Analyzing Enterprise Storage Workloads With Graph Modeling and Clustering

Yang Zhou, *Member, IEEE*, Ling Liu, *Fellow, IEEE*, Sangeetha Seshadri, *Member, IEEE*, and Lawrence Chiu

Abstract*—Utilizing graph analysis models and algorithms to exploit complex interactions over a network of entities is emerging as an attractive network analytic technology. In this paper, we show that traditional column or row-based trace analysis may not be effective in deriving deep insights hidden in the storage traces collected over complex storage applications, such as complex spatial and temporal patterns, hotspots and their movement patterns. We propose a novel graph analytics framework, GraphLens, for mining and analyzing real world storage traces with three unique features. First, we model storage traces as heterogeneous trace graphs in order to capture multiple complex and heterogeneous factors, such as diverse spatial/temporal access information and their relationships, into a unified analytic framework. Second, we employ and develop an innovative graph clustering method that employs two levels of clustering abstractions on storage trace analysis. We discover interesting spatial access patterns and identify important temporal correlations among spatial access patterns. This enables us to better characterize important hotspots and understand hotspot movement patterns. Third, at each level of abstraction, we design a unified weighted similarity measure through an iterative dynamic weight learning algorithm. With an optimal weight assignment scheme, we can efficiently combine the correlation information for each type of storage access patterns, such as random versus sequential, read versus write, to identify interesting spatial/temporal correlations hidden in the traces. Some optimization techniques on matrix computation are proposed to further improve the efficiency of our clustering algorithm on large trace datasets. Extensive evaluation on real storage traces shows GraphLens can provide broad and deep trace analysis for better storage strategy planning and efficient data placement guidance. GraphLens can be applied to both a single PC with multiple disks and a distributed network across a cluster of compute nodes to offer a few opportunities for optimization of storage performance.

***Index Terms*—Heterogeneous Trace Graph, Random Walk, Unified Weighted Spatial/Temporal Similarity, Spatial Extent Clustering, Temporal Cycle Clustering, Full-rank Approximation.**

I. Introduction

PERFORMANCE optimization in enterprise storage systems has always relied heavily on the ability to isolate and control workloads that were relatively well understood [2]–[5]. With virtualized environments and cloud implementations, enterprise storage systems see a mix of a large number of disparate workloads from a varying set of applications [6]–[8]. The systems not only need to deal with changes within a single workload, but also need to deal with changes to the workload mix while the underlying infrastructure is shared. While Flash, DRAM and newer high performance storage hardware can help alleviate performance problem, intelligence and automation are still required to identify the right data to be placed on these devices.

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 granularities of spatial and temporal dimensions 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.

In this paper we present GraphLens, a novel graph analytics framework for mining and analyzing real world storage traces. GraphLens offers the following 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, temporal and spatial-temporal correlations among storage addresses at different levels of granularity in terms of their access patterns. Second, an innovative graph clustering method is developed, which conduct storage trace analysis via two tiers of clustering abstractions. At the first tier, we discover interesting spatial correlations by clustering storage addresses with a dynamic weighting scheme that continuously refines the weights to 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. At the second tier, we employ temporal clustering abstraction to discover important temporal correlations by utilizing significant spatial correlations. We introduce a unified weighted temporal similarity measure through an iterative dynamic weight-learning algorithm. Our entropy-based optimal 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

Manuscript received August 15, 2015; revised October 15, 2015; accepted January 10, 2016. Date of publication February 3, 2016; date of current version March 15, 2016. This work was 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. A preliminary version of this paper was presented at the IEEE International Congress on Big Data, Anchorage, AK, USA, June 27–July 2, 2014 [1].

Y. Zhou and L. Liu are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: yzhou@gatech.edu; lingliu@cc.gatech.edu).

S. Seshadri and L. Chiu are with IBM Almaden Research Center, San Jose, CA 95120 USA (e-mail: seshadrs@us.ibm.com; lchiu@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2016.2525478

both spatial and temporal dimensions of the trace. We also propose the optimization technique of full-rank approximation based on the Matrix Neumann Series [9] to speed up the matrix computation for random walk similarity calculation. It reduces the number of matrix multiplication from $(l-1)$ to $(\lceil \log(l+1) \rceil + 1)$ where l is the length limit of the random walks. 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 strategy planning and efficient data placement guidance.

This intelligent tool can be applied to both a single PC with multiple disks and a distributed network across a cluster of compute nodes to offer a few opportunities for optimization of storage performance such as better storage strategy planning and efficient data placement guidance: (1) migrating the data with similar access patterns to the same disks/nodes, in order to improve the performance of disk I/O and communication I/O; (2) migrating the data frequently accessed to the disks/nodes with fast storage devices, such as SSD, and migrating the data infrequently accessed to the disks/nodes with slow storage devices, such as HDD, to achieve a good balance between improving system performance and reducing overall disk/communication cost; (3) during weekends and holidays, increasing the amount of backend jobs on the "Cold" disks/nodes with few access activities to improve resource utilization, while decreasing the amount of backend jobs on the "Hot" disks/nodes to maintain system performance during workdays.

II. Background and Motivation

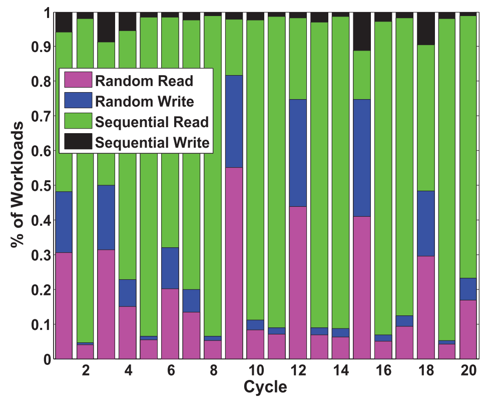
Storage Trace Analysis. Storage traces of production servers are valuable and critical in gaining insights on design, implementation and optimization of both modern storage servers and I/O intensive applications. However, mining and analyzing storage access patterns from real world workload traces has 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 [10]. In the last decade, there are only a few studies [2], [10]–[14] have dedicated to developing methodologies for characterization of real world workload traces. Kavalanekar *et al*. [11] 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 [5] 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, Chen *et al*. [10] studied the same large scale network file systems workloads as reported in [12] using a multi-dimensional trace analysis methodology instead of single dimension based method. Tarasov [14] demonstrated the challenges of evaluating complex storage systems in his Ph.D. dissertation and proposed a Multi-Dimensional Histogram workload analysis technique to design a variety of evaluation tools for analyzing workloads and system behaviors. 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.

Heterogeneous Network Analysis. Recent studies on heterogeneous social network analysis combine links and content into heterogeneous information networks to improve the quality of querying, ranking, clustering and classification [15]–[21]. Taskar *et al*. [15] proposed a general class of models for classification and clustering in relational domains that capture probabilistic dependencies between related instances in a relational database containing both attributes and links. Yang *et al*. [16] proposed a unified model to combine link and content analysis for community detection. The conditional link model and the discriminative content model are combined via a probabilistic framework through the shared variables of community memberships. Ji *et al*. [17] groups objects into pre-specified classes, while generating the ranking information for each type of object in a heterogeneous information network. It is therefore beneficial to integrate classification and ranking in a simultaneous, mutually enhancing framework. Yu *et al*. [18] presented a query-driven discovery system for finding semantically similar substructures in heterogeneous networks. A filter-and-verification search framework is proposed to generate promising subgraph candidates using off-line indices, and verify candidates with a recursive pruning matching process. Zhou *et al*. [19] proposed a reinforcement algorithm is provided to tightly integrate ranking and clustering by mutually and simultaneously enhancing each other. Zhou and Liu [20] presented an activity-edge centric multi-label classification framework for analyzing heterogeneous information networks by doing multi-label classification of friendship multigraph based on activity-based edge classification.

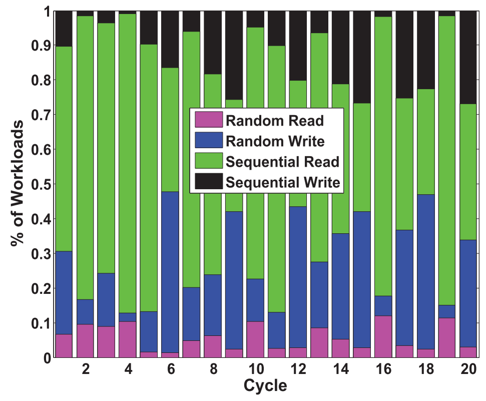
Graph Clustering. Graph as an expressive data structure is popularly used to model structural relationship between objects in many application domains, ranging from web, social networks, biological networks to sensor networks [22]–[26]. Graph clustering has attracted active research in the last decade [21], [24], [27]–[40]. Most of existing graph clustering techniques have focused on the topological structure based on various criteria, including normalized cuts [24], modularity [27], structural density [28], stochastic flows [31] or clique [33]. The clustering results often contain densely connected components within clusters. However, such methods usually ignore vertex attributes in the clustering process. On the other hand, K-SNAP [30] presented OLAP-style aggregation approach to summarize large graphs by grouping nodes based on the user-selected attributes. Kenley and Cho [35] exploited an information-theoretic model for clustering by growing a random seed in a manner that minimizes graph entropy. This kind of methods achieve homogeneous attribute values within clusters, but ignore the intra-cluster topological structure. Recently, Shiga *et al*. [29] presented a clustering method which integrates numerical vectors with modularity into a spectral relaxation problem. SA-Cluster [32] and Inc-Cluster [34] perform clustering based on both structural and attribute similarities by incorporating attributes as augmented edges to its vertices, transforming attribute similarity to vertex closeness. BAGC [37] constructs a Bayesian probabilistic model to capture both structural and attribute aspects. GenClus [41] proposed a model-based method for clustering heterogeneous networks with different link types and different attribute types. SI-Cluster [38] performs social influence based clustering over heterogeneous networks by dynamically combining self-influence from social graph and multiple types of co-influences from activity graphs. VEPathCluster [40] is the first clustering algorithm to tightly integrate vertex-centric clustering and edge-centric clustering by mutually enhancing each other with combining different types of meta paths over heterogeneous information network. However, to the best of our knowledge, GraphLens is the first one that applies and extends graph clustering 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.

Scalable Graph Processing. With continued advances in computing and information technology, big graphs have grown at an astonishing rate in terms of volume, variety, and velocity. Efficient iterative computation on such huge graphs is widely recognized as a challenging big data research problem, which has received heated attention recently. We can broadly classify existing research activities on scaling iterative graph computations into two categories: (1) Distributed solutions and (2) Single PC based solutions. Most of existing research efforts are dedicated to the distributed graph partitioning strategies that can effectively break large graphs into small, relatively independent parts [42]–[47]. Several recent efforts [48]–[52] have successfully demonstrated huge opportunities for optimizing graph processing on a single PC through graph parallel abstractions that are efficient in both storage organization and in-memory computation. However, most existing approaches rely on vertex-centric graph parallel computation model. Many existing algorithms fail to work effectively under the vertex-centric computation model for several scenarios: (1) when the algorithms require to load the whole graph into the main memory but the graph and its intermediate results of computation together are too big to fit into the available memory; (2) when high degree vertices and their edges combined with the necessary intermediate results are too big to fit into the working memory; (3) when the time of computing on a vertex and its edges is much faster than the time to access to the vertex state and its edge data in memory or on disk; and (4) when the computation workloads on different vertices are significantly imbalanced due to the highly skewed vertex degree distribution.

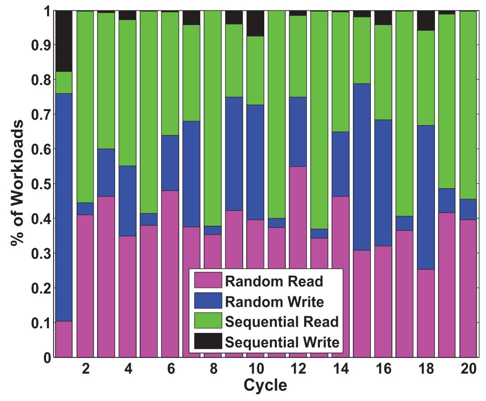
Enterprise Storage Traces. We analyze block-level traces collected from three large enterprise storage installations: a live banking environment, a retail backend system environment and an email server environment. Each of the traces consists of storage workloads collected over every 15 minute period (referred to as "cycle") for storage addresses, called "extents", each extent representing 1 GB logical address unit. The traces



(a) Bank Trace



(b) Email Trace



(c) Store Trace

Fig. 1. I/O Workloads by Four Access Patterns.

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

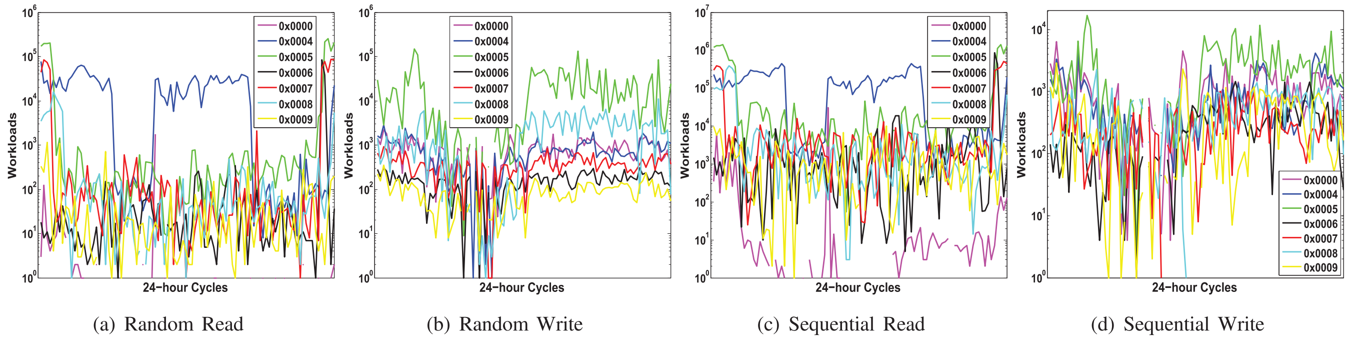


Fig. 2. 24-hour I/O Workloads by Each Lun on Email Trace.

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. From these three figures, we observe that real trace datasets in different areas often have diverse access patterns. 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.

Figures 2(a), (b), (c) and (d) present the activity distribution of seven Luns on each of four access patterns observed on Email Trace. Each figure summarizes some access activities of seven Luns within one-day storage log. As shown in Figures 2(a) and (c), there are a lot of read accesses ("random read" and "sequential read") on Lun "0x0004" and other read activities are evenly distributed to other Luns except "0x0000", which has relatively low workloads on "sequential read". On the other hand, from Figures 2(b) and (d), there are heavy write workloads ("sequential read" and "sequential read") on Lun "0x0005". Other Luns have less write activities, especially Luns "0x0006" and "0x0009". To sum up, each Lun has different workloads on diverse access patterns and the distribution of access activities for each access patterns are quite different. Thus, a unified hotspot identification method may not make sense for multiple access patterns.

The traditional clustering algorithms, such as K-Means [53] or K-Medoids [54], are usually non-graph clustering methods. They partition the extents into clusters in terms of only their direct correlations (self-similarity). However, partitioning the extents without considering their indirect relationships based on the view of graph analytics may lead to lots of cluster outliers and inaccurate extent clustering. Our proposed unified neighborhood random walk model provides a natural way to capture both direct and indirect access correlations between extents based on the heterogeneous graph representation of trace data.

As discussed above, there are usually multiple access patterns in the context of enterprise storage systems. Most of the conventional clustering methods only address each individual access pattern separately and then simply combine multiple access patterns into a unified framework with the equal weighting factors. We argue that such static combination method in the conventional approaches performs poorly when different access patterns are correlated rather than completely independent. We propose an iterative dynamic weight learning algorithm to find the optimal weight assignment scheme for multiple access patterns to enable us to better characterize important hotspots of storage access and understand hotspot movement 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, temporal 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 and temporal patterns interact with one another? what types of spatial/temporal access patterns are common in real world traces? and how hot spots move across extents (spatial) and across cycles (temporal)?

One approach to 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. As 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.

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

TABLE I
A SAMPLE TRACE FOR A CYCLE

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

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 (extent) 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 3(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 3(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.

Case 1: summarization of all paths between any pair of extents. In Figure 4(a), there exists four 2-hop paths between extents "0x0021, 0" and "0x0021, 1" through four attribute vertices respectively. In comparison to Figure 4(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 5(a) and Figure 5(b), two extents are reachable by two 2-hop paths through RR and RW respectively. However, the two addresses in Figure 5(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).

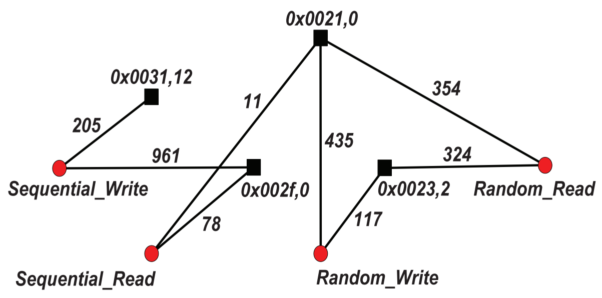
Case 3: attribute differentiation by attribute weight significance. In both Figure 6(a) and Figure 6(b), two extents are reachable by two 2-hop paths through RR and RW respectively and the corresponding attribute edge weights are the same respectively. However, the corresponding attribute edge weights (100 for RR and 200 for RW) in Figure 6(a) are larger than that (10 for RR and 20 for RW) in Figure 6(b). Thus, two extents in Figure 6(a) are more similar than two extents in Figure 6(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 we calculate the similarity scores between two extents. The only difference between Figures 7(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"). Similarly, we can generate other k-hop paths between extent "0x0021, 3" and extent "0x0021, 4" by executing a random walk process with starting at "0x0021, 3", walking through different intermediate extents and arriving at "0x0021, 4". We argue that extents "0x0021, 3" and "0x0021, 4" in Figure 7(a) have larger attribute edge weights and thus are more similar than extents "0x0021, 9" and "0x0021, 10" in Figure 7(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).

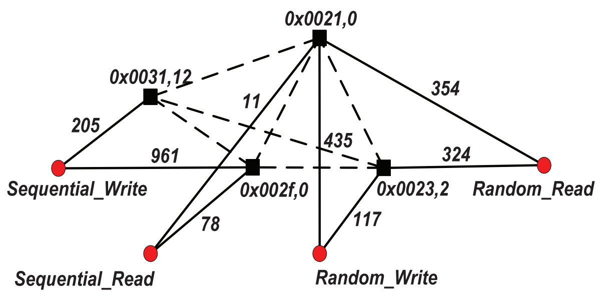
B. *Trace Analysis With GraphLens*

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

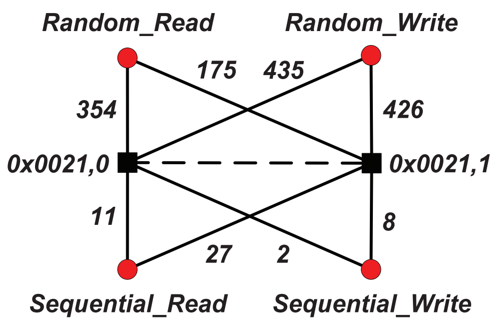


(a) Graph Model of a Single Cycle

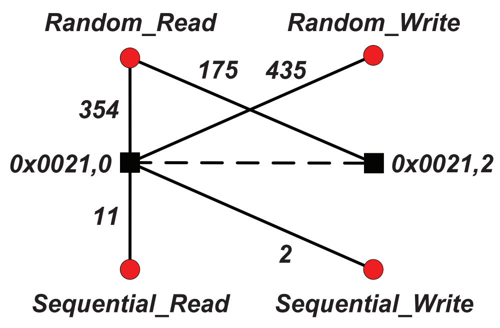


(b) Structure Relationship Deriving

Fig. 3. An Illustrating Example of Heterogeneous Trace Graph.

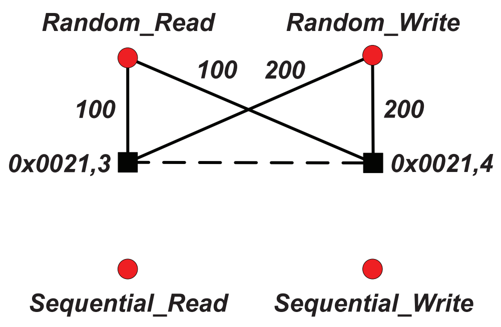


(a) More Paths

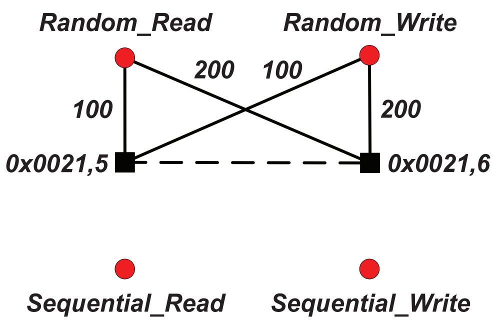


(b) Less Paths

Fig. 4. Summarization of all Possible Paths.

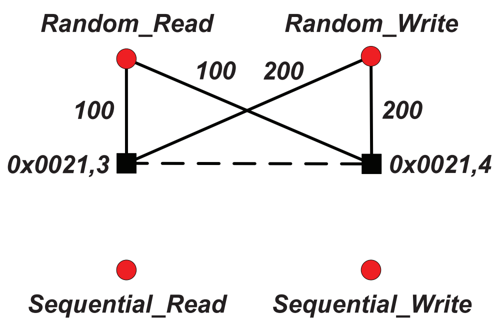


(a) Weight Match

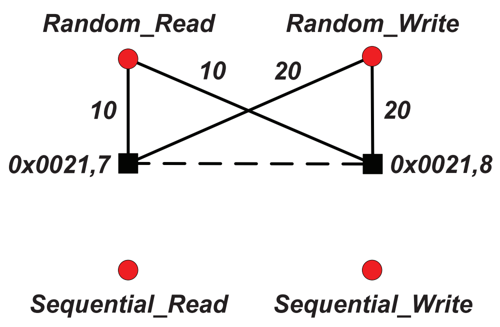


(b) Weight Mismatch

Fig. 5. Attribute Weight Match.



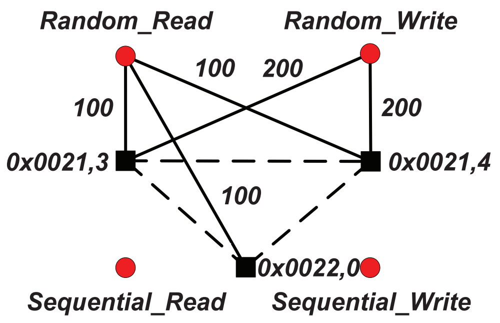
(a) Significant Weights



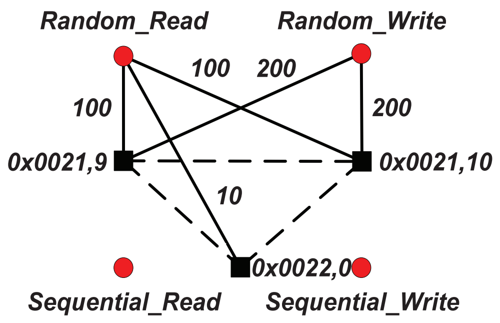
(b) Insignificant Weights

Fig. 6. Attribute Weight Significance.

between extents (structure vertices). 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



(a) Significant 4-hop Path



(b) Insignificant 4-hop Path

Fig. 7. k-hop Path.

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 $\forall p, q, 1 \leq p, q \leq k_i, p \neq q$, 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.

In Phase II, pairwise cycle similarity is computed in terms of similarities between the corresponding extents in two cycles to capture the temporal access similarity between cycles from historical storage workloads and predict future workload tendency. Similarly, we define a **unified weighted cycle similarity measure** to measure the closeness between cycles based on connectivity, vicinity and random walk similarities between each extent and their neighbors in the respective cycles.

The problem of **temporal cycle clustering** is to partition the objective cycles $T = \{t_i | i = 1, 2, \ldots, N\}$ into k disjoint clusters $C_1, C_2, \ldots, C_k$, where N is the number of cycles, $T = \bigcup_{p=1}^{k} C_p$ and $C_p \bigcap C_q = \phi$ for $\forall p, q, 1 \leq p, q \leq k, p \neq q$, to ensure: (1) the cycles within each cluster have larger similarity scores, while the cycles in different clusters have smaller similarity scores; and (2) the corresponding extents and their neighbors within cycle clusters have similar access patterns on all attributes, while the corresponding extents and their neighbors in different cycle clusters may have diverse access patterns.

The spatial based clustering combined with the temporal based clustering can indicate where the hotspots are and how such hotspots move across extents (along spatial dimension) and/or across cycles (along temporal dimension).

IV. Methodology

In this section we describe the two-phase correlation analysis methodology in GraphLens: extent-similarity based spatial clustering and cycle-similarity based temporal 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 3, 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.

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

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 $P(i)^2$, $P(i)^3$, …, 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.

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

where e_{pq} represents the count of access pattern a_q that storage extent v_p has in the given cycle t_i. For instance, a storage extent of "0x0021, 0" has the count of 354 on "random read" in the example cycle of Figure 3(a). α_{iq} denotes the weight of attribute edges from any of the structure vertices to attribute vertex a_q in the heterogeneous graph G_i. Since each row of transition probability matrix should sum to 1, we employ the row-wise normalization for P_{SA}.

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

where e_{pq} specifies the count of an access pattern a_p achieved by a storage extent v_q in t_i. α_{ip} denotes the weight of attribute edges from attribute vertex a_p to structure vertices in G_i. 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.

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

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. A Unified Temporal Similarity Measure

The access similarity between cycles in a workload trace can help us identify hotspot periodicity and hotspot movement across different extents and different cycles and discover deeper correlations among different cycles to predict future trends of workloads. Intuitively, two cycles are considered more (or less) similar if (1) more (or less) corresponding extents in two cycles share the same access patterns; and if (2) more (or less) neighbors of the corresponding extents with the same access patterns have the same access patterns.

Figure 8 presents an illustrative example of heterogeneous trace graphs for three cycles. Cycle 1 in Figure 8(a) is more similar to Cycle 2 in Figure 8(b) than to Cycle 3 in Figure 8(c) for two reasons. This is because the corresponding extents in Cycle 1 and Cycle 2 are more similar in terms of their extent similarity and also have large weights, compared to Cycle 1 and Cycle 3.

Similar to the dynamic weight tuning method to generate an optimal weight assignment for various types of attribute links when we calculate the unified spatial random walk similarity for each cycle, we also need to provide a unified cycle based temporal similarity measure with diverse and tunable optimal weight assignments. Thus, we split the original heterogeneous trace graph for each cycle into m_i subgraphs where m_i is the number of attribute vertices in cycle t_i. For example, we first split the heterogeneous trace graph in Figure 3(a) into four subgraphs based on RR, RW, SR and SW respectively. For each subgraph, we only keep the related attribute edges. We then calculate the individual spatial random walk similarity $s_{ij}(u,v)$ or $R_j(i)$ for the subgraph based on attribute a_j in cycle t_i by using the same formulae of Eqs. (1) and (5).

Definition 3 (Inter-cycle Extent Similarity): Let V be the set of n extents, A be the set of m associated attributes, the inter-cycle extent similarity, denoted by $s_{ijh}(v_p)$, based on attribute $a_h \in A$ between extent $v_p \in V$ in cycle t_i and extent $v_p \in V$ in cycle t_j is defined as follow.

$$s_{ijh}(v_p) = 1 - \sqrt{\frac{\sum_{q=1}^{n}(s_{ih}(v_p,v_q)a_{ih}(v_q) - s_{jh}(v_p,v_q)a_{jh}(v_q))^2}{\sum_{q=1}^{n}(s_{ih}(v_p,v_q)a_{ih}(v_q) + s_{jh}(v_p,v_q)a_{jh}(v_q))^2}} \tag{6}$$

where $s_{xh}(v_p,v_q)$ represents the individual spatial random walk similarity between extents v_p and v_q for the subgraph based on attribute a_h in cycle t_x. $a_{xh}(v_q)$ specifies the count achieved by extent v_q on attribute a_h in cycle t_x.

If v_p has the similar similarity scores with its neighbors in both cycles and the corresponding neighbors in both cycles have the similar access count on attribute a_h, then the inter-cycle extent similarity $s_{ijh}(v_p)$ should be larger.

Definition 4 (Unified Cycle Similarity): Let V be the set of n extents, A be the set of m associated attributes, the unified cycle similarity, denoted by $s(t_i,t_j)$, between cycle $t_i \in T$ and cycle $t_j \in T$ is defined as follow.

$$\begin{aligned} s_h(t_i,t_j) &= \frac{1}{n}\sum_{p=1}^{n} s_{ijh}(v_p) \\ s(t_i,t_j) &= \sum_{h=1}^{m} \beta_h s_h(t_i,t_j) \end{aligned} \tag{7}$$

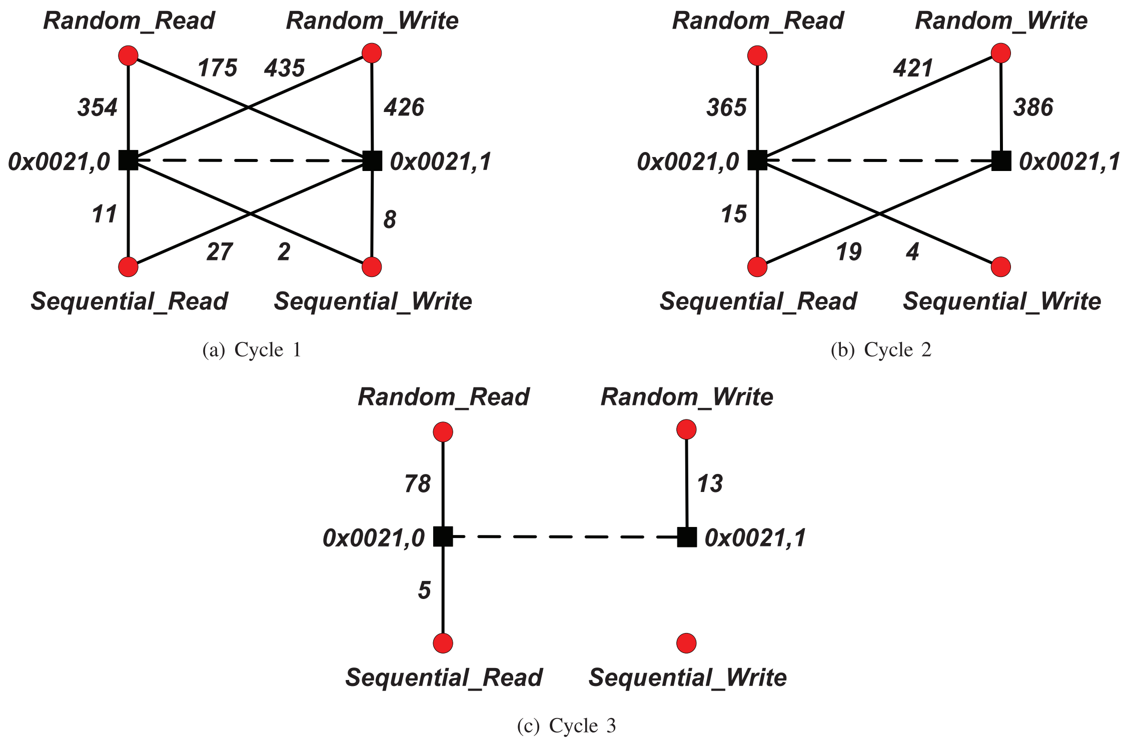


Fig. 8. Cycle Differentiation.

where $s_h(t_i, t_j)$ represents the individual temporal similarity between cycles t_i and t_j based on attribute a_h. β_h is a weighting factor for the individual temporal similarity based on attribute a_h. The unified temporal similarity $s(t_i, t_j)$ sums over the dependency of all access patterns between two cycles.

We argue that each type of access patterns may have different degrees of importance in cycle similarity. Thus we assign an individual weight for each kind of access patterns. All weights are initialized to $1/m$, but will be automatically updated during clustering according to their contributions to clustering convergence.

C. Spatial Extent Clustering

Our spatial extent clustering framework, SE-CLUSTER, partitions extents in a heterogeneous trace graph G_i into k_i densely connected clusters. SE-CLUSTER follows the traditional K-Medoids clustering method [54] by using the unified random walk similarity $R(i)$ with k_i dense extents as the initial centroids and the initial weights $\alpha_{i1}^0, \ldots, \alpha_{im}^0$ 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 is reduced to three subproblems: (1) cluster assignment, (2) centroid update and (3) weight adjustment, each with the goal of maximizing the objective function.

1) Initialization: We will address two main issues in the initialization step: (1) initial weight setup and (2) cluster centroid initialization.

Good weight assignment for our unified extent neighborhood random walk similarity measure is crucial to produce a good clustering result. We first assign the initial value of 1.0 to all m weights, i.e., $\alpha_{i1} = \ldots = \alpha_{im} = 1.0$, based on the assumption that each kind of attribute edges has the same degree of importance. By assigning an equal initial weight, we start to combine the structure edges and the m kinds of attribute edges into a heterogeneous trace graph with the unified transition matrix (Recall Eqs. (1)–(3) in Section IV-A). We will update the m weight values in the subsequent iteration of the clustering process using our dynamic weight update scheme, which continuously quantify and adjust the weights on m kinds of attribute edges towards the clustering convergence, while maintaining the constraint $\sum_{j=1}^{m} \alpha_{ij} = m$. As a result, at each iteration, weights of important attribute edges are increased while weights of trivial attribute edges are decreased or become zero, ensuring the clustering process progresses towards the convergence. Note that choosing a weight assignment randomly often results in incorrect clustering results. In Section IV-C5, we will show that there exists one and only one optimal weight assignment to maximize the clustering objective. Good initial centroids are essential for the success of partitioning clustering algorithms. SE-CLUSTER produces good cluster characteristics from noisy data by choosing the local maxima of the density function as centroids. Concretely, we first compute the density for each extent vertex in the trace graph and then find the local maxima of the density function as the centroids.

Definition 5: [Extent Density Function] The density function of one extent vertex v_p is the sum of the unified similarity scores between v_p and all other extent vertices in V.

$$ED(v_p) = \sum_{v_q \in V, v_q \neq v_p} s_i(v_p, v_q) \tag{8}$$

We know that the unified random walk similarity captures both direct and indirect relationships between extents in the trace graph. If one extent vertex v_p has a large density value, it means that, either v_p connects to many extent vertices through the structure edges within the trace graph, or v_p has the similar access patterns with many extent vertices through the attribute links. Based on the density value of each extent vertices, we find the extent vertices with a local maximum of the density value by following the hill-climbing strategy in *DENCLUE* [55]. An extent vertex which has a local maximum of the density value often can diffuse its influence to many extent vertices along multiple paths. A centroid-based cluster is thus formed when influence is diffused to the margin of the trace graph. We sort all such extent vertices in the descending order of their density values and select top-k_i extent vertices as the initial k_i centroids $\{c_1^0, \ldots, c_{k_i}^0\}$.

2) Vertex Assignment and Centroid Update: With k_i centroids in the t^{th} iteration, we assign each extent vertex $v_p \in V$ to its closest centroid $c^* = argmax_{c_q^t} s_i(v_p, c_q^t)$, *i.e.*, a centroid $c^* \in \{c_1^t, \ldots, c_{k_i}^t\}$ with the largest unified similarity from v_p. When all vertices are assigned to some cluster, the centroid will be updated with the most centrally located vertex in each cluster. To find such a vertex, we first compute the "average point" $\overline{v_p}$ of a cluster C_p in terms of the unified random walk similarity matrix as

$$s_i(\overline{v_p}, v_q) = \frac{1}{|C_p|} \sum_{v_r \in C_p} s_i(v_r, v_q), \forall v_q \in C_p \tag{9}$$

Thus, $s_i(\overline{v_p}, :)$ is the average unified similarity vector for extent cluster C_p in the trace graph for cycle t_i. Then we find the new centroid c_p^{t+1} in cluster C_p as

$$c_p^{t+1} = argmin_{v_q \in C_p} \|s_i(v_q, :) - s_i(\overline{v_p}, :)\| \tag{10}$$

Therefore, we find the new centroid c_p^{t+1} in the $(t+1)$th iteration whose unified similarity vector is the closest to the cluster average.

3) Clustering Objective Function: The objective of clustering is to maximize intra-cluster similarity and minimize inter-cluster similarity. We design our clustering objective function according to this general goal. As our similarity measure is the unified random walk similarity, we will maximize the intra-cluster unified random walk similarity and minimize the inter-cluster unified random walk similarity.

Definition 6 (Inter-cluster Similarity): Let G_i be a heterogeneous trace graph, $s_i(v_p, v_q)$ denote the unified random walk similarity between extents v_p and v_q, the inter-cluster similarity between clusters C_x and C_y, denoted by $s_i(C_x, C_y)$, is defined as follow.

$$s_i(C_x, C_y) = \frac{\sum_{v_p \in C_x, v_q \in C_y} s_i(v_p, v_q)}{|C_x| \times |C_y|} \tag{11}$$

where $|C_x|$ or $|C_y|$ represents the cardinality of vertex set C_x or C_y. This inter-cluster similarity measure is designed to quantitatively measure the extent of similarity between two clusters of V.

Definition 7: [Extent Clustering Objective Function] Let G_i be a heterogeneous trace graph with the weight factors $\alpha_{i1}, \ldots, \alpha_{im}$ for m kinds of attribute edges respectively, and k_i be a number of extent clusters. The goal of SE-CLUSTER is to find k_i partitions $\{C_x\}_{x=1}^{k_i}$ such that $V = \bigcup_{x=1}^{k_i} C_x$ and $C_x \bigcap C_y = \phi$ for $\forall x, y, 1 \le x, y \le k_i, x \neq y$, and the following objective function $O(\{C_x\}_{x=1}^{k_i}, \alpha_{i1}, \ldots, \alpha_{im})$ is maximized.

$$O(\{C_x\}_{x=1}^{k_i}, \alpha_{i1}, \ldots, \alpha_{im}) = \frac{\sum_{p=q=1}^{k_i} s_i(C_p, C_q)}{\sum_{p=1}^{k_i} \sum_{q=1, q \neq p}^{k_i} s_i(C_p, C_q)} \tag{12}$$

subject to $\sum_{l=1}^{m} \alpha_{il} = m, \alpha_{il} \ge 0, l = 1, \ldots, m$.

According to Eqs. (1)–(5), the objective function O is a fractional function of multi variables $\alpha_{i1}, \ldots, \alpha_{im}$ with non-negative real coefficients. On the other hand, the numerator and the denominator of O are both polynomial functions of the above variables. Without loss of generality, we rewrite Eq. (12) as follow.

$$\begin{aligned} &\max_{\alpha_{i1}, \ldots, \alpha_{im}} O(\{C_x\}_{x=1}^{k_i}, \alpha_{i1}, \ldots, \alpha_{im}) \\ &= \max_{\alpha_{i1}, \ldots, \alpha_{im}} \frac{\sum_{j=1}^{p} a_j \prod_{l=1}^{m} (\alpha_{il})^{b_{jl}}}{\sum_{j=1}^{q} o_j \prod_{l=1}^{m} (\alpha_{il})^{r_{jl}}} \\ &\quad a_j, b_{jl}, o_j, r_{jl} \ge 0, b_{jl}, r_{jl} \in \mathbb{Z}, \\ &s.t. \sum_{l=1}^{m} \alpha_{il} = m,\ \alpha_{i1}, \ldots, \alpha_{im} \ge 0 \end{aligned} \tag{13}$$

where there are p polynomial terms in the numerator and q polynomial terms in the denominator, a_j and o_j are the coefficients of the j^{th} terms respectively, and b_{jl} and r_{jl} are the exponents of corresponding variables in the j^{th} terms respectively.

For ease of presentation, we revise the original objective as the following nonlinear fractional programming problem (NFPP).

Definition 8: [Nonlinear Fractional Programming Problem] Let $f(\alpha_{i1}, \ldots, \alpha_{im}) = \sum_{j=1}^{p} a_j \prod_{l=1}^{m} (\alpha_{il})^{b_{jl}}$ and $g(\alpha_{i1}, \ldots, \alpha_{im}) = \sum_{j=1}^{q} o_j \prod_{l=1}^{m} (\alpha_{il})^{r_{jl}}$, the clustering goal is revised as follow.

$$\max_{\alpha_{i1}, \ldots, \alpha_{im}} \frac{f(\alpha_{i1}, \ldots, \alpha_{im})}{g(\alpha_{i1}, \ldots, \alpha_{im})},\ s.t. \sum_{l=1}^{m} \alpha_{il} = m,\ \alpha_{i1}, \ldots, \alpha_{im} \ge 0 \tag{14}$$

Our clustering objective is equivalent to maximize a quotient of two polynomial functions of multiple variables. It is very hard to perform function trend identification and estimation to determine the existence and uniqueness of solutions.

4) Unique Optimal Weight Assignment: This section provides the theoretical analysis to demonstrate that there exists a unique optimal assignment for the weights $\alpha_{i1}, \ldots, \alpha_{im}$ to maximize the clustering objective function in Eq. (12).

Definition 9: [Nonlinear Parametric Programming Problem] Let $f(\alpha_{i1},\dots,\alpha_{im}) = \sum_{j=1}^{p} a_j \prod_{l=1}^{m} (\alpha_{il})^{b_{jl}}$ and $g(\alpha_{i1},\dots,\alpha_{im}) = \sum_{j=1}^{q} o_j \prod_{l=1}^{m} (\alpha_{il})^{r_{jl}}$, the NPPP is defined as follow.

$$\digamma(\gamma) = \max_{\alpha_{i1},\dots,\alpha_{im}} f(\alpha_{i1},\dots,\alpha_{im}) - \gamma g(\alpha_{i1},\dots,\alpha_{im}),$$
$$s.t. \sum_{l=1}^{m} \alpha_{il} = m,\ \alpha_{i1},\dots,\alpha_{im} \ge 0 \tag{15}$$

Theorem 1: The NFPP in Definition 8 is equivalent to the NPPP in Definition 9, i.e., $\gamma = \max_{\alpha_{i1},\dots,\alpha_{im}} \frac{f(\alpha_{i1},\dots,\alpha_{im})}{g(\alpha_{i1},\dots,\alpha_{im})}$ if and only if $\digamma(\gamma) = \max_{\alpha_{i1},\dots,\alpha_{im}} f(\alpha_{i1},\dots,\alpha_{im}) - \gamma g(\alpha_{i1},\dots,\alpha_{im}) = 0$. See Proof in the Appendix.

Now the original NFPP has been successfully transformed into the straightforward NPPP. This transformation can help us conveniently identify the existence and uniqueness of solutions.

Theorem 2: $\digamma(\gamma)$ is convex. See Proof in the Appendix.

Theorem 3: $\digamma(\gamma)$ is monotonically decreasing. See Proof in the Appendix.

Theorem 4: $\digamma(\gamma) = 0$ has a unique solution. See Proof in the Appendix.

5) Vote-Based Weight Self-Adjustment: We propose a dynamic weight adjustment method to iteratively improve the spatial extent clustering objective. Let $\alpha_{il}^{t} (l = 1,\dots,m)$ be the weights for m kinds of attribute edges between structure (extent) vertices and attribute vertex a_l in the transition probability $P(i)$ of G_i in the t^{th} iteration. All α_{il}^{0}s are first initialized as 1.0. We then iteratively adjust α_{il}^{t+1} with an increment $\triangle\alpha_{il}^{t}$, which denotes the weight update of attribute edges between structure vertices and attribute vertex a_l in $P(i)$. The attribute weight α_{il}^{t+1} in the $(t+1)$th iteration is computed as

$$\alpha_{il}^{t+1} = \frac{1}{2}(\alpha_{il}^{t} + \triangle\alpha_{il}^{t}) \tag{16}$$

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

$$vote_{il}(u,v) = \begin{cases} 1, & 1 - \dfrac{|a_l(u) - a_l(v)|}{|a_l(u) + a_l(v)|} > \epsilon \\ 0, & otherwise \end{cases} \tag{17}$$

where $a_l(x)$ specifies the count achieved by x on a_l in G_i. A positive number ϵ denotes a threshold of similar extent of u and v on a_l. Then the votes $vote_{il}^{t}$ of α_{il}^{t} for the entire trace graph G_i is estimated by counting the number of structure vertices within clusters which share similar access counts with the centroids c_j in cluster C_j on attribute a_l.

$$vote_{il}^{t} = \sum_{j=1}^{k_i} \sum_{v \in C_j} vote_{il}(c_j, v) \tag{18}$$

The weight increment $\triangle\alpha_{il}^{t}$ is then calculated as

$$\triangle\alpha_{il}^{t} = \frac{vote_{il}^{t}}{\frac{1}{m}\sum_{p=1}^{m} vote_{ip}^{t}} = \frac{\sum_{j=1}^{k_i} \sum_{v \in C_j} vote_{il}(c_j, v)}{\frac{1}{m}\sum_{p=1}^{m} \sum_{j=1}^{k_i} \sum_{v \in C_j} vote_{ip}(c_j, v)} \tag{19}$$

The larger number of extent vertices which have similar access counts on attribute a_l, the larger $\triangle\alpha_{il}^{t}$ is. Then the adjusted weight is calculated as

$$\alpha_{il}^{t+1} = \frac{1}{2}(\alpha_{il}^{t} + \triangle\alpha_{il}^{t}) = \frac{1}{2}\left(\alpha_{il}^{t} + \frac{vote_{il}^{t}}{\frac{1}{m}\sum_{p=1}^{m} vote_{ip}^{t}}\right) \tag{20}$$

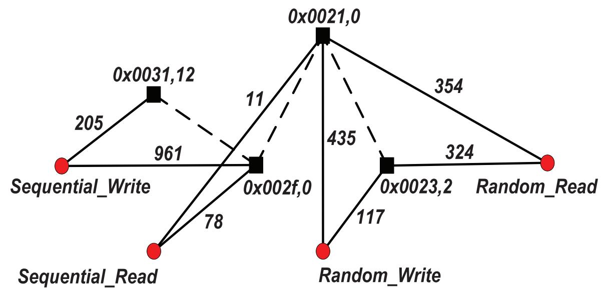
The denominator in Eq. (19) ensures that the constraint $\sum_{l=1}^{m} \alpha_{il} = m$ in Definition 7 is still satisfied after weight adjustment. The adjusted weights may be increasing, decreasing, or unchanged depending on the value of $\triangle\alpha_{il}^{t}$. If $\triangle\alpha_{il}^{t} > \alpha_{il}^{t}$, then $\alpha_{il}^{t+1} > \alpha_{il}^{t}$, *i.e.*, the attribute edges between $\forall v \in V$ and a_l make an increasing contribution to the unified random walk similarity. Similarly, if $\triangle\alpha_{il}^{t} < \alpha_{il}^{t}$, then $\alpha_{il}^{t+1} < \alpha_{il}^{t}$. If $\triangle\alpha_{il}^{t} = \alpha_{il}^{t}$, then $\alpha_{il}^{t+1} = \alpha_{il}^{t}$.

An important property of the weight self-adjustment mechanism is that the updated weights are adjusted towards the direction of increasing the clustering objective function value. We briefly illustrate this property qualitatively: if a large number of structure (extent) vertices within clusters have the similar access counts on a_l, then the weight is increased, i.e., $\alpha_{il}^{t+1} > \alpha_{il}^{t}$; on the other hand, if extent vertices within clusters have quite different access counts on a_l, the weight is then decreased, i.e., $\alpha_{il}^{t+1} < \alpha_{il}^{t}$. There must be some weights with increasing updates and some other weights with decreasing updates, since $\sum_{l=1}^{m} \alpha_{il} = m$ 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 extent vertices tend to be clustered into the same cluster, thus increasing the objective function towards convergence.

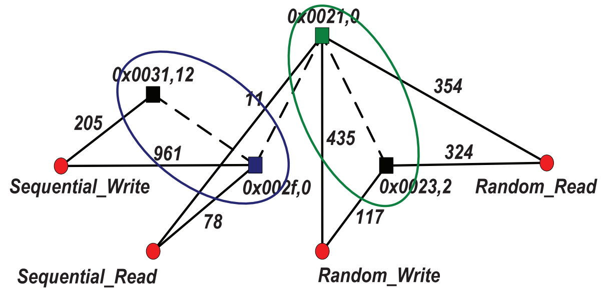
By assembling different parts, our spatial extent clustering algorithm SE-Cluster is presented in Algorithm 1.

6) An Illustrating Example of Spatial Extent Clustering: We use Table I, Figure 3 and Figure 9 as an example to illustrate each component of our proposed spatial extent clustering algorithm. Suppose that we are given the original trace dataset in Table I, we generate the heterogeneous graph representation of the trace dataset, as shown in Figure 3(a) and we only consider 2-hop paths between extents to derive the structure relationships between extent vertices, as shown in Figure 9(a).

First, we compute the unified spatial similarity measure in terms of Eq. (5) based on 2-hop random walk paths between extents. In the context of this simple example, we will



(a) 2-hop Path



(b) Spatial Extent Clustering

Fig. 9. An Illustrating Example of Spatial Extent Clustering.

Algorithm 1. Spatial Extent Clustering

```
Input: a heterogeneous trace graph $G_i = (V, A, E_i, F_i)$ for cycle $t_i$, a length limit $l$ of random walk paths, a restart probability $c$, a cluster number $k_i$, a threshold $\epsilon$ of similar extent, the initial weights $\alpha_{ip}^0 = 1.0$.
Output: $k_i$ extent clusters $C_1, \ldots, C_{k_i}$.
1: Calculate $R(i)$ according to Eqs. (1)–(5);
2: Compute $ED(v_p)$ for each extent vertex $v_p \in V$ in terms of Eq. (8);
3: Select $k_i$ initial centroids $c_1^0, \ldots, c_{k_i}^0$;
4: Repeat until $O(\{C_r\}_{r=1}^{k_i}, \alpha_{i1}, \ldots, \alpha_{im})$ converges:
5:   Assign each extent vertex $v_p \in V$ to a cluster $C^*$ with a centroid $c^*$ where $c^* = argmax_{c_q^t} s_i(v_p, c_q^t)$;
6:   Update $k_i$ centroids $c_1^t, \ldots, c_{k_i}^t$ in terms of Eqs. (9)–(10);
7:   Update each weight $\alpha_{il}^t$ according to Eqs. (17)–(20);
8:   Re-calculate $R(i)$ with the updated $\alpha_{il}^t$;
9: Return $C_1, \ldots, C_{k_i}$.
```

have three positive spatial similarity scores: a similarity score between extent vertices "0x0021, 0" and "0x0023, 2" since they share two access patterns (RR and RW), a similarity score between extent vertices "0x0021, 0" and "0x002f, 0" since they share one access pattern (SR), and a similarity score between extent vertices "0x002f, 0" and "0x0031, 12" since they share one access pattern (SW).

Second, if we want to partition four extent vertices in Figure 9(a) into 2 clusters, then we need to compute the extent density value for each of four extent vertices in terms of Eq. (8). Notice that two extent vertices "0x0021, 0" and "0x002f, 0" have two structure edges (dashed lines) and two extent vertices "0x0023, 2" and "0x0031, 12" have only one structure edge respectively. Thus, "0x0021, 0" and "0x002f, 0" have larger density values and then we choose these two denser extent vertices as cluster centroids.

Next, we assign other two extent vertices to their closest centroids. In Figure 9(a), "0x0023, 2" has a structure edge to connect to centroid "0x0021, 0" since they have a positive spatial similarity score and there does not exist any structure edge between "0x0023, 2" and centroid "0x002f, 0". Thus, "0x0023, 2" is closer to centroid "0x0021, 0" and it will be assigned to the cluster where "0x0021, 0" belongs to. Similarly, "0x0031, 12" is closer to centroid "0x002f, 0" and it will be assigned to the cluster where "0x002f, 0" belongs to, since they have a positive spatial similarity score.

D. Temporal Cycle Clustering

Our temporal cycle clustering framework, TC-CLUSTER, partitions N cycles in the workload trace into k clusters. TC-CLUSTER follows the same clustering framework as SE-CLUSTER by using the unified similarity $s(t_i, t_j)$ with k dense cycles as initial centroids and $\beta_1^0, \ldots, \beta_m^0$ as initial weights. At each iteration, based on unified cycle similarity scores, we select the most centrally located cycles in a cluster as a centroid, and assign the rest of cycles to their closest centroids. The objective of clustering is to maximize intra-cluster cycle similarity and minimize inter-cluster cycle similarity. Unlike SE-CLUSTER, TC-CLUSTER utilizes the entropy measure to determine the degree of importance of attributes when we balance the individual temporal similarity based on each attribute. The weight update method computes the entropy of each kind of attributes, and updates m weights accordingly after each iteration. This process is repeated until convergence.

1) Initialization: We choose the weights $\beta_1 = \ldots = \beta_m = 1/m$ as an initial input. Similarly, we utilize the hill-climbing strategy to select k cycles, which have the local maxima of the following density function, as the initial centroids $\{c_1^0, \ldots, c_k^0\}$.

Definition 10: [Cycle Density Function] The density function of one cycle t_i is the sum of the unified similarity scores between t_i and all other cycles in the objective cycles T.

$$CD(t_i) = \sum_{t_j \in T, t_j \neq t_i} s(t_i, t_j) \tag{21}$$

2) Vertex Assignment and Centroid Update: At the t^{th} iteration, we assign each cycle $t_i \in T$ to its closest centroid $c^* = argmax_{c_j^t} s(t_i, c_j^t)$. The "average point" $\overline{t_i}$ of a cluster C_i is calculated as

$$s(\overline{t_i}, t_j) = \frac{1}{|C_i|} \sum_{t_l \in C_i} s(t_l, t_j), \forall t_j \in C_i \tag{22}$$

where $s(\overline{t_i}, :)$ is the average unified similarity vector for cycle cluster C_i. The new centroid c_i^{t+1} in cluster C_i is generated as

$$c_i^{t+1} = argmin_{t_j \in C_i} \|s(t_j, :) - s(\overline{t_i}, :)\| \tag{23}$$

3) Clustering Objective Function: The objective of clustering is to maximize intra-cluster unified similarity and minimize inter-cluster unified similarity.

Definition 11 [Inter-cluster Similarity]: Let T be the set of N cycles, $s(t_i, t_j)$ denote the unified cycle similarity between cycles t_i and t_j, the inter-cluster similarity between clusters C_x and C_y, denoted by $s(C_x, C_y)$, is defined as follow.

$$s(C_x, C_y) = \frac{\sum_{t_i \in C_x, t_j \in C_y} s(t_i, t_j)}{|C_x| \times |C_y|} \tag{24}$$

Definition 12 [Cycle Clustering Objective Function]: Let T be the set of N cycles with the weight factors $\beta_1, \ldots, \beta_m$ for m attributes respectively, and k be a number of cycle clusters. The goal of TC-CLUSTER is to find k partitions $\{C_x\}_{x=1}^k$ such that $T = \bigcup_{x=1}^k C_x$ and $C_x \bigcap C_y = \phi$ for $\forall x, y, 1 \le x, y \le k, x \ne y$, and the following objective function $O(\{C_x\}_{x=1}^k, \beta_1, \ldots, \beta_m)$ is maximized.

$$O(\{C_x\}_{x=1}^k, \beta_1, \ldots, \beta_m) = \frac{\sum_{p=q=1}^k s(C_p, C_q)}{\sum_{p=1}^k \sum_{q=1, q \ne p}^k s(C_p, C_q)} \tag{25}$$

subject to $\sum_{l=1}^m \beta_l = 1, \beta_l \ge 0, l = 1, \ldots, m$.

According to Eq. (7), we know that the numerator and the denominator of the objective function O are both linear functions of multi variables $\beta_1, \ldots, \beta_m$ with non-negative real coefficients. Thus, our clustering objective is equivalent to maximize a quotient of two linear functions of multiple variables. We can easily utilize the same strategy proposed in Section IV-C4 to prove that there exists a unique optimal assignment for the weights $\beta_1, \ldots, \beta_m$ to maximize the clustering objective function in Eq. (25).

4) Entropy-Based Weight Self-Adjustment: Our dynamic weight tuning method iteratively refines the initial weights $\beta_1^0, \ldots \beta_m^0$ and continuously improves the temporal cycle clustering objective at each iteration. Let $\beta_l^t (l = 1, \ldots, m)$ be the attribute weights for each individual temporal similarity in the t^{th} iteration. All β_l^0s are first initialized as $1/m$. We then iteratively adjust β_l^{t+1} with an increment $\triangle\beta_l^t$. An information-theoretic measure *entropy* is defined to reflect the homogeneity of individual cycle clusters in different semantic environments, i.e., measure how the various attributes with individual semantics are distributed within each cluster.

$$entropy(a_l) = \frac{\beta_l}{\sum_{p=1}^m \beta_p} \sum_{i=1}^k \sum_{j=1}^n - \sum_{q=1}^{n_l} \rho_{lijq} \log \rho_{lijq} \tag{26}$$

where n_l specifies the number of values of attribute a_l. For example, there are four values of 354, 175, 365 and 78 on RR in three cycles in Figure 8. ρ_{lijq} is the percentage of extent v_j in cycle cluster C_i which have the q^{th} value on attribute a_l. A small entropy indicates the generated clusters have higher attribute homogeneity. Namely, if each extent has similar counts for some access pattern in different cycles within cycle clusters, then the entropy for this access pattern is very small. The weight learning mechanism is designed as follows: if $entropy(a_l)$ is small, i.e., each extent has the similar counts on a_l within each cycle cluster, which means it has a good clustering tendency, then the weight β_l should be increased; on the other hand, if $entropy(a_l)$ is large, then β_l should be decreased. The weight increments $\triangle\beta_l^t$ are calculated as

$$\triangle\beta_l^t = \frac{\frac{1}{entropy(a_l)}}{\frac{1}{m}\sum_{p=1}^m \frac{1}{entropy(a_p)}} \tag{27}$$

Similarly, the denominator in Eq. (27) ensures that the constraint $\sum_{l=1}^m \beta_l = m$ in Definition 12 is still satisfied after weight adjustment. Then the adjusted weight is calculated as

$$\beta_l^{t+1} = \frac{1}{2}(\beta_l^t + \triangle\beta_l^t) = \frac{1}{2}\left(\beta_l^t + \frac{\frac{1}{entropy(a_l)}}{\frac{1}{m}\sum_{p=1}^m \frac{1}{entropy(a_p)}}\right) \tag{28}$$

By assembling different parts, our temporal cycle clustering algorithm TC-Cluster is presented in Algorithm 2.

Algorithm 2. Temporal Cycle Clustering

```
Input: a cycle set $T = \{t_i | i = 1, \ldots, m\}$, a cluster number $k$, the initial weights $\beta_p^0 = 1.0$.
Output: $k$ cycle clusters $C_1, \ldots, C_k$.
1: Calculate $s(t_i, t_j)$ for $\forall t_i, t_j \in T$ according to Eqs. (6)–(7);
2: Select $k$ initial centroids with a local maximum of #neighbors;
3: Repeat until $O(\{C_i\}_{i=1}^k, \beta_1, \ldots, \beta_m)$ converges:
4:   Assign each cycle $t_i \in T$ to a cluster $C^*$ with a centroid $c^*$ where $c^* = argmax_{c_j} s(t_i, c_j)$;
5:   Update the centroids with the most centrally located cycle in each cluster according to Eqs. (22)–(23);
6:   Update each weight $\beta_p^t$ according to Eqs. (26)–(28);
7:   Re-calculate $s(t_i, t_j)$ for $\forall t_i, t_j \in T$;
8: Return $C_1, \ldots, C_k$.
```

V. Efficient Computation of Random Walk Similarity

A. Reducing the Number of Matrix Multiplication

One issue with SE-Cluster is the computational complexity. We need to compute $(n+m)^2$ pairs of random walk similarity scores among n structure (extent) vertices and m attribute vertices through matrix multiplication. As $\alpha_{i1}, \ldots, \alpha_{im}$ are updated, the random walk similarity scores need to be recalculated, as shown in lines 7-8 in Algorithm 1. The cost analysis of SE-Cluster can be expressed as follow.

$$t \cdot (T_{random\_walk} + T_{centroid\_update} + T_{assignment} + T_{weight\_update}) \tag{29}$$

where t is the number of iterations in the clustering process, T_{random_walk} is the cost of computing the random walk similarity matrix $R(i)$, $T_{centroid\_update}$ is the cost of updating cluster centroids, $T_{assignment}$ is the cost of assigning all points to cluster centroids, and T_{weight_update} is the cost of updating weights.

The time complexity of $T_{centroid\_update}$ or $T_{assignment}$ is $O(n)$, since each of these two operations performs a linear scan of the graph vertices. The time complexity of T_{weight_update} is $O(n * m)$ since we need to check the value of each extent vertex on each attribute. On the other hand, the time complexity of T_{random_walk} is $O(n^3)$ because the random walk similarity calculation consists of a series of matrix multiplication and addition. According to Eq. (5), $R(i) = \sum_{\gamma=1}^{l} c(1-c)^{\gamma} P(i)^{\gamma}$ where l is the length limit of a random walk. To compute $R(i)$, we have to compute $P(i)^2, P(i)^3, \ldots, P(i)^l$, *i.e.*, $(l-1)$ matrix multiplication operations in total. It is clear that T_{random_walk} is the dominant factor in the clustering process.

In this section, we aim to improve the efficiency of SE-Cluster with a fast matrix computation technique. In other words, we are studying how to compute the random walk similarity matrix $R(i)$ more efficiently. The core idea is to use the Matrix Neumann Series [9] to reduce the number of matrix multiplication operations.

We first introduce some preliminary concepts before discussing the fast random walk computation. Given a square matrix A, $\sum_{i=0}^{\infty} c_i A^i$ is called the *Power Series* of A (assume $A^0 = I$). In particular, when all coefficients c_i $(i = 0, 1, \ldots)$ are equal to 1, the Power series of A is reduced to $\sum_{i=0}^{\infty} A^i$. We call it the *Neumann Series* of A. In related literature, we can find the following theorems on Neumann series. (Theorems 5–9 and the detailed proofs can be found in [9] and [56].)

Theorem 5: The Neumann series of A converges in the normed space X if and only if A is a convergent matrix, *i.e.*, $\lim_{i\to\infty} A^i = 0$.

Theorem 6: An arbitrary square matrix A is convergent if and only if the module of each eigenvalue of A is smaller than 1.

Theorem 7: If the infinite Neumann series of a square matrix A converges in the normed space X, then $I - A$ is invertible and its inverse is the sum of the series.

$$(I - A)^{-1} = \sum_{i=0}^{\infty} A^i \tag{30}$$

where I is the identity matrix in X.

The Neumann series provides approximations of $(I - A)^{-1}$ when A has entries of small magnitude. For example, a first-order approximation is $(I - A)^{-1} \approx I + A$.

We know that a square matrix is invertible if and only if its determinant is not equal to 0 or it has a full rank. A matrix that is invertible is called nonsingular matrix. Singular matrices are rare in the sense that if we pick a random square matrix, it is almost surely not singular.

Theorem 8: If $I - A$ is invertible, then the sum of the finite Neumann series of a square matrix A is as follow.

$$\sum_{i=0}^{l} A^i = (I - A)^{-1} \cdot (I - A^{l+1}) \tag{31}$$

or

$$\sum_{i=1}^{l} A^i = (I - A)^{-1} \cdot (I - A^{l+1}) - I \tag{32}$$

where I is the identity matrix in X.

If we compare the right hand side of Eq. (5), *i.e.*, $\sum_{\gamma=1}^{l} c(1-c)^{\gamma} P(i)^{\gamma}$ with the left hand side of Eq. (32), *i.e.*, $\sum_{i=1}^{l} A^i$, we find that they are very much alike. Eq. (5) is actually the sum of the finite Power series of the transition matrix $P(i)$. If we denote $(1-c)P(i)$ as a new square matrix P' and move the factor c outside of $\sum$, then $R(i)$ is the sum of the finite Neumann series of P'. Based on Eq. (32), we can rewrite the random walk similarity matrix as follow.

$$\begin{aligned} R(i) &= \sum_{\gamma=1}^{l} c(1-c)^{\gamma} P(i)^{\gamma} \\ &= c(I - (1-c)P(i))^{-1} \cdot (I - ((1-c)P(i))^{l+1}) - cI \end{aligned} \tag{33}$$

To compute $((1-c)P(i))^{l+1}$ in Eq. (33), we need to compute $((1-c)P(i))^2$, $((1-c)P(i))^4$, $\ldots$, $((1-c)P(i))^{2^{\lfloor \log(l+1) \rfloor}}$. Thus the number of matrix multiplication operations can be reduced from $(l-1)$ to $(\lceil \log(l+1) \rceil + 1)$. The additional "1" here refers to the computation for $(I - (1-c)P(i))^{-1}$.

We call our clustering algorithm which uses Eq. (33) for random walk similarity computation *SE-Cluster-Opt*, and call the original version which uses Eq. (5) *SE-Cluster*. These two algorithms only differ in the number of matrix multiplication operations, but achieve the same clustering results. We will compare their efficiency in Section VI.

B. Addressing The Non-invertible Matrix

In the above computation defined in Eq. (33), we assume that the matrix $I - (1-c)P(i)$ is invertible. We denote the matrix $C = I - (1-c)P(i)$. If it is not invertible, the Neumann series formula can not be directly applied to solve the problem. In this subsection, we will propose techniques to address this problem. The core idea is to attempt to find an invertible matrix C_n to approximate the non-invertible matrix $C = I - (1-c)P(i)$ while minimizing the difference between the two matrices.

The first step is to construct the Singular Value Decomposition (SVD) form of the matrix $C = I - (1-c)P(i)$. There is a theorem on SVD in [56].

Theorem 9: Suppose r is the rank of an m-by-n matrix C whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a singular value decomposition of the form below.

$$C = U \Sigma V^T \tag{34}$$

where U is an m-by-m unitary matrix over K and the columns of U are the orthogonal eigenvectors of $C^T C$; Σ is an m-by-n diagonal matrix with nonnegative real numbers on the diagonal; and V is an n-by-n unitary matrix over K and the columns of V are the orthogonal eigenvectors of CC^T.

Suppose $\lambda_1, \ldots, \lambda_r$ are the eigenvalues of $C^T C$ or CC^T, with $\lambda_i \geq \lambda_{i+1}$. Let $\sigma_i = \sqrt{\lambda_i}$ for $1 \leq i \leq r$. Then we have $\Sigma_{ii} = \sigma_i$ in the matrix Σ for $1 \leq i \leq r$, and zero otherwise. The diagonal entries Σ_{ii} are ordered in the descending order. In this case, the diagonal matrix Σ is uniquely determined by C but not the matrices U or V. The diagonal entries Σ_{ii} are known as the singular values of C.

With singular value decomposition, we can generate the eigenvalues and eigenvectors of C^TC or CC^T. Given an arbitrary matrix C, we have $rank(CC^T) = rank(C^TC) = rank(C)$. Therefore, a full-rank approximation of C is equivalent to that of C^TC or CC^T.

Next, we will use singular value decomposition to create an invertible matrix C_n to approximate the non-invertible matrix C. Since C_n has a full rank, we call this problem the *full-rank matrix approximation* problem. It is actually a reverse operation of the commonly used low-rank approximation operation. Given an n-by-n square matrix C with a rank r and an n-by-n square matrix C_n with a full rank n, the matrix difference between C and C_n is measured by the Frobenius norm of $X = C_n - C$. The Frobenius norm can be defined as follow.

$$\|X\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^2} \tag{35}$$

Our goal is to find a new matrix C_n that minimizes the Frobenius norm, while constraining C_n with the full rank n. The following three steps are adopted to solve this full-rank approximation problem:

1) Construct the SVD form of an n-by-n square matrix $C = U\Sigma V^T$.
2) A new matrix Σ' is derived by substituting the $n - r$ zero diagonal entries of Σ with different and very small nonzero values.
3) Calculate the full-rank approximation of $C_n = U\Sigma' V^T$.

Theoretical analysis of this full-rank approximation solution is given in the following theorem.

Theorem 10: C_n is the full-rank approximation of C to minimize the Frobenius norm of $X = C_n - C$. See Proof in the Appendix.

Therefore, we can substitute a singular matrix $C = I - (1 - c)P_A$ with a full-rank matrix C_n in the finite Neumann series in Eq. (33).

C. Handling Massive Attributes and Continuous Attributes

When the number of attributes is large, the computational overhead for computing the random walk similarity on the heterogeneous trace graph is nontrivial. In this case, we can perform correlation analysis to detect correlation between attributes and then perform dimensionality reduction to retain a smaller set of orthogonal dimensions. Widely used dimensionality reduction techniques such as principal component analysis (PCA) and multifactor dimensionality reduction (MDR) can be used to create a mapping from the original space to a new space with fewer dimensions. According to the mapping, we can compute the new attribute values of an extent vertex based on the values of its original attributes. Then we can construct a more compact heterogeneous trace graph in the new feature space and perform graph clustering.

To handle continuous attributes, discretization can be applied to convert them to nominal features. Typically the continuous values are discretized into K partitions of an equal interval (equal width) or K partitions each with the same number of extents (equal frequency). In our experiment, there are four attributes: RR, RW, SR and SR for each extent in the trace datasets indicating whether the extent has frequent or rare access on a certain access pattern. The access account achieved by an extent on an access pattern is a continuous attribute. According to the distribution of the access accounts in the trace datasets, we discretized the access accounts into 12 partitions: [0, 0], [1, 100], [101, 200], [201, 300], [301, 400], [401, 500], [501, 600], [601, 700], [701, 800], [801, 900], [901, 1000], [1001, +∞).

TABLE II
TRACE DATASET SUMMARY

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

VI. 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. All three storage trace datasets are collected every fifteen minutes. A cycle represents an interval of five minutes and thus they will be three-cycle apart. 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 with the corresponding access count to each data unit within the cycle. We totally construct 2,013, 2,008 and 2,011 trace graphs for three storage traces respectively.

A. Spatial Correlation Analysis

Figures 10(a), (b), (c) and (d) show the individual spatial similarity comparison based on each access pattern in the bank trace. Both the x-axis and y-axis represent 8,097 extents in a cycle. The similarity score is on a scale of 0-1 with 1 indicating highest similarity. Figure 10(a) presents the spatial similarity between any pair of extents based on random read access pattern. We observe that a large number of extents across different volumes are very similar in behavior with respect to random reads. The similarity matrices in Figure 10(b) (random write) and Figure 10(d) (sequential write) are very similar to each other but different from random read patterns. The same

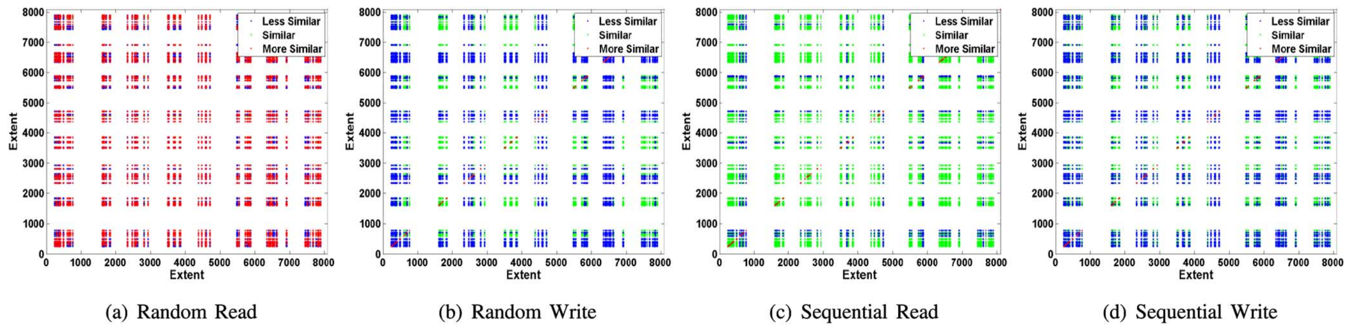


Fig. 10. Extent Similarity on Bank Trace.

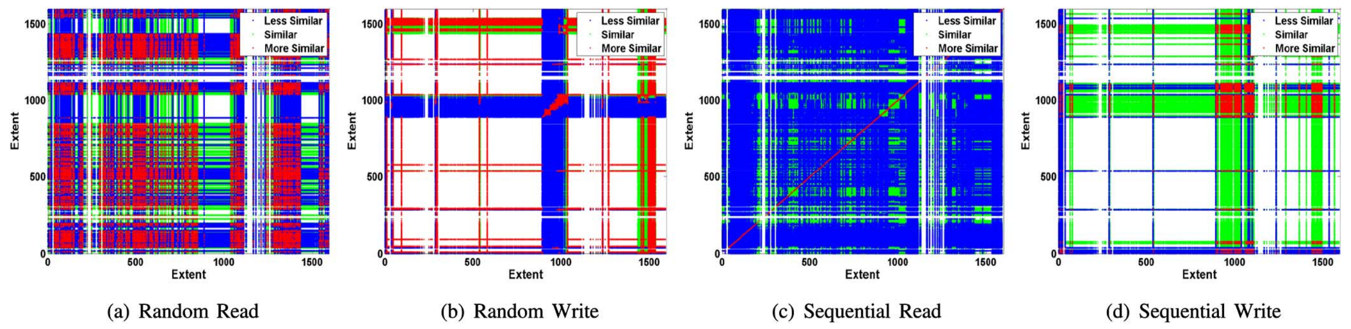


Fig. 11. Extent Similarity on Email Trace.

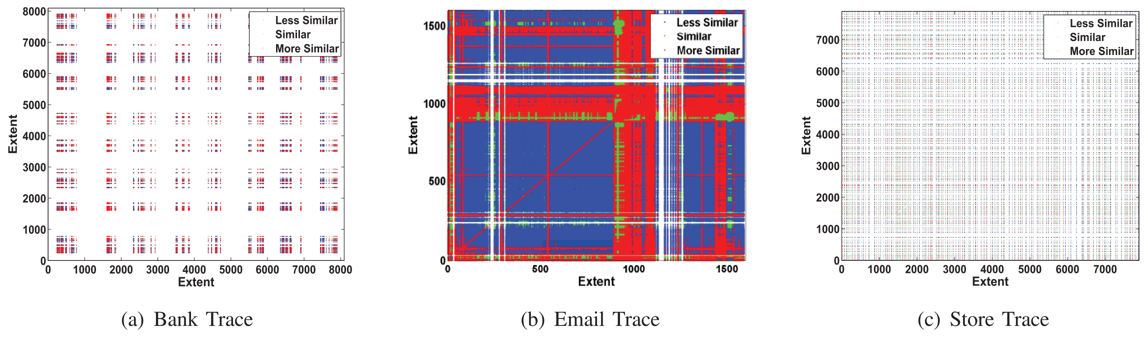


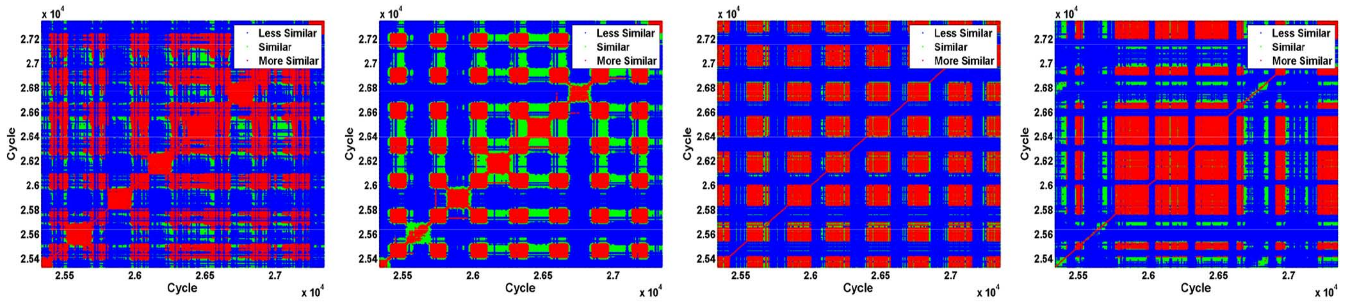
Fig. 12. Unified Extent Similarity on Different Traces.

extents that exhibit strong similarity for random read accesses, exhibit weak similarity for random write and sequential write accesses.

Figures 11(a), (b), (c) and (d) represent the spatial similarity matrix based on each access pattern for the email trace. Figure 11(a) shows that most of spatial similarity between extents based on random reads shows several regions of strong similarity. Next observe that 11(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 11(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.

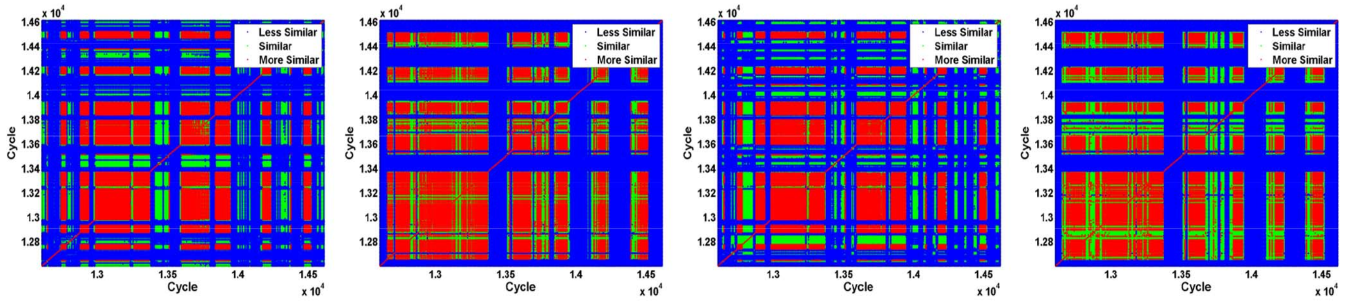
- **Observation 1:** Similar behaviors are exhibited both within and across volumes.
- **Observation 2:** Spatial similarity varies by the dimension under consideration.
- **Observation 3:** Data expresses stronger similarity under the random read access pattern.

Figures 12(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-C 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 12(a) is similar to the similarity matrix in Figure 10(a). This demonstrates that the unified random walk similarity on Bank Trace is



(a) Random Read (b) Random Write (c) Sequential Read (d) Sequential Write

Fig. 13. Cycle Similarity on Bank Trace.



(a) Random Read (b) Random Write (c) Sequential Read (d) Sequential Write

Fig. 14. Cycle Similarity on Email Trace.

dominated by the access pattern of random read. In comparison with Figures 11(b) and (d), the similarity matrix in Figure 12(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 12(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.

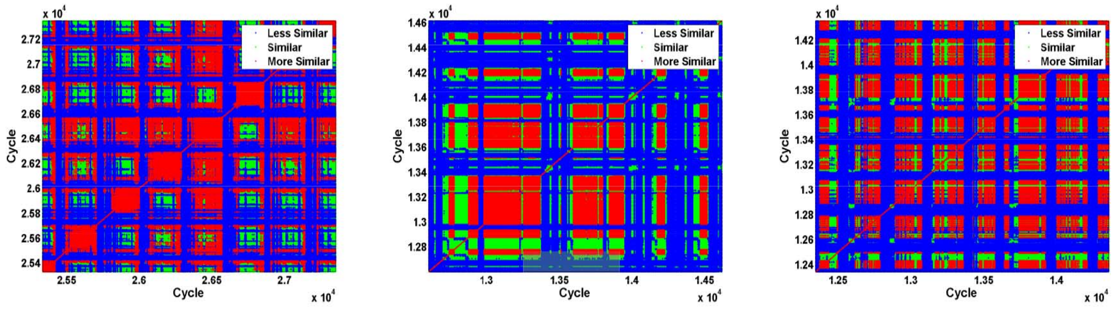
- **Observation 4:** When all data exhibits all types of access pattern, the strongest access pattern (which is most often random read) dominates the similarity metric. This implies that data placement taking only random read patterns into consideration is likely to provide good results.
- **Observation 5:** When access type distributions are not uniform across all extents, extents that exhibit more rare access patterns exhibit 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 12(c), this indicates that there is no need to further explore attribute-specific spatial access patterns.

B. Temporal Correlation Analysis

Figures 13(a), (b), (c) and (d) show the cycle similarity matrix based on each individual access pattern on Bank Trace. Both the x-axis and y-axis represent a period of one week with 2,013 cycles in the trace dataset. Although the distribution of cycle similarity scores in each figure is quite different, the following observations hold true for all figures.

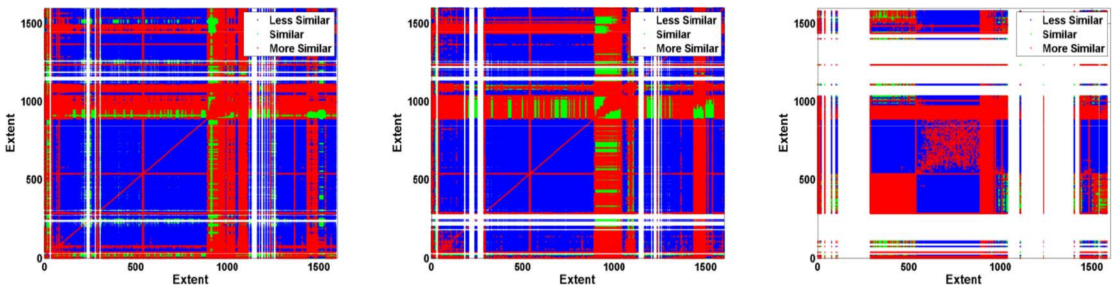
- **Observation 7:** The red areas of "More Similar" are usually symmetric rectangle blocks, i.e., most of cycles are more similar to its immediate neighboring cycles.
- **Observation 8:** There is clear periodicity exhibited by the workloads. The red blocks occur periodically row-wise and column-wise, i.e., the cycles similar to a certain cycle occur periodically. The length of each block and the length of the region in between gives us the repeating period of the workload for that specific access pattern.

Figures 14(a), (b), (c) and (d) present the cycle similarity comparison based on each access pattern on Email Trace. The distributions of cycle similarity scores in Figure 14 are quite different from the distributions in Figure 13: The distribution of "More similar" cycles in Figure 14 are very dense in some cycles but are relatively sparse in some cycles. By checking the original trace dataset, we first divide 2,011 cycles into parts: the first 30% of cycles correspond to the workloads of Saturday and Sunday, the next 14% corresponds to workload on Monday and the next 14% corresponds to the workloads of a holiday; the remain cycles correspond to the rest of the week (workdays). We observe the following phenomena.



(a) Bank Trace (b) Email Trace (c) Store Trace

Fig. 15. Unified Cycle Similarity on Different Traces.



(a) Cycle 12,605 (b) Cycle 12,608 (c) Cycle 13,541

Fig. 16. Cycle Comparison by Extent Similarity on Email Trace.

- **Observation 9:** There are very few or no access activities during weekends and holidays. As a result the extents within each cycle become relatively cold.
- **Observation 10:** Some extents have heavy workloads but some extents have light workloads in workday cycles such that the corresponding extents in different cycles have quite different access patterns and the cycle similarity scores among the workday cycles are relatively random.

Figures 15(a), (b) and (c) exhibit the unified cycle similarity matrix on different trace datsets. We exploit our dynamic entropy-based weight adjustment approach in Section IV-D to learn an optimal weight assignment for cycle similarities based on four kinds of attributes: RR, RW, SR and SW, to achieve high intra-cluster similarity and low inter-cluster similarity, i.e., the corresponding extents and their neighbors within each cycle cluster have similar access patterns on all attributes, while the corresponding extents and their neighbors in different cycle clusters may have diverse access patterns. The similarity matrix in Figure 15(a) is different from any similarity matrix in Figure 13(a), (b), (c) and (d), i.e., decided by all four similarity matrices in Figure 13. We check the original bank trace dataset and observe that the workloads on each access pattern are evenly distributed to different extents or diverse cycles such that the workloads on any access pattern does not dominate the unified cycle similarity matrix. In comparison with Figures 14(c), the similarity matrix in Figure 15(b) is mainly dominated by the access pattern of sequential read since the workloads on sequential read are much more than each of other three kinds of workloads. Figure 15(c) shows the unified cycle similarity matrix on Store Trace, which depends on all similarity matrices on four access patterns since customers' purchasing behaviors are relatively random during daytime. This leads us to the next observation.

- **Observation 11:** when the similarity scores based on each access pattern have the similar weighting factors. This implies that we should first resort to the unified metric to identify cycles with heavy workloads to guide us to discover extents with frequent access activities in those cycles.

Figures 16(a), (b) and (c) present the unified cycle similarity comparison on Email Trace by using the unified extent similarity matrix in each cycle. According to the distribution of extent similarity scores, Cycle 12,605 are very similar to its successor, Cycle 12,608 but it is less similar to Cycle 13,541. This demonstrates that our unified cycle similarity definition is a good metric to measure the similar extent of two cycles. We check the original email trace dataset and make the following observations about Cycle 12,605 and Cycle 12,608: (1) most of corresponding extents in two cycles have similar access patterns with similar access counts; and (2) the corresponding predecessors and successors for each extent in two cycles also have similar access patterns with similar access counts. On the other hand, we observe that most of same extents in Cycle 12,608 and Cycle 13,541 also have similar access patterns with similar access counts. however, the corresponding predecessors and successors for each extent in these two cycles have quite different access patterns or relatively diverse access counts.

Figures 16(a), (b) and (c) present the unified cycle similarity comparison on Email Trace by using the unified extent similarity matrix in each cycle. According to the distribution of extent similarity scores, Cycle 12,605 are very similar to its successor, Cycle 12,608 but it is less similar to Cycle 13,541. This

demonstrates that our unified cycle similarity definition is a good metric to measure the similar extent of two cycles. We check the original email trace dataset and make the following observation about Cycle 12,605 and Cycle 12,608.

- **Observation 12:** Most of corresponding extents in two cycles have similar access patterns with similar access counts; and the corresponding predecessors and successors for each extent in two cycles also have similar access patterns with similar access counts.
- **Observation 13:** Most of same extents in Cycle 12,608 and Cycle 13,541 also have similar access patterns with similar access counts. however, the corresponding predecessors and successors for each extent in these two cycles have quite different access patterns or relatively diverse access counts.

C. Clustering Quality Evaluation

We compare our **SE-Cluster** and **TC-Cluster** with two representative non-graph clustering algorithms, K-Medoids [54] and K-Means [53]. For the last two clustering methods, we combine the attribute-based similarity scores on each of multiple access patterns into a unified similarity matrix with the equal weighting factors. The optimized version SE-Cluster-Opt based on Neumann series is also tested for efficiency evaluation.

We use two measures to evaluate the quality of extent clustering by different methods. We report the average metric value for each measure based on N trace graphs, i.e., N cycles. The Dunn index [57] is defined as the ratio between the minimal intra-cluster similarity and the maximal inter-cluster similarity.

$$Dunn(\{C_l\}_{l=1}^{k_i}) = \frac{\min\limits_{1\le p\le k_i} \sum_{u,v\in C_p} s_i(u,v)}{\max\limits_{1\le p<q\le k_i} \sum_{u\in C_p, v\in C_q} s_i(u,v)} \tag{36}$$

where $Dunn(\{C_l\}_{l=1}^{k_i})$ is bounded in the range $[0, +\infty)$. A larger value of $Dunn(\{C_l\}_{l=1}^{k_i})$ indicates a better clustering.

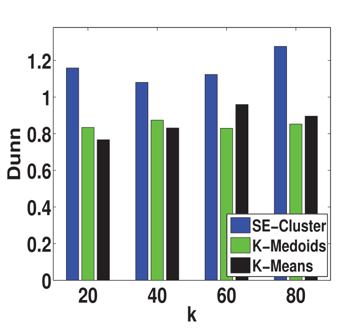
The *Davies-Bouldin Index (DBI)* [57] measures the uniqueness of clusters with respect to the unified similarity measure. It tends to identify set of clusters that are compact and well separated in terms of the unified similarity.

$$DBI(\{C_l\}_{l=1}^{k_i}) = \frac{1}{k_i}\sum_{p=1}^{k_i} max_{q\neq p}\frac{s_i(c_p,c_q)}{\sigma_p+\sigma_q} \tag{37}$$

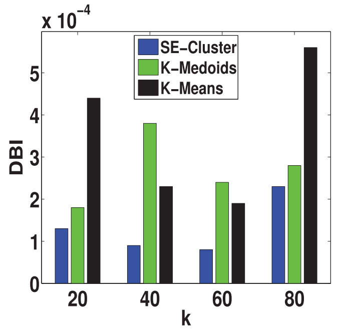
where c_x is the centroid of cluster C_x, $s_i(c_p, c_q)$ is the unified similarity between two centroids c_p and c_q, σ_x is the average similarity of extents in C_x to c_x. The smaller the value, the better the quality.

Similarly, we use the same two measures to evaluate the quality of cycle clustering by different clustering algorithms.

Figures 17 and 18 exhibit the extent clustering quality on Store Trace and Email Trace by varying the number of clusters. First, among all three clustering methods, SE-Cluster achieves the best clustering performance for both evaluation measures in most cases. Compared to other algorithms, SE-Cluster averagely achieves 36.1% Dunn increase and 157.2% DBI improvement on Store Trace, 28.6% Dunn growth and 395.7% DBI

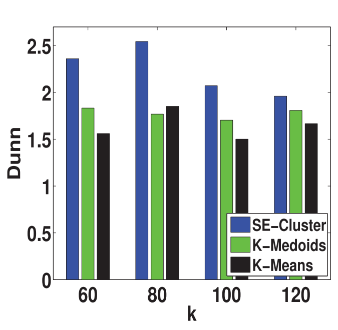


(a) Dunn

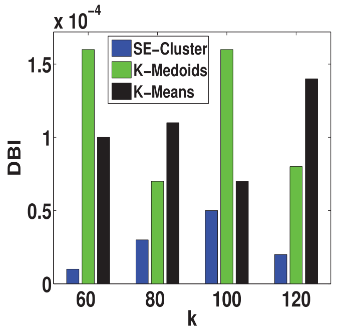


(b) DBI

Fig. 17. Extent Clustering Quality on Store Trace.



(a) Dunn



(b) DBI

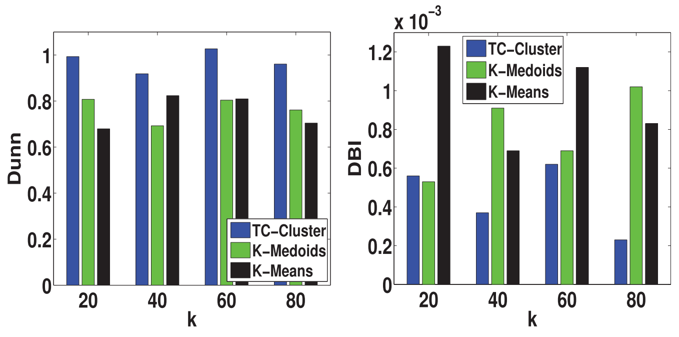
Fig. 18. Extent Clustering Quality on Email Trace.

decrease on Email Trace, respectively. Concretely, there are two critical reasons for high accuracy of SE-Cluster: (1) the unified random walk similarity based on graph model provides a natural way to capture both direct and indirect dependencies between extents within each trace graph; and (2) the iterative learning algorithm help the clustering model learn the optimal weight assignment for different types of access patterns to achieve a good balance between different types of attribute edges in the heterogeneous trace graph. Second, it is observed that K-Medoids usually outperforms K-Means on two datasets. This is because K-Medoids improves the clustering quality by restricting the centroids to real extents of the dataset. In addition, in contrast to K-Means, K-Medoids is more robust to noise and outliers because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

Figures 19 and 20 present the cycle clustering quality by three algorithms on two datasets with different k respectively. Similar trends are observed for the cycle clustering quality comparison: TC-Cluster achieves the largest Dunn values (> 0.92) and the lowest DBI around 0.00006-0.00062, which are obviously better than other two methods. As k increases, the measure scores achieved by TC-Cluster remain relatively stable, while the measure scores of other two methods oscillate in a fairly large range.

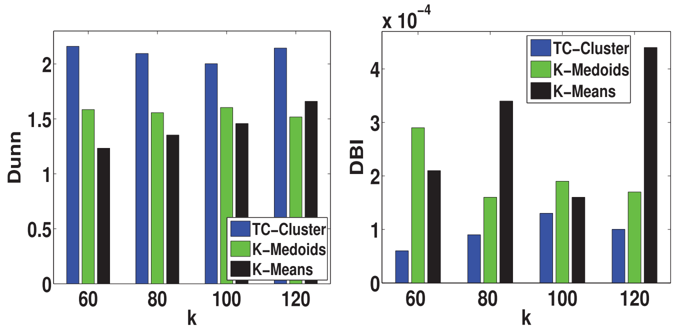
D. Clustering Efficiency Evaluation

In this experiment, we compare the efficiency of different clustering algorithms in Figure 21 on Store Trace and Email



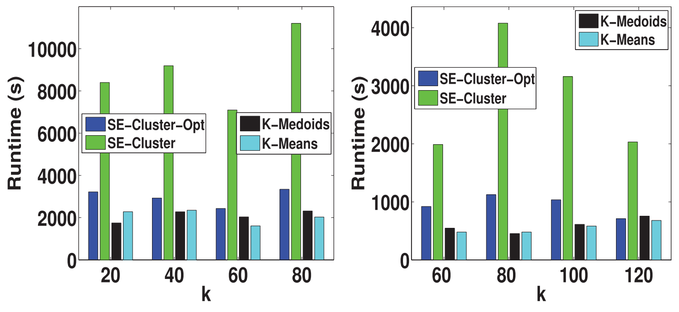
(a) Dunn (b) DBI

Fig. 19. Cycle Clustering Quality on Store Trace.



(a) Dunn (b) DBI

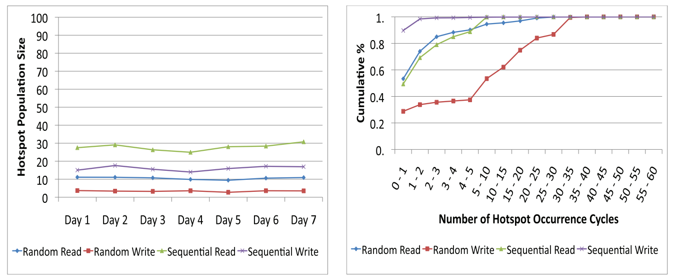
Fig. 20. Cycle Clustering Quality on Email Trace.



(a) Store Trace (b) Email Trace

Fig. 21. Clustering Efficiency.

Trace respectively. Figure 21 shows that all methods have a short clustering runtime on two datasets except SE-Cluster. From two figures, we can observe that K-Medoids and K-Means are the most efficient. SE-Cluster is usually 3 to 8 times slower than these two non-graph clustering methods, as it iteratively updates the weights, computes the unified random walk similarity and performs clustering, while K-Medoids and K-Means calculate the unified attribute-based similarity scores only once. In addition, we observe that SE-Cluster-Opt significantly reduces the runtime cost of SE-Cluster to around $\frac{1}{4} - \frac{1}{2}$, while achieving the same clustering quality. This result shows that, with the Neumann series technique, SE-Cluster only causes a relatively small overhead compared with SE-Cluster.



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

Fig. 22. Bank Trace.

E. Hotspot Characterization

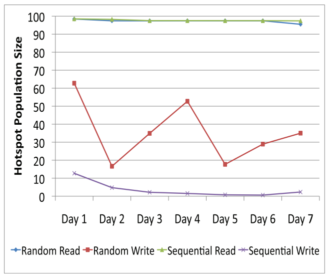
Hotspots can be defined as regions that have relatively higher activity (hence "temperature" or "heat") in comparison to its surroundings. Understanding hotspot characteristics is essential to data placement strategies, such as caching at host or storage server by utilizing the "recency" [3], [58], and tiering by exploring the "frequency" aspect [4], [59].

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 two measures to classify hotspots from the temporal and spatial clustering analysis results.

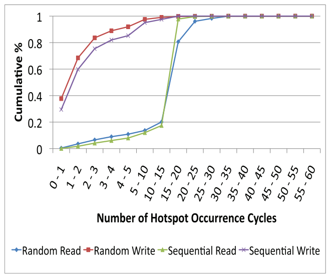
- **Population Size:** A summarization measure which describes the number of unique extents that exhibit hotspot behavior within a window (24 hours in this study) of observation i.e. size of the hotspot.
- **Intra-window Stability or Burstiness:** The frequency distribution of number of hotspot occurrences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

Banking Transactions Workload. Figure 22(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 22(b) shows the frequency distribution during a 24-hour period for which an extent exhibits 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 the other hand, sequential

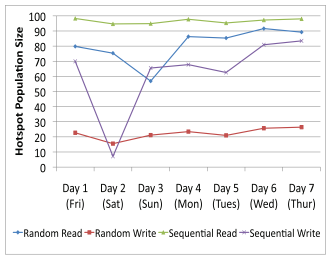


(a) Store Trace: Population Size

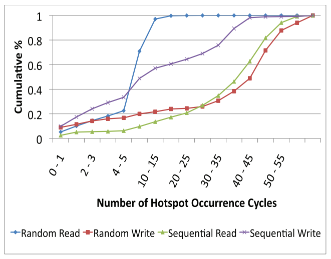


(b) Store Trace: Burstiness

Fig. 23. Store Trace.



(a) Email Trace: Population Size



(b) Email Trace: Burstiness

Fig. 24. Email Trace.

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 read and sequential read 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 write exhibits highly bursty behavior, which could be addressed with prefetch caching.

Store Backend Workload. In the Store trace Figure 23(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.

In Figure 23(b), sequential read and random read show very identical behavior with 80% of the extents exhibit hot spot behavior for nearly 4 to 5 hours a day. In comparison, Sequential write and random write 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).

Email Server Workload. In this trace, unlike the other workloads where the behavior was uniform across days, we see a clear distinction between workload characteristics on regular working days and holidays. Based on public holidays at the customer location, Saturday (half-day), Sunday and Tuesday (not confirmed) may have been holidays.

Figure 24(a) shows the population size of hotspots for the four different access patterns. First, we observe that the average population sizes are very high for sequential read (96%) and random read (80%), moderate for Sequential write (62%) and low for random write (22%). Also, note that random read and sequential write patterns are impacted on holidays.

Next, Figure 24(b) shows the burstiness of various access patterns in the workload. Note that sequential read and random write are the least bursty (75% of hotspots lasting longer than 7.5 hours in a day) followed by sequential write (30% of the hotspots lasting for longer than 7.5 hours). Random read is relatively burstier, without hotspots lasting longer than 7.5 hours and yet, 70% of the hotspots last between 1-5 hours. Although bursty compared to the other access patterns, it may still be considered as a long-duration hotspot.

The observations above lead us to the following conclusions. Sequential read exhibits large population size (almost spanning the entire data), low burstiness and good stability across days but should be treated as a secondary optimization criteria since the skew is poor and population size is large. As a result, one may be able to address this workload by proactive data placement or prefetch algorithms based on temporal analysis. Random write with small population size, low burstiness and high overlap can be addressed by tiering. Random read and sequential write demonstrate large population size, mixed burstiness and significant overlap between days. These two types of workloads may be addressed with a combination of high performance tiers and caching.

VII. Conclusions

We have presented a novel graph analytics framework, GraphLens, for mining and analyzing real storage traces. By modeling storage traces as heterogeneous trace graphs to incorporate multiple complex and heterogeneous interactions among storage entities into a unified network analytic framework, we show that we can develop an innovative two-level graph clustering method to identify and discover spatial/temporal correlations and hotspot characterization. To combine multiple correlations from different spatial and temporal interaction networks, we design a dynamic weight tuning method and a unified similarity measure with optimal weight assignment scheme to integrate multiple trace graphs into a unified network analytic framework by employing a heterogeneous graph clustering method. We propose the optimization technique of full-rank approximation to further improve the efficiency of our clustering algorithm on large trace datasets. Extensive evaluation shows the effectiveness of GraphLens for deriving deep insights from graph based trace analysis. This intelligent tool can be applied to both a single PC with multiple disks and a distributed network across a cluster of compute nodes to offer a few opportunities for optimization of storage performance such as better storage strategy planning and efficient data placement guidance.

Appendix: Proofs of Theorems

Theorem 1: The NFPP in Definition 8 is equivalent to the NPPP in Definition 9, i.e., $\gamma = \max_{\alpha_{i1},\ldots,\alpha_{im}} \frac{f(\alpha_{i1},\ldots,\alpha_{im})}{g(\alpha_{i1},\ldots,\alpha_{im})}$ if and only if $F(\gamma) = \max_{\alpha_{i1},\ldots,\alpha_{im}} f(\alpha_{i1},\ldots,\alpha_{im}) - \gamma g(\alpha_{i1},\ldots,\alpha_{im}) = 0$.

Proof: If $(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})$ is a feasible solution of $F(\gamma) = 0$, then $f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) = 0$. Thus $f(\alpha_{i1},\ldots,\alpha_{im}) - \gamma g(\alpha_{i1},\ldots,\alpha_{im}) \leq f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) = 0$. We have $\gamma = f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})/g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) \geq f(\alpha_{i1},\ldots,\alpha_{im})/g(\alpha_{i1},\ldots,\alpha_{im})$. Thus γ is a maximum value of NFPP and $(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})$ is an optimal solution of NFPP.

Conversely, if $(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})$ solves NFPP, then we have $\gamma = f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})/g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) \geq f(\alpha_{i1},\ldots,\alpha_{im})/g(\alpha_{i1},\ldots,\alpha_{im})$. Thus $f(\alpha_{i1},\ldots,\alpha_{im}) - \gamma g(\alpha_{i1},\ldots,\alpha_{im}) \leq f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) = 0$. We have $F(\gamma) = 0$ and the maximum is taken at $(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})$.

Theorem 2: $F(\gamma)$ is convex.

Proof: Suppose that $(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})$ is an optimum of $F((1-\lambda)\gamma_1 + \lambda\gamma_2)$ with $\gamma_1 \neq \gamma_2$ and $0 \leq \lambda \leq 1$. $F((1-\lambda)\gamma_1 + \lambda\gamma_2) = f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - ((1-\lambda)\gamma_1 + \lambda\gamma_2) g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) = \lambda(f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma_2 g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})) + (1-\lambda)(f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma_1 g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})) \leq \lambda \max_{\alpha_{i1},\ldots,\alpha_{im}} f(\alpha_{i1},\ldots,\alpha_{im}) - \gamma_2 g(\alpha_{i1},\ldots,\alpha_{im}) + (1-\lambda) \max_{\alpha_{i1},\ldots,\alpha_{im}} f(\alpha_{i1},\ldots,\alpha_{im}) - \gamma_1 g(\alpha_{i1},\ldots,\alpha_{im}) = \lambda F(\gamma_2) + (1-\lambda)F(\gamma_1)$. Thus, $F(\gamma)$ is convex.

Theorem 3: $F(\gamma)$ is monotonically decreasing.

Proof: Suppose that $\gamma_1 > \gamma_2$ and $(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im})$ is an optimal solution of $F(\gamma_1)$. Thus, $F(\gamma_1) = f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma_1 g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) < f(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) - \gamma_2 g(\overline{\alpha}_{i1},\ldots,\overline{\alpha}_{im}) \leq \max_{\alpha_{i1},\ldots,\alpha_{im}} f(\alpha_{i1},\ldots,\alpha_{im}) - \gamma_2 g(\alpha_{i1}, \cdots, \alpha_{im}) = F(\gamma_2)$.

Theorem 4: $F(\gamma) = 0$ has a unique solution.

Proof: Based on the above-mentioned theorems, we know $F(\gamma)$ is continuous as well as decreasing. In addition, $lim_{\gamma\to+\infty}F(\gamma) = -\infty$ and $lim_{\gamma\to-\infty}F(\gamma) = +\infty$.

Theorem 5: C_n is the full-rank approximation of C to minimize the Frobenius norm of $X = C_n - C$.

Proof: Since the Frobenius norm is unitarily invariant and the SVD form of C is $C = U\Sigma V^T$, an equivalent statement can be generated as follow.

$$min_{C_n} \|U^T C_n V - \Sigma\|_F \tag{38}$$

as $U^T U = V^T V = I$. According to step (3) listed above, we compute C_n by $U\Sigma' V^T$. Thus we have

$$U^T C_n V = U^T(U\Sigma' V^T)V = \Sigma' \tag{39}$$

Thus,

$$\begin{aligned} min_{C_n} \|U^T C_n V - \Sigma\|_F &= min_{C_n} \|\Sigma' - \Sigma\|_F \\ &= min_{\sigma_i'} \sqrt{\sum_{i=1}^{n} (\sigma_i' - \sigma_i)^2} \end{aligned} \tag{40}$$

where Σ has r non-zero diagonal entries and Σ' has n non-zero diagonal entries. When $\sigma_i = \sigma_i'$ for $1 \leq i \leq r$, the above equation is equivalent to

$$min_{\sigma_i'} \sqrt{\sum_{i=1}^{r} (\sigma_i' - \sigma_i)^2 + \sum_{i=r+1}^{n} \sigma_i'^2} = min_{\sigma_i'} \sqrt{\sum_{i=r+1}^{n} \sigma_i'^2} \tag{41}$$

Here σ_i' for $1 \leq i \leq n$ are different from each other since both $C_n^T C_n$ and $C_n C_n^T$ are full rank matrices. To minimize Eq. (40), σ_i' for $r+1 \leq i \leq n$ should be very small positive numbers. Thus, the above equation is close to zero.

$$min_{\sigma_i'} \sqrt{\sum_{i=r+1}^{n} \sigma_i'^2} \approx 0 \tag{42}$$

Thus, C_n is the full-rank approximation of C to minimize the Frobenius norm when $\sigma_i' = \sigma_i$ for $1 \leq i \leq r$ and $\sigma_i' \approx 0$ for $r+1 \leq i \leq n$.

References

[1] Y. Zhou, S. Seshadri, L. Chiu, and L. Liu, "Graphlens: Mining enterprise storage workloads using graph analytics," in *Proc. IEEE Int. Congr. Big Data (BigData'14)*, Anchorage, AK, USA, Jun. 27/Jul. 2 2014, pp. 1–8.

[2] M. Bhadkamkar *et al.*, "Borg: Block-reorganization for self-optimizing storage systems," in *Proc. 7th Conf. File Storage Technol. (FAST'09)*, San Francisco, CA, USA, Feb. 2009, pp. 183–196.

[3] Y. Zhang *et al.*, "Warming up storage-level caches with bonfire," in *Proc. 11th USENIX Conf. File Storage Technol. (FAST'13)*, San Jose, CA, USA, Feb. 2013, pp. 59–72.

[4] 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 *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST'10)*, Washington, DC, 2010, pp. 1–6.

[5] 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 *Proc. 4th USENIX Conf. Hot Topics Storage File Syst. (HotStorage'12)*, Boston, MA, USA, Jun. 12–15, 2012, p. 3.

[6] V. Tarasov, D. Hildebrand, G. Kuenning, and E. Zadok, "Virtual machine workloads: The case for new benchmarks for nas," in *Proc. 11th USENIX Conf. File Storage Technol. (FAST'13)*, San Jose, CA, USA, Feb. 2013, pp. 307–320.

[7] A. Gulati, C. Kumar, I. Ahmad, and K. Kumar, "Basil: Automated IO load balancing across storage devices," in *Proc. 8th USENIX Conf. File Storage Technol. (FAST'10)*, San Jose, CA, USA, Feb. 2010, p. 13.

[8] N. Park, I. Ahmad, and D. J. Lilja, "Romano: Autonomous storage management using performance prediction in. multitenant datacenters," in *Proc. 3rd ACM Symp. Cloud Comput. (SoCC'12)*, San Jose, CA, USA, 2012, pp. 21:1–21:14.

[9] G. Strang, *Linear Algebra and Its Applications*, 4th ed. Pacific Grove, CA, USA: Brooks/Cole, 2005.

[10] Y. Chen, K. Srinivasan, G. R. Goodson, and R. H. Katz, "Design implications for enterprise storage systems via multi-dimensional trace analysis," in *Proc. 23rd ACM Symp. Oper. Syst. Princ. (SOSP'11)*, Cascais, Portugal, Oct. 23–26, 2011, pp. 43–56.

[11] S. Kavalanekar, B. L. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production windows servers," in *Proc. IEEE Int. Symp. Workload Charact. (IISWC'08)*, Seattle, WA, USA, Sep. 14–16, 2008, pp. 119–128.

[12] A. W. Leung, S. Pasupathy, G. Goodson, and E. L. Miller, "Measurement and analysis of large-scale network file system workloads," in *Proc. USENIX Annu. Tech. Conf. (ATC'08)*, Boston, MA, USA, Sep. 14–16, 2008, pp. 213–226.

[13] G. Wallace *et al.*, "Characteristics of backup workloads in production systems," in *Proc. 10th USENIX Conf. File Storage Technol. (FAST'12)*, San Jose, CA, USA, Feb. 2012, p. 4.

[14] V. Tarasov, "Multi-dimensional workload analysis and synthesis for modern storage systems," Ph.D. dissertation, Department of Computer Science, Stony Brook Univ., Stony Brook, NY, USA, Dec. 2013.

[15] B. Taskar, E. Segal, and D. Koller, "Probabilistic classification and clustering in relational data," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI'01)*, 2001, pp. 870–878.

[16] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD'09)*, Paris, France, Jun. 28/Jul. 1, 2009, pp. 927–936.

[17] M. Ji, J. Han, and M. Danilevsky, "Ranking-based classification of heterogeneous information networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD'11)*, San Diego, CA, USA, Aug. 2011, pp. 1298–1306.

[18] X. Yu, Y. Sun, P. Zhao, and J. Han, "Query-driven discovery of semantically similar substructures in heterogeneous networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD'12)*, Beijing, China, Aug. 2012, pp. 1500–1503.

[19] 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 *Proc. 20th Int. Conf. Web Serv. (ICWS'13)*, Santa Clara, CA, USA, Jun. 27/Jul. 2, 2013, pp. 26–33.

[20] Y. Zhou and L. Liu, "Activity-edge centric multi-label classification for mining heterogeneous information networks," in *Proc. 20th ACM SIGKDD Conf. Knowl. Discov. Data Min. (KDD'14)*, New York, NY, USA, Aug. 24–27 2014, pp. 1276–1285.

[21] Y. Zhou *et al.*, "Clustering service networks with entity, attribute and link heterogeneity," in *Proc. 22nd Int. Conf. Web Serv. (ICWS'15)*, New York, NY, USA, Jun. 27/Jul. 2, 2015, pp. 257–264.

[22] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proc. 7th Int. World Wide Web Conf. (WWW'98)*, Brisbane, Australia, Apr. 1998, pp. 107–117.

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

[24] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[25] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proc. 12th Int. Conf. World Wide Web (WWW'03)*, Budapest, Hungary, May 20–24, 2003, pp. 640–651.

[26] H. Hwang, V. Hristidis, and Y. Papakonstantinou, "Objectrank: A system for authority-based search on databases," in *Proc. ACM-SIGMOD Int. Conf. Manage. Data (SIGMOD'06)*, Chicago, IL, USA, Jun. 2006, pp. 796–798.

[27] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, 2004.

[28] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: A structural clustering algorithm for networks," in *Proc. Int. Conf. Knowl. Discov. Data Min. (KDD'07)*, San Jose, CA, USA, Aug. 2007, pp. 824–833.

[29] M. Shiga, I. Takigawa, and H. Mamitsuka, "A spectral clustering approach to optimally combining numericalvectors with a modular network," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Databases (KDD'07)*, San Jose, CA, USA, 2007, pp. 647–656.

[30] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *Proc. ACM-SIGMOD Int. Conf. Manage. Data (SIGMOD'08)*, Vancouver, Canada, Jun. 2008, pp. 567–580.

[31] V. Satuluri and S. Parthasarathy, "Scalable graph clustering using stochastic flows: Applications to community discovery," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD'09)*, Paris, France, Jun. 2009, pp. 737–746.

[32] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," in *Proc. 35th Int. Conf. Very Large Data Bases (VLDB'09)*, Lyon, France, Aug. 24–28, 2009, pp. 718–729.

[33] K. Macropol and A. Singh, "Scalable discovery of best clusters on large graphs," in *Proc. 36th Int. Conf. Very Large Data Bases (VLDB'10) /PVLDB 3(1)*, Sep. 2010, pp. 693–702.

[34] Y. Zhou, H. Cheng, and J. X. Yu, "Clustering large attributed graphs: An efficient incremental approach," in *Proc. 10th IEEE Int. Conf. Data Min. (ICDM'10)*, Sydney, Australia, Dec. 14–17, 2010, pp. 689–698.

[35] E. C. Kenley and Y.-R. Cho, "Entropy-based graph clustering: Application to biological and social networks," in *Proc. 11th Int. Conf. Data Min. (ICDM'11)*, Vancouver, Canada, Dec. 11–14, 2011, pp. 1116–1121.

[36] Y. Zhou and L. Liu, "Clustering analysis in large graphs with rich attributes," in *Data Mining: Foundations and Intelligent Paradigms: Clustering, Association and Classification*, vol. 1, D. E. Holmes and L. C. Jain, Eds. New York, NY, USA: Springer, 2011.

[37] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proc. 2012 ACM SIGMOD Int. Conf. Manage. Data (SIGMOD'12)*, Scottsdale, AZ, USA, May 20–24, 2012, pp. 505–516.

[38] Y. Zhou and L. Liu, "Social influence based clustering of heterogeneous information networks," in *Proc. 19th ACM SIGKDD Conf. Knowl. Discov. Data Min. (KDD'13)*, Chicago, IL, USA, Aug. 11–14, 2013, pp. 338–346.

[39] Y. Zhou and L. Liu, "Social influence based clustering and optimization over heterogeneous information networks," *ACM Trans. Knowl. Discov. Data (TKDD)*, vol. 10, pp. 1–53, 2015.

[40] Y. Zhou, L. Liu, and D. Buttler, "Integrating vertex-centric clustering with edge-centric clustering for meta path graph analysis," in *Proc. 21st ACM SIGKDD Conf. Knowl. Discov. Data Min. (KDD'15)*, Sydney, Australia, Aug. 10–13, 2015, pp. 1563–1572.

[41] Y. Sun, C. C. Aggarwal, and J. Han, "Relation strength-aware clustering of heterogeneous information networks with incomplete attributes," *Proc. VLDB Endowment (PVLDB)*, vol. 5, no. 5, pp. 394–405, 2012.

[42] G. Malewicz *et al.*, "Pregel: A system for large-scale graph processing," in *Proc. Int. Conf. Manage. Data (SIGMOD'10)*, Indianapolis, IN, USA, Jun. 6–11, 2010, pp. 135–146.

[43] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endowment (PVLDB)*, vol. 5, no. 8, pp. 716–727, 2012.

[44] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *Proc. 10th USENIX Symp. Oper. Syst. Des. Implement. (OSDI'12)*, Hollywood, CA, USA, Oct. 2012, pp. 17–30.

[45] Apache Giraph. (Feb. 6, 2012). *Giraph* [Online]. Available: http://giraph.apache.org/

[46] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: Graph processing in a distributed dataflow framework," in *Proc. 11th USENIX Symp. Oper. Syst. Des. Implement. (OSDI'14)*, Broomfield, CO, USA, Oct. 2014, pp. 599–613.

[47] K. Lee *et al.*, "Scaling iterative graph computations with graphmap," in *Proc. 27th IEEE Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC'15)*, Austin, TX, USA, Nov. 15–20, 2015.

[48] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Graphlab: A new framework for parallel machine learning," in *Proc. Annu. Conf. Uncertainty Artif. Intell. (UAI'10)*, Catalina Island, CA, USA, Jul. 2010, pp. 340–349.

[49] A. Kyrola, G. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a pc," in *Proc. 10th USENIX Symp. Oper. Syst. Des. Implement. (OSDI'12)*, Hollywood, CA, USA, Oct. 2012, pp. 31–46.

[50] A. Roy, I. Mihailovic, and W. Zwaenepoel, "X-stream: Edge-centric graph processing using streaming partitions," in *Proc. 24th ACM Symp. Oper. Syst. Princ. (SOSP'13)*, Farmington, PA, USA, Nov. 2013, pp. 472–488.

[51] Y. Zhou, L. Liu, K. Lee, C. Pu, and Q. Zhang, "Fast iterative graph computation with resource aware graph parallel abstractions," in *Proc. 24th ACM Symp. High Perform. Parallel Distrib. Comput. (HPDC'15)*, Portland, OR, USA, Jun. 15–19, 2015, pp. 179–190.

[52] Y. Zhou, L. Liu, K. Lee, and Q. Zhang, "Graphtwist: Fast iterative graph computation with two-tier optimizations," in *Proc. 41st Int. Conf. Very Large Data Bases (VLDB'15)*, Kohala Coast, HI, USA, Aug. 31/Sep. 4, 2015, pp. 1262–1273.

[53] L. Botton and Y. Bengio, "Convergence properties of the k-means algorithms," in *Advances in Neural Information Processing Systems 7 (NIPS'94)*, Denver, CO, USA, Dec. 1994, pp. 585–592.

[54] L. Kaufmann and P. Rousseeuw, "Clustering by means of medoids," *Statistical Data Analysis Based on the L1-Norm and Related Methods*, North-Holland, Amsterdam, The Netherlands, pp. 405–416, 1987.

[55] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 1998 Int. Conf. Knowl. Discov. Data Min. (KDD'98)*, New York, NY, USA, Aug. 1998, pp. 58–65.

[56] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[57] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *Proc. Int. Conf. Data Min. (ICDM'10)*, Sydney, Australia, Dec. 2010, pp. 911–916.

[58] S. Byan *et al.*, "Mercury: Host-side flash caching for the data center," in *Proc. 2012 IEEE 28th Symp. Mass Storage Syst. Technol. (MSST'12)*, Monterey, CA, USA, 2012, pp. 1–12.

[59] J. Guerra, H. Pucha, J. Glider, W. Belluomini, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," in *Proc. 9th USENIX Conf. File Storage Technol. (FAST'11)*, San Jose, CA, USA, Feb. 2011, p. 20.

Yang Zhou (M'13) is currently pursuing the Ph.D. degree at the College of Computing, Georgia Institute of Technology, Atlanta, GA, USA. He has authored more than a dozen papers in top journals and conferences in these fields. Grounded in real-world problems, he and his colleagues from diverse research fields, such as storage systems, web services, cloud computing, trust management and software engineering, have published brought about fruitful research results in these research areas. His research interests include big data analytics, data mining, databases, machine learning, parallel and distributed computing, security and privacy, information retrieval, with an emphasis on networked data.

Ling Liu (M'94–SM'06–F'15) is a Professor with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA. She directs the research programs in Distributed Data Intensive Systems Laboratory (DiSL), examining various aspects of large scale data intensive systems, including performance, availability, security, privacy and trust. She has authored over 300 international journal and conference articles Her research interests include big data systems and data analytics, cloud computing, database systems, distributed computing, Internet data management, and service oriented computing. In addition to services as the General Chair and PC Chair of numerous IEEE and ACM conferences in data engineering, very large databases and distributed computing fields, she has served on the editorial boards of over a dozen international journals. Currently, she is the Editor-in-Chief of the IEEE TRANSACTIONS ON SERVICE COMPUTING, and serves on the Editorial Board of international journals, including *ACM Transactions on Web* (TWEB), *ACM Transactions on Internet Technology* (TOIT). Her current research is primarily sponsored by NSF, IBM, and Intel. Dr. Liu is a recipient of the IEEE Computer Society Technical Achievement Award (2012) and a recipient of the Best Paper Award from a number of top venues, including ICDCS 2003, WWW 2004, 2005 Pat Goldberg Memorial Best Paper Award, the IEEE Cloud 2012, the IEEE ICWS 2013, and the IEEE/ACM CCGrid 2015.

Sangeetha Seshadri (S'07–M'09) received the B.E. degree (with Hons.) in computer science and the M.Sc. degree (with Hons.) in mathematics both from the Birla Institute of Technology and Science (BITS), Pilani, India, and the Ph.D. degree from Georgia Tech, Atlanta, GA, USA. She has been a Research Staff Member in the High Velocity Persistent Data Services Research Group, IBM Almaden Research Center, since October 2009. Her current work focuses on application-aware optimizations for storage technologies such as Flash and storage workloads such as real-time analytics. Her research interests include large scale storage systems.

Lawrence Chiu received the M.S. degree in computer engineering from the University of Southern California, Los Angeles, CA, USA, and the M.S. degree in technology commercialization from the University of Texas at Austin, Austin, TX, USA. He is Storage Research Manager and a Distinguished Engineer with the IBM Almaden Research Center. He cofounded both the IBM EasyTier and the IBM SAN Volume Controller products. EasyTier is a highly successful product for automated placement of data among different storage tiers in order to achieve optimal performance; the SAN Volume Controller is a leading storage virtualization engine, which has held the SPC-1 benchmark record for several years. In 2008, he led a research team in the US and U.K. to demonstrate a one-million-IOPS storage system using solid state disks. He is currently working on expanding solid state disk use cases in enterprise system and software.