based access pattern) or the row-based statistical analysis using vector similarity. Characterizing workloads in depth and from different levels of granularity of spatial dimension is challenging. The challenge can be more demanding when a single volume represents a varying mix of workloads and such workload mix may change over time. We argue that understanding similarity and causality of access patterns can offer many opportunities for optimization of performance such as intelligent data placement.

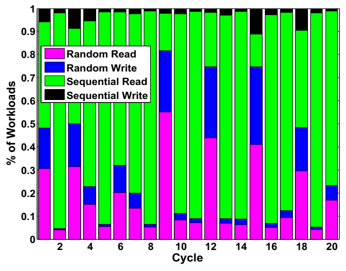
This paper presents a storage trace analysis architecture, GraphLens, for mining and analyzing real world storage traces. By innovative exploration of graph analytics, GraphLens offers three original contributions. First, storage traces are modeled as heterogeneous trace graphs in order to use a unified analytic model to study the complex spatial correlations among storage addresses at different levels of granularity in terms of their access patterns. Second, an innovative graph clustering method is developed to discover interesting spatial correlations by clustering storage addresses with a dynamic weighting scheme that continuously refines the weights on different access patterns of the storage addresses towards clustering convergence. This allows us to identify deeper spatial correlations among storage addresses beyond direct neighboring addresses. Third, our weight assignment scheme can efficiently combine the correlation information for each type of storage access patterns, such as random v.s. sequential, read v.s, write, to identify hotspots and their movements along diverse spatial dimensions of the trace. Extensive evaluation on three real storage traces demonstrates that GraphLens can perform deep trace analysis to derive new insights and new values for better storage planning and more efficient data placement strategies.

II. Motivation and Background

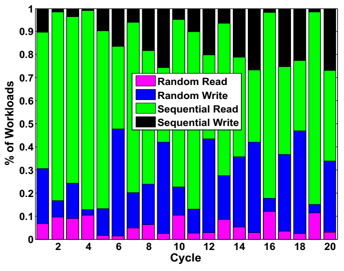
Related Work. Storage traces of production servers are valuable and critical in gaining insights on design, implementation and optimization of both modern storage servers and I/O intensive applications. However, mining and analyzing storage access patterns from real world workload traces has been scarce and superficial for a number of reasons, including difficulty in obtaining traces of production servers in diverse domains and absence of effective trace analysis models and algorithms that can infer deeper insight from limited traces of production systems. More seriously, many past trace-based studies have predated technology trends [8]. In the last decade, there are only a few studies [9], [1], [8], [10], [11] have dedicated to developing methodologies for characterization of real world workload traces. [9] analyzed four storage workload traces of production Windows servers with respect to block level requests, file access frequencies and read/write ratios. It performs trace analysis by measuring the spatial and temporal self-similarity based on variance and mean of storage log data. The approach in [4] assumes that workloads are well defined and can be cleanly isolated in order to train a classifier to identify workload phases using supervised learning. In addition, [8] studied the same large scale network file system workloads as reported in [10] using a multi-dimensional trace analysis methodology instead of single dimension based method. However, none of existing work has analyzed workload traces of production storage servers based on graph analytics. A unique advantage of modeling storage traces using graphs is the ability to conduct deep-analytics on both self-similarity and neighborhood similarity from both spatial and temporal perspectives. More importantly, GraphLens derives insights from traces with no assumption of a priori knowledge about workloads. Graph as an expressive data structure is popularly used to model structural relationship between objects in many application domains, ranging from web, social networks, biological networks to sensor networks [12], [13], [14], [18], [15], [16], [17], [19], [20], [21]. However, to the best of our knowledge, GraphLens is the first one that applies and extends graph mining to storage trace analysis. A unique feature of GraphLens is its ability to effectively identify fine-grained behavioral similarity across spatial and temporal dimensions using storage block traces.

Workload Traces. We analyze block-level traces collected from three large enterprise storage installations: a live banking environment, a retail backend system environment and an email server environment. Each of the traces consists of storage workloads collected over every 15 minute period (referred to as "cycle") for storage addresses, called "extents", each extent representing 1 GB logical address unit. The traces provide summary information on the number of random read, random write, sequential read and sequential write IO accesses over one week period (7 days). The only knowledge we know about each of these environments is that multiple workloads (such as applications and backup) may have been executing simultaneously. But we have no details regarding the exact nature of the workloads.

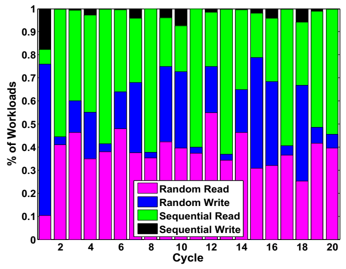
Figures 1 (a), (b) and (c) exhibit the distribution four access patterns observed on the three real world storage traces. For ease of presentation, we group the total of 2010 cycles into 20 cycle groups and summarize the access account for each extent in each cycle group. Figure 1 (a) presents the access activities on Bank Trace and "sequential read" is obviously the dominating access pattern. However, the access activities on Email Trace (Figure 1(b)) are often dominated by "sequential read" access pattern, with "random write" and "sequential read" as secondary behavior. In contrast, the workloads on Store Trace is mainly dominated by "random read" access pattern. We argue that by analyzing spatial, temporal and hot spot correlations from block-level traces we can provide broader and deeper insights for better tradeoffs in storage system design and implementation.



(a) Bank Trace



(b) Email Trace



(c) Store Trace

Figure 1. IO Workloads by Four Access Patterns

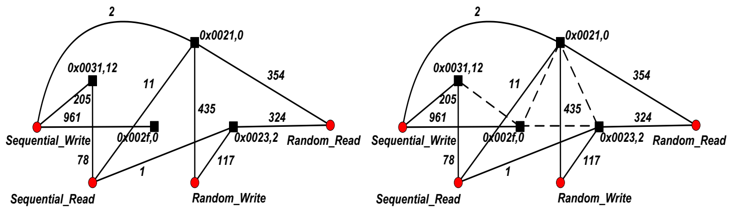
III. Overview

GraphLens by design aims at exploiting graph data analytic techniques on multi-dimensional storage traces to derive deep insights hidden in the storage logs, such as spatial access correlation and hot-spot dynamics. For example, how are different addresses accessed similarly within a cycle or amongst cycles? how does the access pattern of a storage address change between cycles? do spatial patterns interact with one another? what types of spatial access patterns are common in real world traces? and how hot spots move across extents (spatial)?

One approach for discovering and mining interesting correlations is to associate different storage addresses by utilizing their common attributes (access patterns), such as random v.s. sequential access and read v.s. write access. This motivates us to introduce two levels of abstractions for analyzing storage traces. First, we model storage traces as heterogeneous graphs. Second, we employ innovative graph analytic methods to measure and discover correlations among storage addresses. This two-level abstraction enables us to study the access correlations among different addresses by examining two types of vertices: structure vertices representing storage addresses (Lun (Volume), Extent) and attribute vertices (access patterns such as random, sequential, read or write) and their explicit and implicit relationships. Compared to naive vector-based correlation (record by record comparison), modeling storage access logs as a graph allows us to observe and understand not only those shallow correlations reflected from direct relationship between an address and its access pattern (attribute) but also enables us to derive deeper correlations that can only be inferred through reasoning over both direct and indirect correlations in a probabilistic manner.

Lun	Extent	RR	RW	SR	SW
0x0021	0	354	435	11	2
0x0023	2	324	117	1	0
0x002f	0	0	0	0	961
0x0031	12	0	0	78	205

Table I
A SAMPLE TRACE FOR A CYCLE



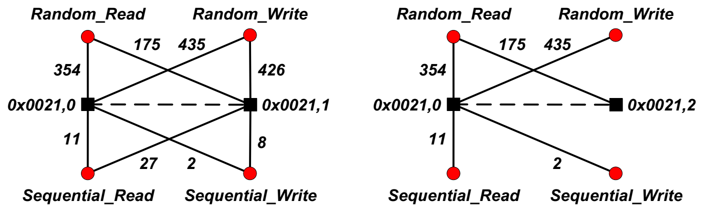
(a) Graph Model of a Single Cycle (b) Structure Relationship Deriving

Figure 2. An Illustrating Example of Heterogeneous Trace Graph

A. Modeling Traces as Graphs

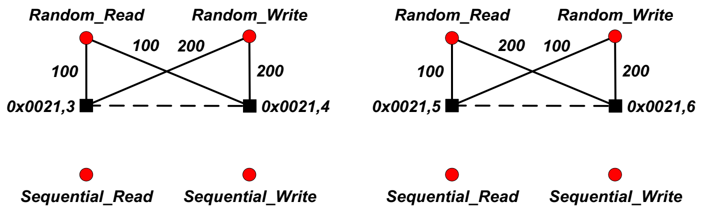
An enterprise storage trace recorded with multiple cycles is logged with a set of attributes (access patterns), such as random read (RR), random write (RW), random transfer (RT), sequential read (SR), sequential write (SW), sequential transfer (ST), etc. We model each cycle t_i of the storage log as **heterogeneous trace graph**, denoted as $G_i = (V, A, E_i, F_i)$, where $n = |V|$ specifies the size of the structure vertex set, $m = |A|$ defines the size of attribute vertex set in the trace, E_i denotes a set of structure edges between structure vertices and F_i represents the set of m types of attribute edges between V and A or between A and V. $v \in V$ is a structure vertex, representing a storage address and $a \in A$ denotes an attribute vertex associated to a structure vertex v, specifying an associated attribute of the address v. A structure edge $e \in E_i$ connects two structure vertices and an attribute edge $f \in F_i$ represents the relationship between a structure vertex and its associated attribute, weighted by the frequency of the corresponding access pattern within the given cycle. The initial heterogeneous graph G_i for each cycle t_i is a bipartite graph with only attribute edges. By employing GraphLens, we learn the correlations between structure vertices via their associated attributes. For instance, the more the access patterns shared by two addresses are, the greater the similarity between two addresses is.

Table I presents an example of a real storage trace. Each combination of Lun and Extent represents a unique storage address. RR, RW, SR, and SW correspond to four kinds of access patterns: random read, random write, sequential read, and sequential write, respectively. Figure 2 (a) is the heterogeneous graph representation of the sample trace in Table I. This trace graph has heterogeneous vertices: structure vertices (black square) represent the storage addresses, attribute vertices (red circle) specify 4 types of attributes: RR, RW, SR and SR. In addition, this trace graph has explicit attribute edges (solid lines), each representing a relationship between an structure vertex and one of its four types of attributes, and derived relationships (dashed lines) that represent the spatial correlation between two structure vertices, as shown in Figure 2 (b). Although the four address vertices have no direct correlations, we can learn the spatial correlations among different structure vertices because they can be reached by traversing the graph via attribute vertices.



(a) More Paths (b) Less Paths

Figure 3. Summarization of all Possible Paths



(a) Weight Match (b) Weight Mismatch

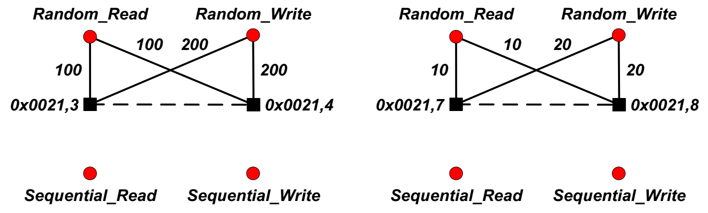
Figure 4. Attribute Weight Match

Case 1: summarization of all paths between any pair of extents. In Figure 3 (a), there exists four 2-hop paths between extents "0x0021, 0" and "0x0021, 1" through four attribute vertices respectively. In comparison to Figure 3 (b), there is only one 2-hop path between two extents through RR. We make the following observation: extents "0x0021, 0" and "0x0021, 1" are more similar than extents "0x0021, 0" and "0x0021, 2" since there are more reachable paths between the first two extents.

Case 2: attribute differentiation by attribute weight match. For both Figure 4 (a) and Figure 4 (b), two extents are reachable by two 2-hop paths through RR and RW respectively. However, the two addresses in Figure 4 (b) on each of RR and RW have diverse edge weights. Thus, extent "0x0021, 3" and extent "0x0021, 4" are more similar than extent "0x0021, 5" and extent "0x0021, 6" because the first pair of extents not only have the same access patterns (RR and RW) but also have the same access counts (100 for RR and 200 for RW).

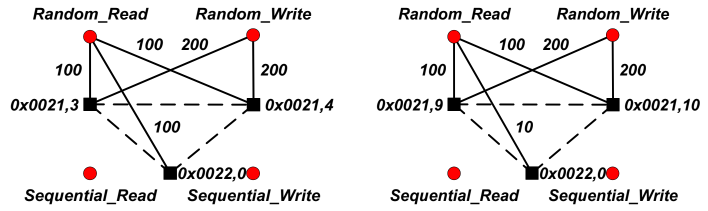
Case 3: attribute differentiation by attribute weight significance. In both Figure 5 (a) and Figure 5 (b), two extents are reachable by two 2-hop paths through RR and RW respectively and the corresponding attribute edge weights are the same respectively. However, the corresponding attribute edge weights (100 for RR and 200 for RW) in Figure 5 (a) are larger than that (10 for RR and 20 for RW) in Figure 5 (b). Thus, two extents in Figure 5 (a) are more similar than two extents in Figure 5 (b).

Case 4: summarization of all possible k-hop paths between pairwise extents. In the above cases, we only consider 2-hop paths between extents, i.e., direct relationships between extents. However, we should consider all possible k-hop paths, i.e., both direct and indirect relationships, to achieve a comprehensive and fair comparison result when



(a) Significant Weights (b) Insignificant Weights

Figure 5. Attribute Weight Significance



(a) Significant 4-hop Path (b) Insignificant 4-hop Path

Figure 6. k-hop Path

we calculate the similarity scores between two extents. The only difference between Figures 6 (a) and (b) is the attribute edge weights between extent "0x0022, 0" and access pattern RR. Note that there is a 4-hop path between extent "0x0021, 3" (or "0x0021, 9") and extent "0x0021, 4" (or "0x0021, 10"): "0x0021, 3" (or "0x0021, 9") $\rightarrow$ "Random_Read" $\rightarrow$ "0x0022, 0" $\rightarrow$ "Random_Read" $\rightarrow$ "0x0021, 4" (or "0x0021, 10"). We argue that extents "0x0021, 3" and "0x0021, 4" in Figure 6 (a) have larger attribute edge weights and thus are more similar than extents "0x0021, 9" and "0x0021, 10" in Figure 6 (b). The similarity between two extents depends on not only their direct relationships (their own access patterns and access counts) but also their indirect relationships (predecessors' and successors' access patterns and access counts).

B. Trace Analysis with GraphLens

GraphLens performs trace analysis to derive deep insights on spatial/temporal access correlations and hotspot characterization in two phases: (1) extent similarity computation and (2) spatial based graph clustering.

In Phase I, we measure pairwise extent similarity within a cycle in terms of two factors: how similar their access patterns (attribute) are (direct correlation) and how similar their k-hop neighbor extents are in terms of their access patterns. In order to capture the spatial access similarity between storage addresses in terms of both direct and indirect correlations, we introduce a **unified weighted extent neighborhood random walk similarity measure**. It is used to measure the closeness between extents based on all four types of attribute edges, each with an initial weight. This unified similarity measure captures the connectivity and the vicinity between extents (structure vertices).

In Phase II, we utilize this unified similarity measure to cluster all extents in trace graph G_i into k_i clusters with initial centroids and initial weights. We employ a dynamic weight tuning method combined with an iterative refinement mechanism of centroid update and vertex assignment to quantitatively estimate the importance of various types of attributes and attribute links in terms of their contribution to the clustering process. Formally, given a heterogeneous trace graph G_i for cycle t_i, the problem of **spatial extent clustering** is to partition the objective extents V into k_i disjoint clusters $C_1, C_2, \ldots, C_{k_i}$, where $i \in \{1, 2, \ldots, N\}$, N is the number of cycles, $V = \bigcup_{p=1}^{k_i} C_p$ and $C_p \bigcap C_q = \phi$ for $\forall p, q, 1 \le p, q \le k_i, p \ne q$, to ensure: (1) the extents within each cluster have larger similarity scores, while the extents in different clusters have smaller similarity scores; and (2) the extents within clusters have low diversity on access patterns, while the extents in different clusters have highly diverse access patterns. The spatial based clustering can indicate where the hotspots are and how such hotspots move across extents (along spatial dimension).

IV. Methodology

This section describes the two-phase correlation analysis in GraphLens: extent-similarity based spatial clustering.

A. A Unified Spatial Similarity Measure

In GraphLens, we propose to use a unified similarity measure based on the neighborhood random walk model to infer the spatial access correlations between extents and the temporal access correlation between cycles. In the heterogeneous graph, some vertices are close to each other while some other vertices are far apart based on connectivity. Random walk distance can accurately capture such pairwise vertex closeness. Recall the example in Figure 2, there exists a random walk path between two extents $v_1, v_2 \in V$ if (1) v_1 and v_2 have the same neighbor extent $v_3 \in V$; or if (2) v_1 and v_2 have the same attribute $a \in A$. If there are multiple random walk paths connecting v_1 and v_2, then they should be very close in terms of similar access patterns. On the other hand, if there are very few or no paths between v_1 and v_2, then they should be far apart in terms of diverse access patterns.

Definition 1: [Transition Probability] Let V be the set of n extents, A be the set of m associated attributes, the transition probability matrix $P(i)$ of a heterogeneous graph G_i for cycle t_i is defined as follow.

$$P(i) = \begin{bmatrix} P_{SS} & P_{SA} \\ P_{AS} & P_{AA} \end{bmatrix} \qquad (1)$$

where P_{SS} is an $n \times n$ matrix representing the transition probabilities between structure vertices; an $n \times m$ matrix P_{SA} specifies the transition probabilities from structure vertices to attribute vertices; P_{AS} denotes the transition probabilities from attribute vertices to structure vertices; and P_{AA} is an $m \times m$ matrix representing the transition probabilities between attribute vertices.

In the context of heterogeneous trace graph, submatrices P_{SS} and P_{AA} have all zero entries since there is no connection between structure vertices or between attribute vertices. However, the corresponding submatrices in the power of $P(i)$, such as $P(i)^2, P(i)^3, \ldots$, may contain non-zero elements since there may exist possible paths through other vertices.

To capture the fact that each type of attribute edges may have different degrees of contribution in random walk similarity, we assign an individual weight for each type of attribute edges. Initially, all weights are set to equal value, say 1.0. We design a dynamic weight tuning method to produce an optimal weight assignment for all types of links in the next section. Based on this weight assignment, each submatrix in $P(i)$ is defined as follow.

$$P_{SA}(p,q) = \begin{cases} \frac{\alpha_q e_{pq}}{\sum_{r=1}^{m_i} \alpha_r e_{pr}}, & if (v_p, a_q) \in F_i \\ 0, & otherwise \end{cases} \tag{2}$$

where e_{pq} represents the count of access pattern a_q that storage extent v_p has in the given cycle. For instance, a storage extent of "0x0021, 0" has the count of 354 on "random read" in the example cycle of Figure 2(a). α_q denotes the weight of attribute edges from any of the structure vertices to attribute vertex a_q. Since each row of transition probability matrix should sum to 1, we employ the row-wise normalization for P_{SA}.

$$P_{AS}(p,q) = \begin{cases} \frac{\alpha_p e_{pq}}{\sum_{r=1}^{n_i} \alpha_p e_{pr}} = \frac{e_{pq}}{\sum_{r=1}^{n_i} e_{pr}}, & if (a_p, v_q) \in F_i \\ 0, & otherwise \end{cases} \tag{3}$$

where e_{pq} specifies the count of an access pattern a_p achieved by a storage extent v_q. α_p denotes the weight of attribute edges from attribute vertex a_p to structure vertices. Different from the normalization in P_{SA}, the count on pattern a_p by extent v_q is normalized by the counts on pattern a_p by all extents.

A random walk on a heterogeneous trace graph G_i is performed in the following way. Suppose a particle starts at a certain vertex v_0 and walks to a vertex v_s in the s^{th} step and it is about to move to one of the neighbors of v_s, denoted as $v_t \in N(v_s)$, with the transition probability $P(i)(s,t)$, where $N(v_s)$ contains all neighbors of vertex v_s. The vertex sequence of the random walk is a Markov chain. The probability of going from v_i to v_j through a random walk of length l can be obtained by multiplying the transition probability matrix l times.

Definition 2: [Unified Random Walk Similarity] Let $P(i)$ be the transition probability of a heterogeneous trace graph G_i, l be the length that a random walk can go, and $c \in (0,1)$ be the restart probability, the unified random walk similarity $s(u,v)$ from vertex $u \in V \bigcup A$ to vertex $v \in V \bigcup A$ in G_i is defined as follow.

$$s_i(u,v) = \sum_{\substack{\tau: u \rightsquigarrow v \\ length(\tau) \leq l}} p(\tau) c (1-c)^{length(\tau)} \tag{4}$$

where τ is a path from u to v whose length is $length(\tau)$ with transition probability $p(\tau)$ which is equal to the multiplication of the transition probability of each step in path τ. $s_i(u,v)$ reflects the extent closeness within cycle t_i based on multiple types of attribute information.

The matrix form of the unified random walk similarity is given as follow.

$$R(i) = \sum_{\gamma=1}^{l} c(1-c)^{\gamma} P(i)^{\gamma} \tag{5}$$

where an $(n+m) \times (n+m)$ matrix $R(i)$ sums over the dependency of all possible paths between two extents. Each entry $s_i(u,v)$ in $R(i)$ measures the similarity between extent vertex u and extent vertex v within cycle t_i.

B. Spatial Extent Clustering

Our extent clustering framework, E-CLUSTER, partitions extents in a heterogeneous trace graph G_i into k_i densely connected clusters. Due to space limit, we will briefly illustrate the extent clustering framework by focusing on different points. E-CLUSTER follows the traditional *K-Medoids* clustering method [22] by using the unified random walk similarity $R(i)$ with k_i extents of high degree as the initial centroids and the initial weights $\alpha_{11}^0, \ldots, \alpha_{im}^0$ as an input. At each iteration, based on unified extent random walk similarity scores, we select the most centrally located extent in each of the k_i clusters to obtain k_i new centroids, and assign the rest of extents to their closest centroids. The objective of clustering is to maximize intra-cluster similarity and minimize inter-cluster similarity. The weight update method computes the weighted contributions of each kind of attribute links to both clustering convergence and clustering objective, and updates m weights accordingly after each iteration. This process is repeated until convergence.

Thus the graph clustering problem can be reduced to three subproblems: (1) cluster assignment, (2) centroid update and (3) weight adjustment, each with the goal of maximizing the objective function. The first two problems are common to all partitioning clustering algorithms. Hence we only focus on the third subproblem, weight adjustment in this paper.

We employ a dynamic weight adjustment method to iteratively improve the spatial extent clustering objective. Let $\alpha_{ip}^t (p = 1, \ldots, m)$ be the weights of attribute edges between structure vertices and attribute vertex a_p in the transition probability $P(i)$ of G_i in the t^{th} iteration. All α_{ip}^0s are first initialized as 1.0. We then iteratively adjust α_{ip}^{t+1} with an increment $\triangle\alpha_{ip}^t$, which denotes the weight update of attribute edges between structure vertices and attribute vertex a_p in $P(i)$. The attribute weight α_{ip}^{t+1} in the $(t+1)^{th}$ iteration is computed as

$$\alpha_{ip}^{t+1} = \frac{1}{2}(\alpha_{ip}^t + \triangle\alpha_{ip}^t) \tag{6}$$

To determine the extent of weight increment $\triangle\alpha_{ip}$, we design a majority vote mechanism: if a large portion of structure vertices within each cluster share the same attribute vertices with similar access counts, which means it has a good clustering tendency, then the structure weight α_{ip} should be increased; on the other hand, if structure vertices within clusters have a very random distribution on attribute or have quite diverse access counts, then the weight α_{ip} should be decreased. We define a *vote* measure which

DataSet Name	Total Size (in GB)	Duration (in cycles)
Bank Trace	8,097	2,013
Store Trace	7,902	2,008
Email Trace	1,599	2,011

Table II
TRACE DATASET SUMMARY

determines whether two structure vertices u and v have similar attributes.

$$vote_{ip}(u,v) = \begin{cases} 1, & 1 - \dfrac{|a_{ip}(u) - a_{ip}(v)|}{|a_{ip}(u) + a_{ip}(v)|} > \epsilon \\ 0, & otherwise \end{cases} \tag{7}$$

where $a_{ip}(u)$ specifies the count achieved by u on attribute a_p in G_i. A positive number ϵ denotes a threshold of similar extent of u and v on a_p. The weight increments $\triangle\alpha_{ip}^t$ are then calculated as

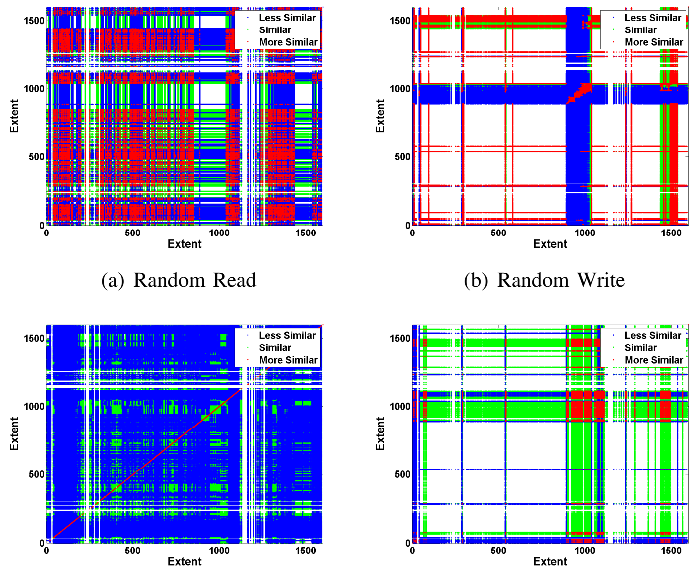
$$\triangle\alpha_{ip}^t = \frac{\sum_{j=1}^{k_i} \sum_{v \in C_j} vote_{ip}(c_j, v)}{\frac{1}{m}\sum_{q=1}^{m} \sum_{j=1}^{k_i} \sum_{v \in C_j} vote_{iq}(c_j, v)} \tag{8}$$

An important property of the weight self-adjustment mechanism is that the updated weights should increase the clustering objective. The detailed proof is omitted due to space limit. We will briefly illustrate this property qualitatively: if a large number of structure vertices within clusters have the similar access counts on a_p, then the weight is increased, i.e., $\alpha_{ip}^{t+1} > \alpha_{ip}^t$; on the other hand, if structure vertices within clusters have quite different access counts on a_p, the weight is then decreased, i.e., $\alpha_{ip}^{t+1} < \alpha_{ip}^t$. There must be some weights with increasing updates and some other weights with decreasing updates, since $\sum_{p=1}^{m} \alpha_{ip} = m$ is a constant. Due to some increased weights, the random walk similarities between pairwise endpoints of attribute edges with the increased weight will be further increased. As a result, these vertices tend to be clustered into the same cluster, thus increasing the clustering objective. Due to space constraint, we omit the pseudo code of our E-Cluster algorithm in this section.

V. EXPERIMENTAL EVALUATION

In this section we discuss insights obtained by employing GraphLens on three different real world traces from three perspectives: spatial extent correlation analysis, temporal cycle correlation analysis and hotspot characterization. For ease of presentation, we divide similarity scores between 0 and 1 into three groups: "More Similar" (red, [0.9, 1]), "Similar" (green, (0.5, 0.9)) and "Less Similar" (blue, [0, 0.5]). In addition, the white area represents the extents without any activities in the given cycle.

We use the three storage trace datasets described in Section II. The trace characteristics are summarized in Table II. We build a heterogeneous trace graph for the workloads in each cycle where structure vertices represent the combinations of Lun and Extent, attribute vertices specify four access patterns of RR, RW, SR and SW. Attribute edges denote the relationships between structure vertices and attribute vertices, weighted by the corresponding access count to each data unit within the cycle. We construct 2,013, 2,008 and 2,011 trace graphs for three storage traces respectively.



(a) Random Read (b) Random Write

(c) Sequential Read (d) Sequential Write

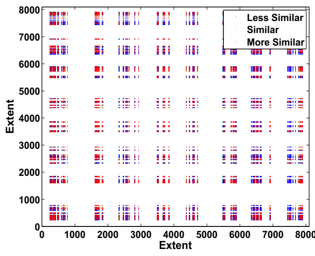
Figure 7. Extent Similarity on Email Trace

A. Spatial Correlation Analysis

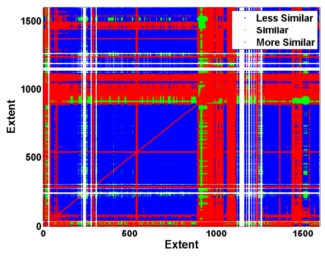
Figures 7 (a), (b), (c) and (d) represent the spatial similarity matrix based on each access pattern for the email trace. Figure 7 (a) shows that most of spatial similarity between extents based on random reads shows several regions of strong similarity. Next observe that 7 (b) and (c) which represents the random write and sequential read patterns are dominated by weak similarity. It is highly likely that most people usually access their emails at least once each day and mainly read the emails without any reply. Thus, most of extents are very similar based on random reads due to frequent as well as random accesses. Figure 7 (d) is dominated by average similarity. Sequential writes typically represent backup and replication activity in these environments and extents would be expected to be similar in behavior for this access pattern. However, since such activities are infrequent, the number of accesses are very low, leading to a decision of "Less Similar". This leads us to the first set of observations:

- **Observation 1:** similar behavior is exhibited both within and across volumes.
- **Observation 2:** spatial similarity varies by the dimension under consideration.
- **Observation 3:** data expresses stronger similarity under the random read access pattern.

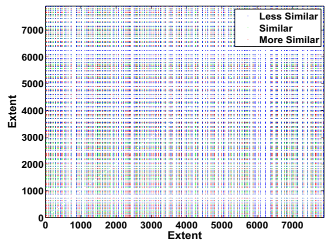
Figures 8 (a), (b) and (c) exhibit the unified random walk similarity matrix on different trace datsets. We use our dynamic vote-based weight tuning method in Section IV-B to learn an optimal weight assignment for four types of attribute links: RR, RW, SR and SW, to achieve high intra-cluster similarity and low inter-cluster similarity, i.e., extents within clusters have similar access patterns, while the extents in different clusters have diverse access patterns. The similarity matrix in Figure 8 (a) is similar to the similarity matrix we have observed in the extent similarity case. This



(a) Bank Trace

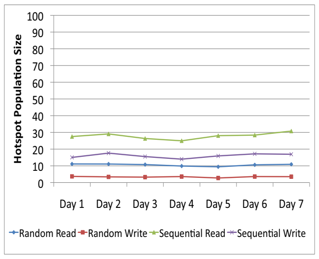


(b) Email Trace

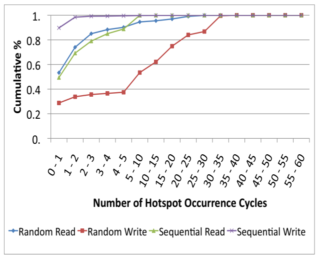


(c) Store Trace

Figure 8. Unified Extent Similarity on Different Traces



(a) Bank Trace: Population Size



(b) Bank Trace: Burstiness

Figure 9. Bank Trace

demonstrates that the unified random walk similarity on Bank Trace is dominated by the access pattern of random read. Due to space constraint, we omit the extent similarity on Bank trace and on Store trace.

Comparing with Figures 7 (b) and (d), the similarity matrix in Figure 8 (b) mainly depends on the access patterns of random write and sequential write. Since most of extents do not have these two kinds (random write and sequential write) of access activities extents that exhibit these activities are more alike each other and more different from extents that do not exhibit the activity. Figure 8 (c) shows the unified random walk similarity matrix on Store Trace, which is a relatively random distribution for each of three kinds of similarities due to the lack of clear distinctions between extent access patterns. This leads us to the following observations.

- **Observation 4:** when all data exhibits all types of access pattern, the strongest access pattern (which is most often random read) dominates the similarity metric. This implies that data placement taking only random read patterns into consideration is likely to provide good results.
- **Observation 5:** when access type distributions are not uniform across all extents, extents that exhibit more rare access patterns have stronger similarity under a unified metric. This implies that under such circumstances, data placement must first consider the unified metric to identify broader distinctions between extents and then consider random reads as a secondary metric.
- **Observation 6:** when unified extent similarity weighted on all access patterns exhibit a relatively random distribution as shown in Figure 8 (c), this indicates that there is no need to further explore attribute-specific spatial access patterns.

B. *Hotspot Characterization*

Hotspots can be defined as regions that have relatively higher activity (hence "temperature" or "heat") in comparison to its surroundings. Understanding hotspot characteristics is essential to data placement strategies, such as caching at host or storage server by utilizing the "recency" [23], [2], and tiering by exploring the "frequency" aspect [24], [3].

By using GraphLens, extents are classified into "Hot", "Warm" and "Cold" clusters for each cycle. An extent that appears in the hot cluster at time t is referred to as a hotspot at time t. A single extent can exhibit hotspot behavior in multiple cycles and multiple extents can exhibit hotspot behavior in the same cycle. Our dynamic weight assignment and update at each clustering iteration reduce the possible bias introduced by a single attribute dominating the clustering outcome. We use the following measures to classify hotspots from the temporal and spatial clustering analysis results:

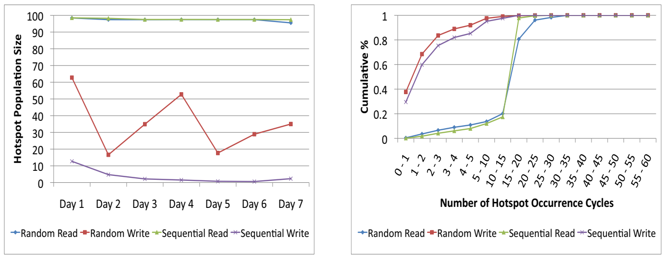
- **Population Size:** a summary measure which describes the number of unique extents that exhibit hotspot behavior within a window (24 hours in this study) of observation i.e. size of the hotspot.
- **Intra-window stability or Burstiness:** frequency distribution of number of hotspot occurrences for a each unique extent *within a window of observation*. This measure is indicative of the burstiness and durability of hotspot behavior within each time window.

Due to space constraint, we omit the hotspot characterization of Email Trace.

Banking Transactions Workload

Figure 9(a) shows the variation in hotspot population size over 7 days for four different access patterns. The x-axis and the y-axis represent the day and the percentage of total dataset population that exhibited hotspot behavior at any time during the day for a specific access pattern. We see that the hotspot population size remains fairly stable from day to day for all workloads. Random Read (3%) and Random write (10%) are most stable. Sequential Read (30%) and Sequential Write (17%) activities span relatively larger population sizes but remain small compared to the total dataset size.

Figure 9(b) shows the frequency distribution during a 24 hour period for which an extent exhibited hotspot behavior. Random write workload exhibits the least burstiness with nearly 63% of the hotspots lasting longer than 75 minutes. Random read and sequential read hotspots are relatively more bursty with only 10% of the sequential read and random read hotspots lasting longer than 75 minutes. On



(a) Store Trace: Population Size (b) Store Trace: Burstiness

Figure 10. Store Trace

the other hand, sequential write is the most bursty with 90% of hotspots lasting less than 15 minutes in a day and nearly all hotspots lasting less than 30 minutes. We conjecture that random write workloads for this application are probably best serviced by a tiering strategy. On the other hand, random reads and sequential reads contain a mix of bursty and stable hotspot behavior, a combination of caching (to catch bursty hotspots) and tiering (to catch more long term behavior) could be used. Sequential writes exhibit highly bursty behavior, which could be addressed with prefetch caching.

Store Backend Workload

In the Store trace (Figure 10 (a)), almost all data exhibits hotspot behavior at some point during the day. Sequential write and random write hotspots are limited to a smaller fraction of the dataset (3% and 35% respectively). However sequential write and random write hotspots show large variations in population size over the week.

Figure 10 (b) shows the frequency distribution during a 24 hour period for which an extent exhibited hotspot behavior. Sequential reads and random read patterns show very identical behavior with 80% of the extents exhibit hot spot behavior for nearly 4 to 5 hours a day. In comparison, Sequential writes and random writes are relatively bursty with nearly 60% and 70% respectively of the hotspots exhibiting hotspot behavior for less than 30 minutes in a day. Given the large population size and the low burstiness, these access patterns may be effectively addressed by provisioning a high performance tier (assuming that 8TB of cache may not be viable option at every host and population such a large cache may itself take several hours).

VI. Conclusions

We have presented a novel graph analytics framework, GraphLens, for mining and analyzing real storage traces. We model storage traces as heterogeneous trace graphs to incorporate multiple complex and heterogeneous factors into a unified analytic framework. An innovative graph clustering method is proposed to identify and discover spatial correlations and hotspot characterization. We design an dynamic weight tuning method to combine multiple correlations into a unified similarity measure with optimal weights.

Acknowledgment

This research is partially supported by grants from NSF CISE NetSE program, SaTC program, I/UCRC, an IBM faculty award and a grant from Intel ICST on Cloud Computing.

References

[1] M. Bhadkamkar, J. Guerra, L. Useche, S. Burnett, J. Liptak, R. Rangaswami, and V. Hristidis, "Borg: Block-reorganization for self-optimizing storage systems," in *FAST*, 2009, 183–196.

[2] Y. Zhang, G. Soundararajan, M. W. Storer, L. N. Bairavasundaram, S. Subbiah, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Warming up storage-level caches with bonfire," in *FAST*, 2013.

[3] G. Zhang, L. Chiu, C. Dickey, L. Liu, P. Muench, and S. Seshadri, "Automated lookahead data migration in ssd-enabled multi-tiered storage system," in *MSST*, 2010, 1–6.

[4] P. Pipada, A. Kundu, K. Gopinath, C. Bhattacharyya, S. Susarla, and P. C. Nagesh, "Loadiq: Learning to identify workload phases from a live storage trace," in *HotStorage'12*.

[5] V. Tarasov, D. Hildebrand, G. Kuenning, and E. Zadok, "Virtual machineworkloads: The case for new benchmarks for nas," in *FAST*, 2013.

[6] A. Gulati, C. Kumar, I. Ahmad, and K. Kumar, "Basil: Automated io load balancing across storage devices," in *FAST'10*.

[7] N. Park, I. Ahmad, and D. J. Lilja, "Romano: Autonomous storage management using performance prediction in. multi-tenant datacenters," in *SoCC*, 2012, pp. 21:1–21:14.

[8] Y. Chen, K. Srinivasan, G. R. Goodson, and R. H. Katz, "Design implications for enterprise storage systems via multi-dimensional trace analysis," in *SOSP*, 2011, pp. 43–56.

[9] S. Kavalanekar, B. L. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production windows servers," in *IISWC*, 2008, pp. 119–128.

[10] A. W. Leung, S. Pasupathy, G. Goodson, and E. L. Miller, "Measurement and analysis of large-scale network file system workloads," in *ATC*, 2008, pp. 213–226.

[11] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of backup workloads in production systems," in *FAST*, 2012.

[12] Y. Zhou, L. Liu, C.-S. Perng, A. Sailer, I. Silva-Lepe, and Z. Su, "Ranking services by service network structure and service attributes," in *ICWS*, 2013.

[13] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *WWW*, 1998, pp. 107–117.

[14] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, pp. 604–632, 1999.

[15] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *WWW*, 2003, pp. 640–651.

[16] Y. Zhou and L. Liu, "Social influence based clustering of heterogeneous information networks," in *KDD*, 2013.

[17] H. Hwang, V. Hristidis, and Y. Papakonstantinou, "Objectrank: a system for authority-based search on databases," in *SIGMOD*, 2006, pp. 796–798.

[18] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *TPAMI*, vol. 22, no. 8, 2000, pp. 888–905.

[19] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: a structural clustering algorithm for networks," in *KDD*, 2007.

[20] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," in *VLDB*, 2009, pp. 718–729.

[21] Y. Zhou, H. Cheng, and J. X. Yu, "Clustering Large Attributed Graphs: An Efficient Incremental Approach," in *ICDM*, 2010.

[22] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," *Statistical Data Analysis based on the L1 Norm*, pp. 405–416, 1987.

[23] S. Byan, J. Lentini, A. Madan, L. Pabon, M. Condict, J. Kimmel, S. Kleiman, C. Small, and M. Storer, "Mercury: Host-side flash caching for the data center," in *MSST*, 2012, 1–12.

[24] J. Guerra, H. Pucha, J. Glider, W. Belluomini, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," in *FAST*, 2011, pp. 20–20.