Article

Not peer-reviewed version

Discovery of Cloud Applications from Logs

Ashot Harutyunyan *, Arnak Poghosyan *, Tigran Bunarjyan, Marine Harutyunyan, Andranik Haroyan, Lilit Harutyunyan, Nelson Baloian

Posted Date: 10 April 2024

doi: 10.20944/preprints202404.0701.v1

Keywords: automated cloud management; application discovert; log analytics; recommender systems

 doi:10.20944/preprints202404.0701.v1

Article

Discovery of Cloud Applications from Logs

Ashot Harutyunyan [1,*], Arnak Poghosyan [2,*], Tigran Bunarjyan [3], Marine Harutyunyan [4], Andranik Haroyan [4], Lilit Harutyunyan [4] and Nelson Baloian [5]

[1] Yerevan State University; harutyunyan.ashot@ysu.am
[2] Institute of Mathematics NAS RA; arnak@instmath.sci.am
[3] Technical University of Munich; tigran.bunarjyan@tum.de
[4] VMware by Broadcom; {marine.harutyunyan;andranik.haroyan;lilit.harutyunyan}@broadcom.com
[5] Department of Computer Science, University of Chile; nbaloian@dcc.uchile.cl
* Correspondence: harutyunyan.ashot@ysu.am (A.H.), arnak@instmath.sci.am (A.P.)

Abstract: Continuous discovery and update of applications or their boundaries running in cloud environments in an automatic way is a highly required function of modern data center operations solutions. Prior attempts to address this problem within various products or projects were/are applying rule-driven approaches or machine learning techniques on specific types of data − network traffic as well as property/configuration data of infrastructure objects, which all have their drawbacks in effectively identifying roles of those resources. The current proposal (ADLog) leverages log data of sources, which contain incomparably richer contextual information, and native constructs of VMware Aria Operations for Logs in terms of Event Types and their distributions to group those entities, potentially automatically enrich with indicative tags, and recommend for further management tasks and policies. Our methods discriminate not only diverse kinds of applications, but also their specific deployments thus providing hierarchical representation of the applications in time and topology. For several applications under Aria Ops management in our experimental test bed, we discover those in terms of similarity behavior of their components with a high accuracy. The validation of the proposal paves the path for an AI-driven solution in the cloud management scenarios.

Keywords: automated cloud management; application discovery; log analytics; event types; hierarchical and density-based clustering; recommender system; dimensionality reduction

1. Introduction

With the rise of cloud-native development practices [1,2], it has been critical to bring agility to legacy apps through application modernization. The primary objective in any modernization roadmap revolves around application assessment through discovery of application interactions, interdependencies as well as boundaries to help rationalize application portfolio and architecture and wide range of other modernization activities. Companies, such as Amazon, IBM, Microsoft, and Deloitte employ application discovery (AD) methodologies to enable enterprise customers with visibility into configuration, usage and behavior of workloads running on their infrastructure [3]. Moreover, the discovery and understanding of applications is critical in terms of creating relevant inventories of technical assets, improving maintenance and operations as well as significantly reducing risk in response to ongoing business needs. Cloud vendors tend to enhance their data center management portfolios with application-aware management capabilities. This introduces flexibility for customers to leverage tools and identify workloads [4] that necessarily make up an application and manage/troubleshoot their data centers with solid understanding of inter-dependencies, communication patterns and potential problems related to those applications. As modern multi-cloud environments advance rapidly with ever-growing complexities, it becomes challenging to accurately define highly dynamic application boundaries. As of this, the application definition itself can reflect different interpretations across different products. Most services usually leverage combination of workload naming conventions, workload tags, security tags and groups to establish application boundaries in their discovery engines. There are yet other approaches (e.g., AWS

 doi:10.20944/preprints202404.0701.v1

Application Discovery Service), that incorporate agent-based AD methodologies capturing system configuration, system performance, running processes, and details of the network connections between systems [5]. They necessarily gather and process information corresponding to server hostnames, IP addresses as well as resource allocation and utilization details related to VM inventory, configuration, and performance history such as CPU, memory, and disk usage data [6]. Although all these solutions in industry address variety of significant use cases in application-aware management frontiers, the accuracy and performance of the underlying models heavily rely on rich and available metadata to be in place. Overall, AD remains an important and complicated administration task under the emerging multi-cloud management paradigm, some vendors drive their vision to.

Our proposal (ADLog) takes a novel approach to AD problem from the log data of cloud sources and fully AI-driven perspectives. It is an attempt to employ unsupervised machine learning to modeling the log streams of cloud resources monitored by Aria Ops for Logs [7]. Log data contains contextually rich behavioral fingerprint of the underlying application. Therefore, we consider it as a more reliable and unbiased source for characterizing the nature of applications and discriminate those in terms of both their kinds and instances through the relevant workload patterns. Building on this intuition, our prototype solution verifies a hypothesis on whether event type (ET) meta-constructs by Log Insight (LI) [7] and their distributions are enough indicative to fingerprint applications across their dynamism in time. ETs were previously utilized in intelligent log analytics in identifying anomalies, changes, and sickness status of IT resources (e.g., [8]), as well as problem root cause analysis purposes, where explainable ML plays an important role in terms of adoptability of generated recommendations [9]. These are abstract groups of similar messages performing dimensionality reduction of the original log data. Users leverage ET representations in various tasks to interactively analyze log streams. An extra functionality (event trends) on top of ETs enables users to compare two-time intervals in the log stream and identify increasing or decreasing types of messages and locate sub-streams of interest.

We share results from our initial experimental work on a set of applications available to us at corporate internal environments. ADLog demonstrates highly accurate grouping outcomes of application entities and their deployment instances monitored and modeled with the above-mentioned ET distributions. We believe that based on this research, cloud management solutions can be furnished with a relevant log-driven discovery capability which is sufficiently reliable to be delivered as a standalone feature/recommender system for users to infer their map of app components, as well as enhance the existing AD solutions from a new angle.

2. Related Art

VMware enables its customers with variety of approaches to define applications within their products through cloud management and deployment tools (such as Configuration Management Databases, Aria for Automation [10], Aria Operations [11]), where customers can manually establish application definitions, specify application dependencies across different entities in datacenter and define application blueprints among resources in use. Additionally, customers can incorporate properties on their deployed entities as well as tags and naming conventions to identify specific resources that belong to subjective applications. Whereas all these approaches might serve well in specific use cases, they all expose drawbacks in terms of consistency in deviations from true application topologies over time. Current AD offerings include the Service and Application Discovery by Aria Ops [12], the network Flow-based Application Discovery (FBAD) [4] within Aria Ops for Networks [13]. The problem of AD can be tackles also from additional perspectives which utilize configuration properties of IT objects and their naming conventions.

From application monitoring, integration and management perspectives, Aria Ops enables Service Discovery (SD) for services running in each VM to build relationship or dependency between those from different VMs. With this, SD leverages VMware tools to discover some known services, but it limits the scope, in which the applications are only discovered within specific contexts of those services connected to each other. On the other hand, FBAD leverages network traffic flows from either the vSphere Distributed Switch [14] or NSX [15] and applies ML techniques (such as

 doi:10.20944/preprints202404.0701.v1

Disconnected Component, Outlier Detection) to discover application boundaries automatically [4]. Whereas the flow-based discovery approach groups application components based on their runtime behaviors, it is generally not capable of accurately capturing the components (e.g., VMs in "dev", "production", or "staging" environments) of a true application that are yet isolated in network. Unlike FBAD, object tags and property-based approaches drive discovery and ML-based inference of potential applications solely based on the configuration properties of compute entities and tags as an additional source for entity labeling from automated datacenter naming systems (not always available).

Even though these methods can serve several successful use cases in AD frontier, they heavily rely on the availability and propriety in contextual representations of data to train accurate and robust ML models. Due to the difference in methodologies, in assumptions and use cases of those as well as dissimilar specifics of environment conditions, the understanding of application topology across different end-use cases remains a state-of-the-art challenge for all these existing approaches. As a result, the AD itself exposes significant bias toward the source of information used in the discovery process.

Apart from application reconciliation and curation roadmap, it has become critical to serve variety of scenarios in different environments, by focusing on co-existence of different sources of discovery independently. With this, there is no prior art looking into the AD problem from log data perspectives of services, which contains contextually rich information about the underlying applications and infrastructures. This paper makes a preliminary validation of the viability of an approach which is based on log data characteristics, applies no expert knowledge, and is completely unsupervised.

3. Materials and Methods

Our approach in identifying similar groups of entities relies on hierarchical clustering of relevant log sources based on their ET (probability) distributions sampled across time axis with an aggregation interval. Then to evaluate the quality of such an approach, we need to compare the obtained groups of sources with the true structure of apps under investigation.

The full solution under our vision contains a recommender system running on trained ML models and advising users on discovered app maps, as well as dynamic changes occurring in those. Additionally, we can automatically reconstruct/prescribe application kinds while enriching detected groups with indicative messages behind dominant ETs using NLP or word clouds for user validation and continuous operator-in-the-loop improvement of the ML method.

3.1. Experimental Set Up for ADLog

To evaluate the ET-based grouping method, it is applied to a set of applications with ground truths coming from SD in Aria Ops (which is approved and labeled by developers) and from Engineering Services managing various internally deployed applications.

3.1.1. Monitored Applications

Ground truth applications monitored in Aria for Logs are listed in Table 1. Kinds of applications and status (whether it is a standalone instance although of the same kind, like ESX16 in the table) of entities within those applications are also indicated. The list includes vR Ops and vR LI apps with relevant services discovered by SD. This adapter or management pack-based discovery mechanism executes commands in the guest OS of the corresponding VMs and finds out which services are running, and which ones communicate through ports. The discovered applications can be both predefined and custom.

In Figure 1, a vR Ops application instance (of Table 1) is observed. This app consists of two VM nodes. There are multiple services in this app: "vROps Analytics", "vROps CaSa", etc. Those services are hosted by different VMs, however, they are part of the same "vRealize Operations" application. Another application discovered by SD is "vRLI" (Figure 2). There are three nodes in the application

 doi:10.20944/preprints202404.0701.v1

and each of them are connected to the other node through services. Both applications definitely exist in the monitored environment. The rest of applications in Table 1 are confirmed by the engineering teams dealing with internally deployed apps under monitoring of Aria Ops for Logs. For instance, F5 load balancer has two distinct instances. Analogously, the host esx16 is not a part of the same service application consisting of the rest of 9 hosts, but a different service running with an NSX.

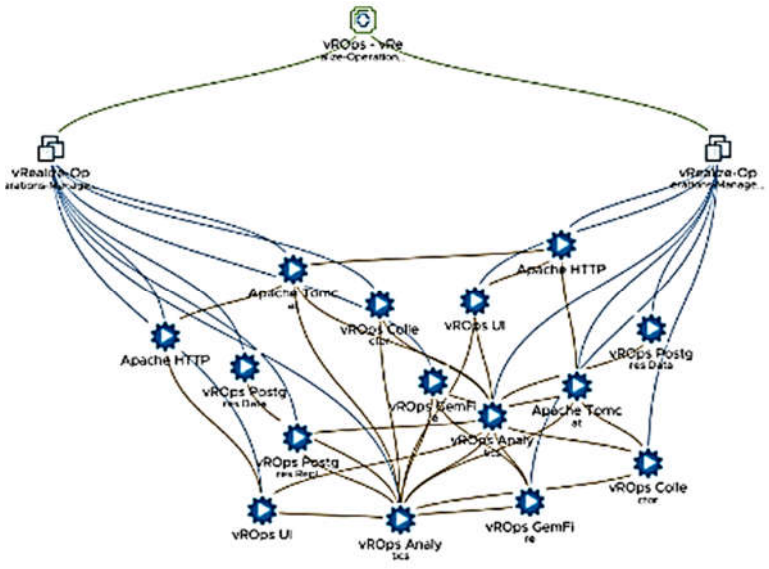


Figure 1. SD-discovered vR Ops app with two nodes.

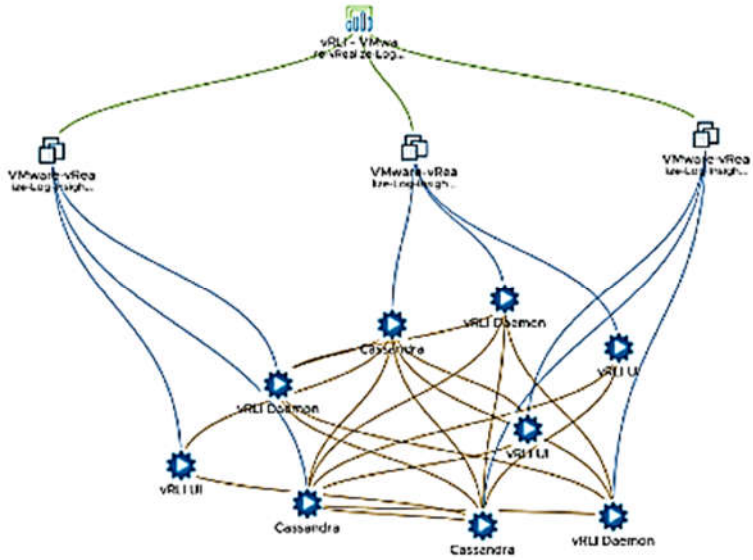


Figure 2. SD-discovered vR Log Insight app with three nodes.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 April 2024 doi:10.20944/preprints202404.0701.v1

Table 1. Ground truth app kinds and their structure.

Log Sources	Application Kind	Deployment
10.27.74.245	F5 Load Balancer v13	Standalone
10.27.74.233	F5 Load Balancer v12	Standalone
10.27.82.46	vRLI	Part of one
10.27.82.45	vRLI	
10.27.82.43	vRLI	
evn1-hs1-a0716.eng.vmware.com	ESXi (+ vSAN)	Under the same vSphere
evn1-hs1-a0719.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0717.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0720.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0722.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0723.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0724.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0721.eng.vmware.com	ESXi (+ vSAN)	
evn1-hs1-a0718.eng.vmware.com	ESXi (+ vSAN)	
sc2-05-r19esx16.oc.vmware.com	ESXi (+ NSX)	Standalone
10.27.82.51	vROPs	Part of one
10.27.82.52	vROPs	
10.27.74.219	Microsoft Internet Information Server	Standalone
10.27.74.218	PaloAltoNetwork	Standalone

An examplary ET distribution charting over time axis is depicted for the load balancer application F5 in Figure 3. For each time interval, the relative presence of ETs can be converted to relevant probability distributions (as discribed in prior art literature, as well as next subsection). Our ML analysis deals with those distributions in identiying behavioral similarity of log sources to potentially make an application kind or its particular deployment in case of an hierarchical discrimenation.

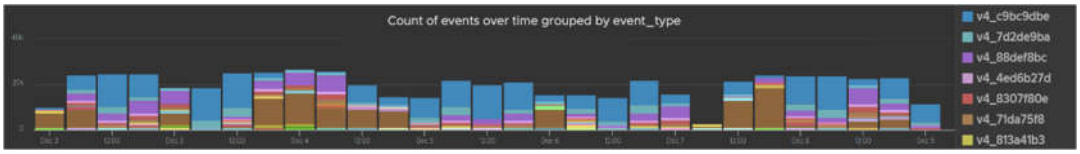


Figure 3. Visual representation of ETs for F5 load balancer charted with 6-hour intervals over the period Dec 2-7, 2022.

3.2. Feature Engineering

All log sources of apps in Table 1 were monitored by an Aria Ops for Logs for a week period. With our prototype scripts we extracted counts of distinct ETs observed in each 4-hour interval for each of the sources under consideration, thus generating a data store of samples to be processed. Then converting those values into relative frequencies or probabilities of ETs, we continued to work with resultant distributions (it means probability elements are summing up to 1 for each time interval). It is important to note, that for the sake of an accurate ML treatment, this feature engineering technique builds the relevant probability vectors on the entire set of distinct ETs observed from all sources thus introducing a sparsity into the data set subject to the hierarchical clustering below. Overall, 607 distributions (with dimensionality=5605) feature vectors, representing behavioral samples from the six application kinds and their sources are collected for over the one-week monitoring window.

Here are some interesting patterns and insights from the descriptive analysis of the ET data. Figure 4 demonstrates a structure in observed ET counts for the same sort of log sources. It seems that even based on this total count criterion app kinds can be differentiated/filtered out in a given time slot.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 April 2024 doi:10.20944/preprints202404.0701.v1

Figure 5 displays a discernable structure of probability feature vectors (with probabilities higher than 0.005) for two 4-hour intervals of distinct instances of the same load balancer F5.

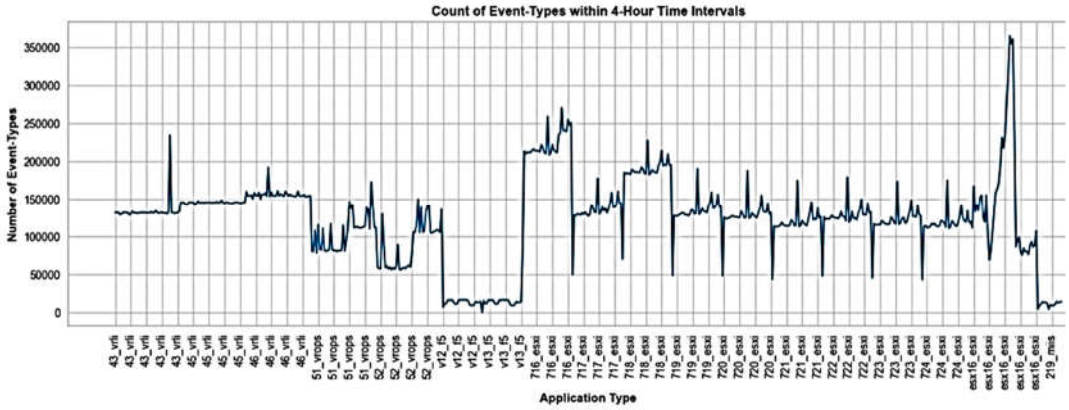


Figure 4. Number of ETs within 4-hour interval for all sources.

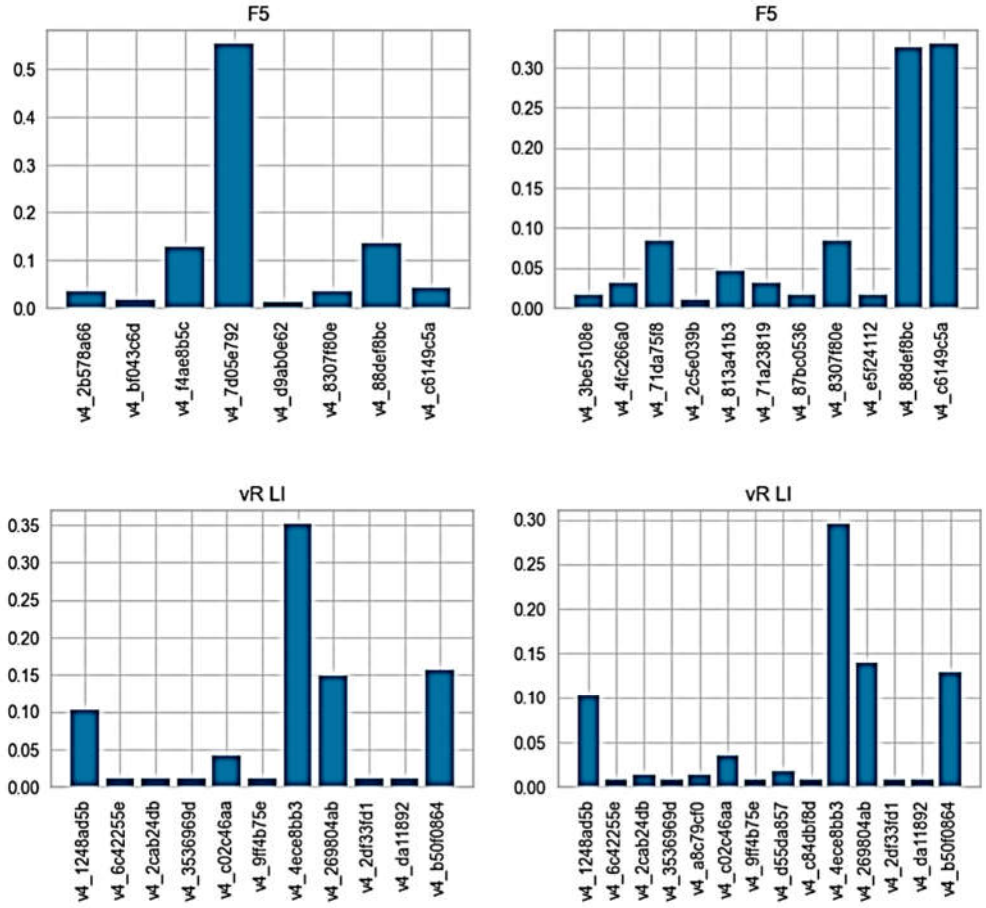


Figure 5. Two ET distributions (top probabilities only) from F5 and vR LI sources of the same deployment.

3.3. ADLog Using Hierarchical Clustering of Event Type Distributions

Features constructed based on ETs occurrence described in section 4.2, are fed into the agglomerative clustering with Jaccard distance and average linkage. Since we don't know how many groups of entities to expect in the cloud environment, Silhouette and Calinski-Harabasz scorings serve a guidance for inferring the best number of clusters. Figures 6 and 7 display the corresponding results, respectively, detecting the number of app kinds in our experiment, where in Figure 5 the

 doi:10.20944/preprints202404.0701.v1

elbow method is used. In general, we suggest experimenting with different clustering scores and making the final decision based on the summary of the results.

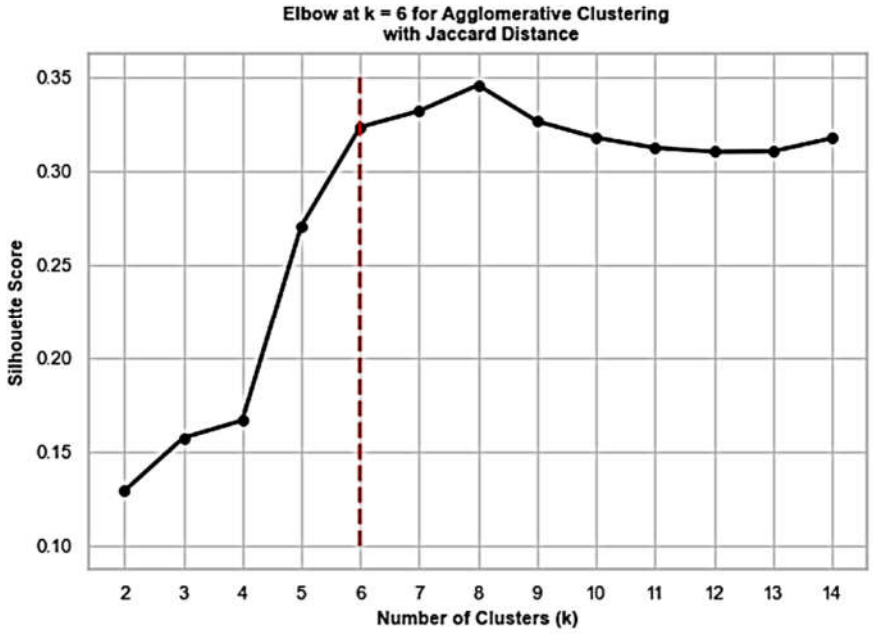


Figure 6. Silhouette scoring for the agglomerative clustering.

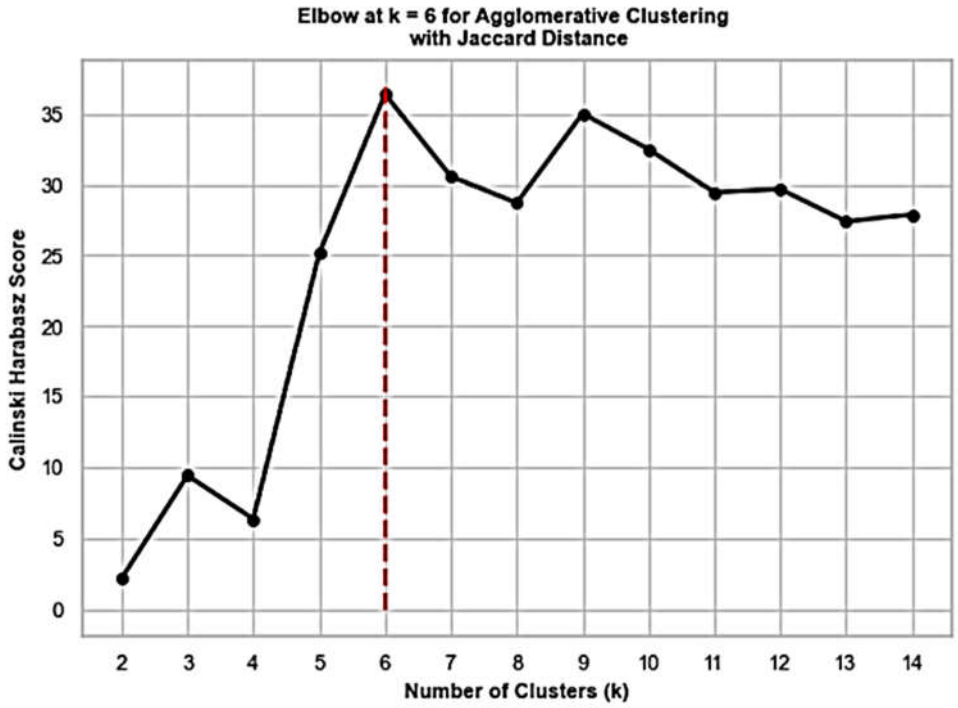


Figure 7. Calinski-Harabasz scoring for the agglomerative clustering.

The dendrogram in Figure 8 illustrates how the various sorts of apps (indicated in differently colored trees) in our analysis are discriminated based on the above-mentioned scoring results. The most sizeable cluster (yellow) represents ESXi-kind application with comparably larger number of sampled ET distributions from related host entities.

Since within ADLog the next objective is to be able to also distinguish app instances of the same kind subject to their workload patterns (reflected in ET distributions), the solution performs further steps in interpreting the high dimensional feature space we deal with into 2-dimensional

 doi:10.20944/preprints202404.0701.v1

representation for more granular observability and an extra tooling in recommending the cloud application map. In that context, Figure 8 reflects the original feature space through t-SNE (t-distributed Stochastic Neighbor Embedding) methodology [16].

Interestingly, Silhouette scoring (Figure 10) for HDBSCAN [17] clustering using t-SNE projections results in six groups again, where we chose the density-based way of grouping as more intuitive in view of projections plot in Figure 9.

We apply HDBSCAN clustering to t-SNE feature space shown in Figure 9 for different values of the number of clusters and calculate the corresponding silhouette scores (see Figure 10).

The graph of Figure 10 suggests that the maximum value of silhouette score is attained at $k = 2$. However, our choice stands on $k = 6$ that corresponds to the next maximum. The reason for this choice is the visual confirmation obtained from Figure 9. Application of HDBSCAN with 6 clusters implies labeled t-SNE representation of Figure 11 with clearly separated constellations of observations sampled from the different kind of app entities/sources.

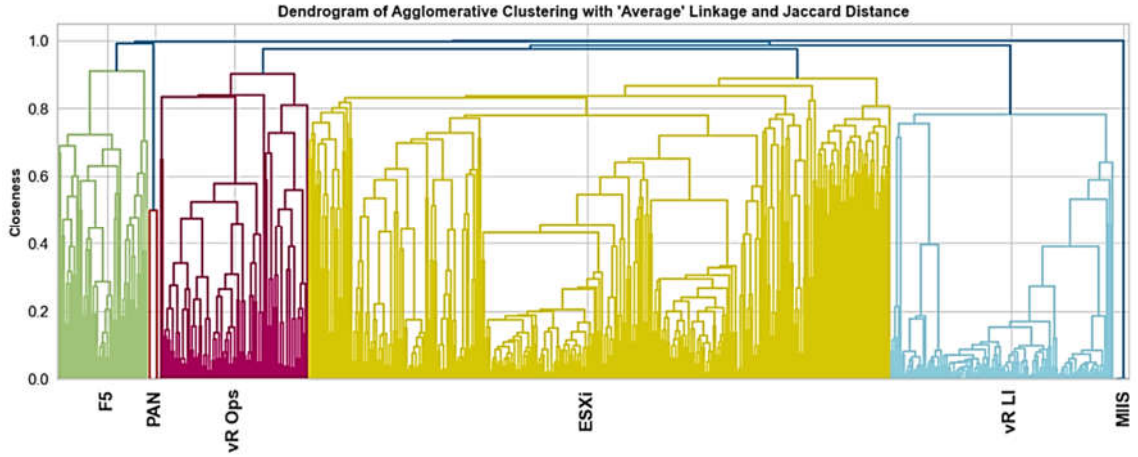
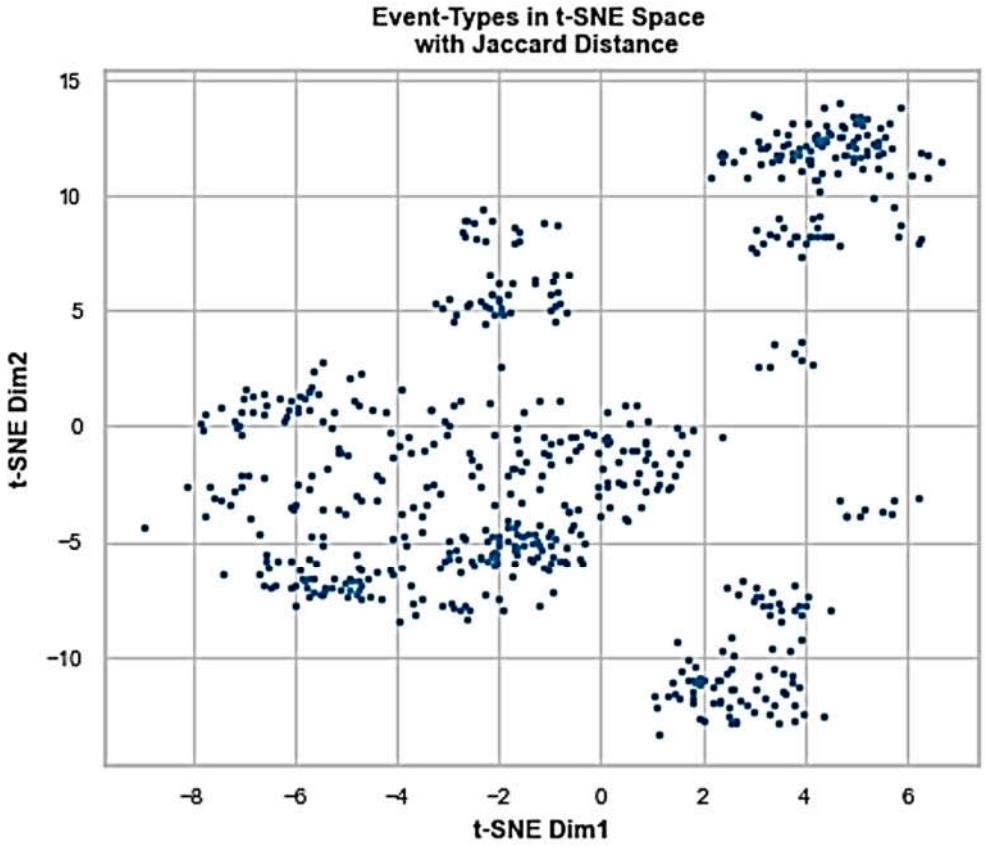


Figure 8. The dendrogram of agglomerative clustering with average linkage and Jaccard distance.



 doi:10.20944/preprints202404.0701.v1

Figure 9. t-SNE representation of features of log sources.

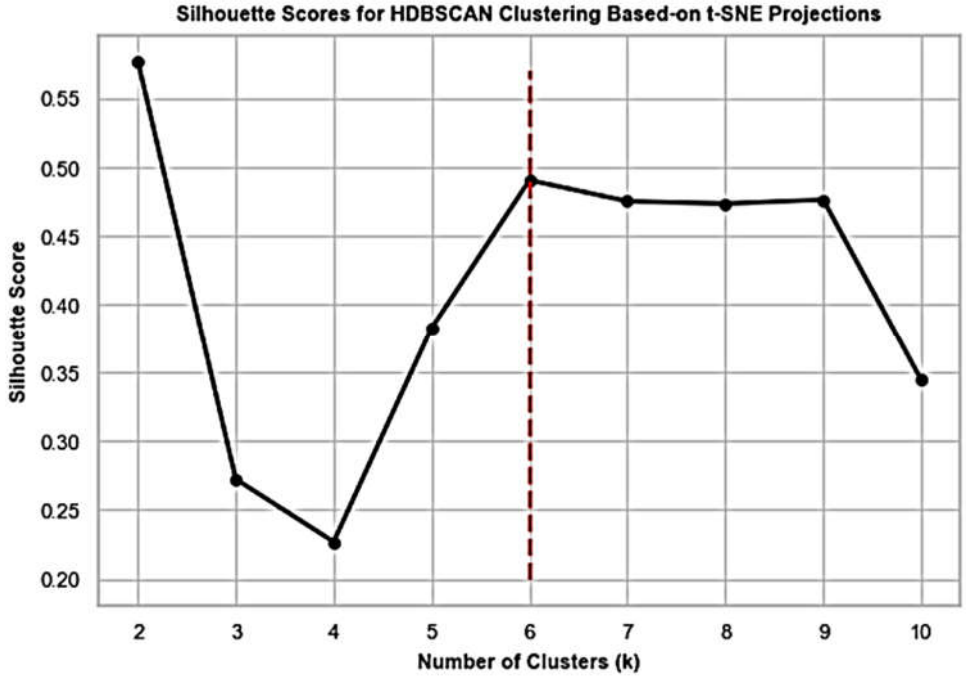


Figure 10. Silhouette scoring of HDBSCAN based on t-SNE projections.

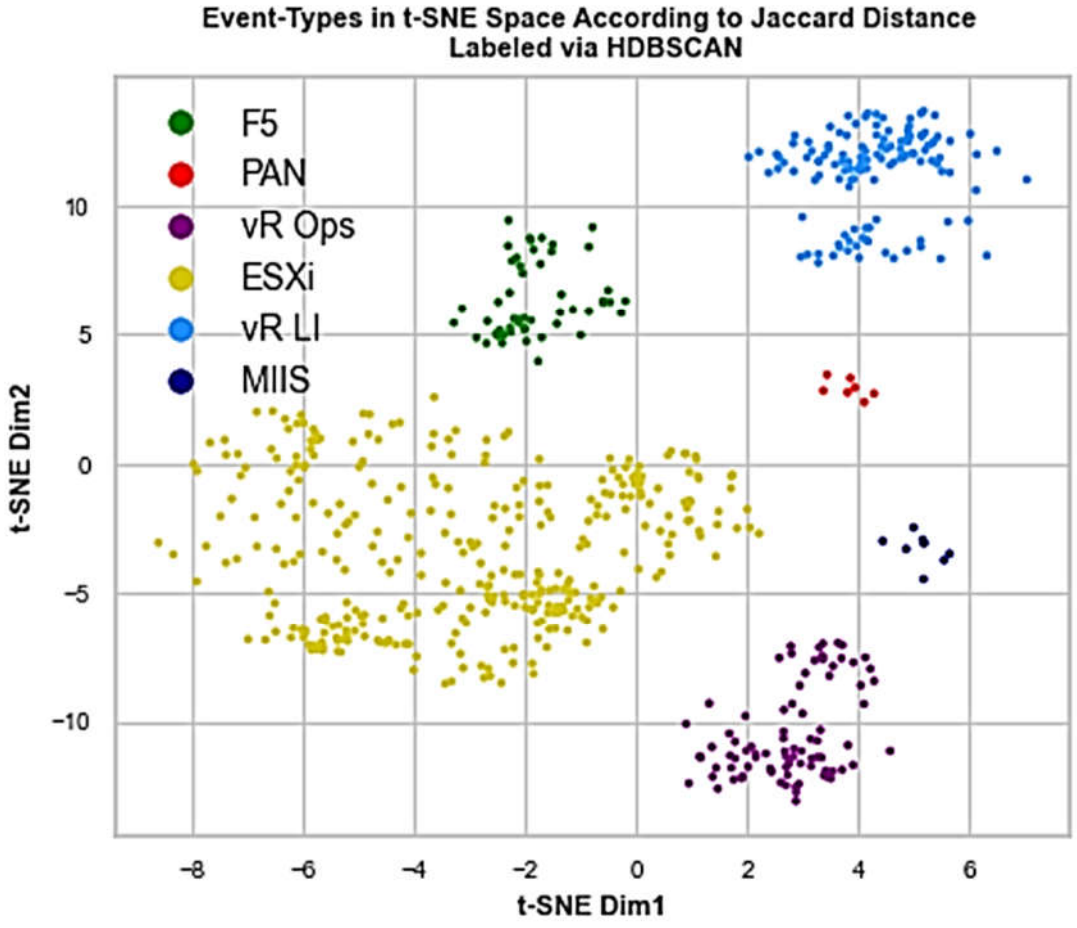


Figure 11. Features in t-SNE space according to Jaccard distance, labeled using HDBSCAN.

 doi:10.20944/preprints202404.0701.v1

Zooming into the ESXi constellation and taking it as an input to the agglomerative clustering with the related t-SNE features, we infer that the second stage grouping for this data set can be performed as the Silhouette scoring in Figure 12 indicates. We highlight this effect in Figure 13, where the samples coming from the source esx16 make a separable and tighter group in the t-SNE plot of features using preserved L_1-distance. Therefore, ADLog leads to a prediction about a distinct app instance which is compliant with the ground truth in Table 1.

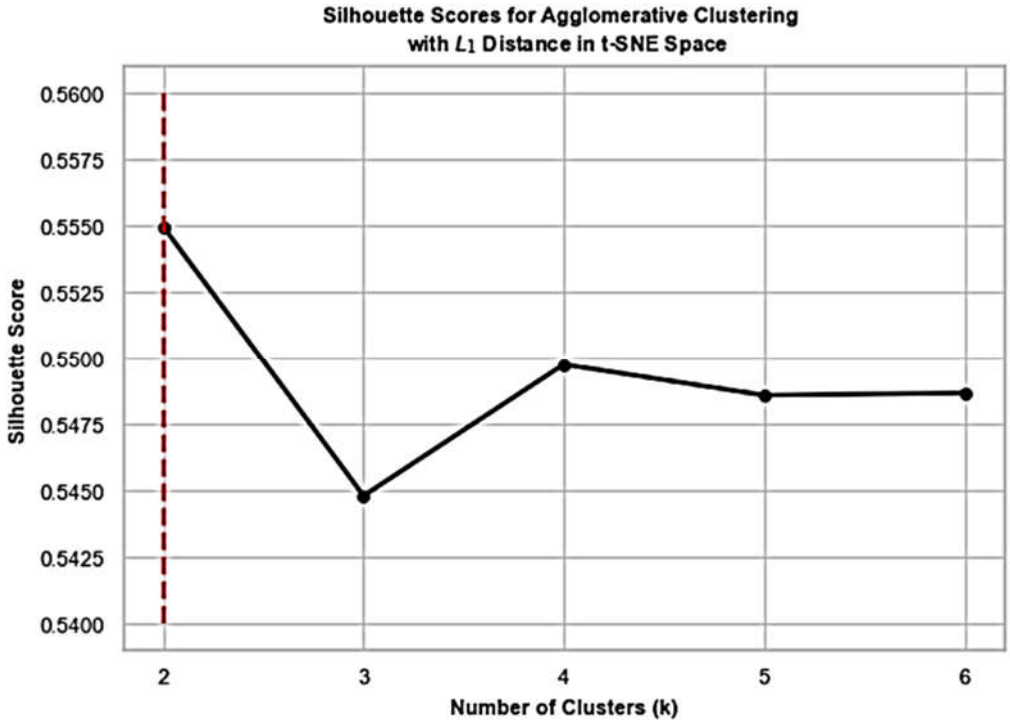


Figure 12. Silhouette scoring for agglomerative clustering in t-SNE space.

 doi:10.20944/preprints202404.0701.v1

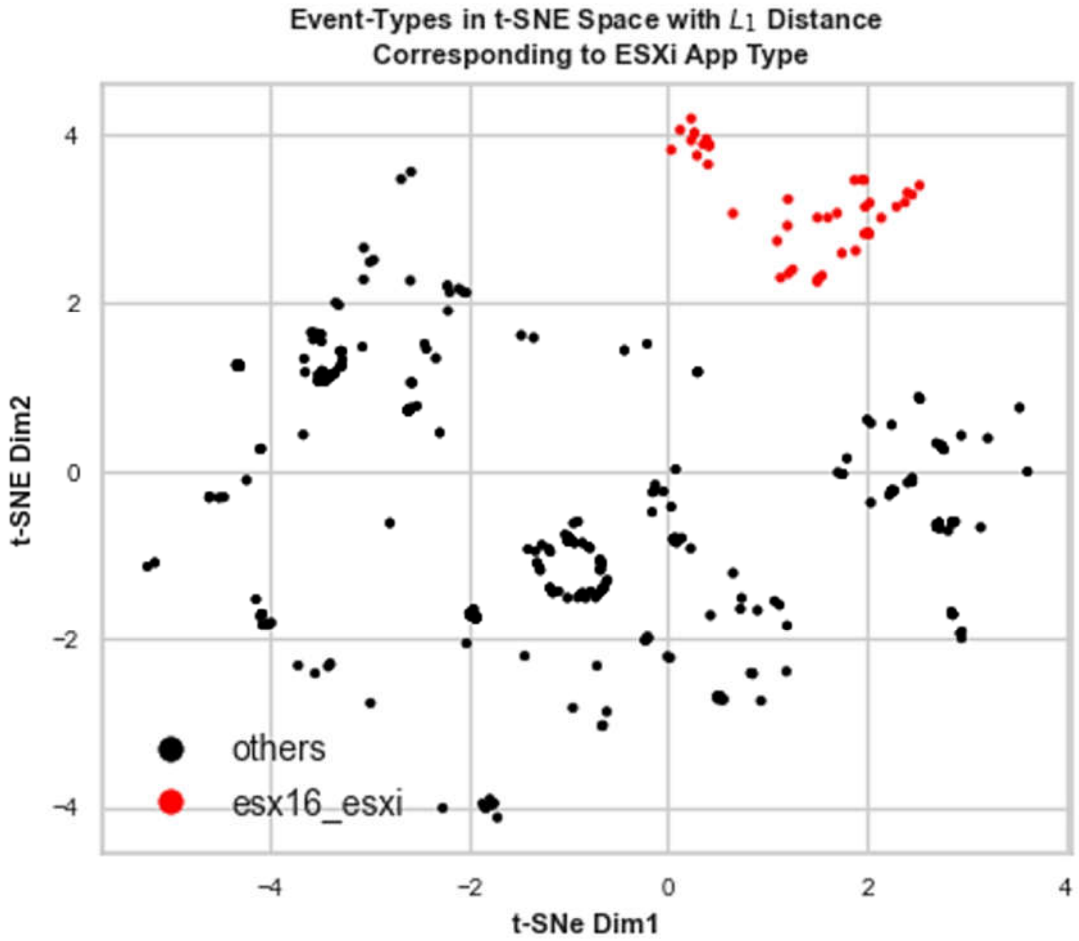


Figure 13. Features in t-SNE space with L_1 distance for ESXi app kind.

4. Evaluation of Results

The insights discussed in the previous subsection confirm the ability of ADLog to perform application categorization using ET distributions by Aria Ops for Logs. This prototype solution demonstrated a high discrimination power and accurately identified the app kinds in ground truth Table 1 with 100% accuracy.

As to the identification of standalone instances of the same kind, only in case of F5 load balancer with two consecutive versions v12 and v13, it was not possible to effectively discriminate those (see Figure 14). Hypothetically, ADLog misses this fragment because of the very nature of F5 and its behavioral footprint collected, needing more distributions to be sampled reflecting various levels of workload stress. In general, Jaccard distance is a suitable measure for identifying the app kinds, while more sensitive measures like L_1 are better for the second-layer and more granular clustering of log sources.

Within a log analytics service, ET distributions can provide also an explainability for discovered applications as footprint structures. However, in general, they might represent rather big vectors of probabilities not suitable for a simple interpretability purposes. At the same time, they are baseline structures of applications which can be utilized not only for the discovery task but also for identifying atypical performance behaviors (abnormalities or anomalies) when diverging from dominant distributions (see [8] for relevant concepts and methods). In this regard, for an anomaly detection task of an application within a supervised learning set up, where training data given by ET distributions and their labels are available, a rule induction and verification approach based on Dempster-Shafer theory (DST) of evidence [18] introduced in [9] is of special value. It can validate simple rules defined on probability degrees of individual ETs and their combinations that lead to

 doi:10.20944/preprints202404.0701.v1

anomalous application states, thus reducing the complex interpretability in characterizing applications and their performance statuses.

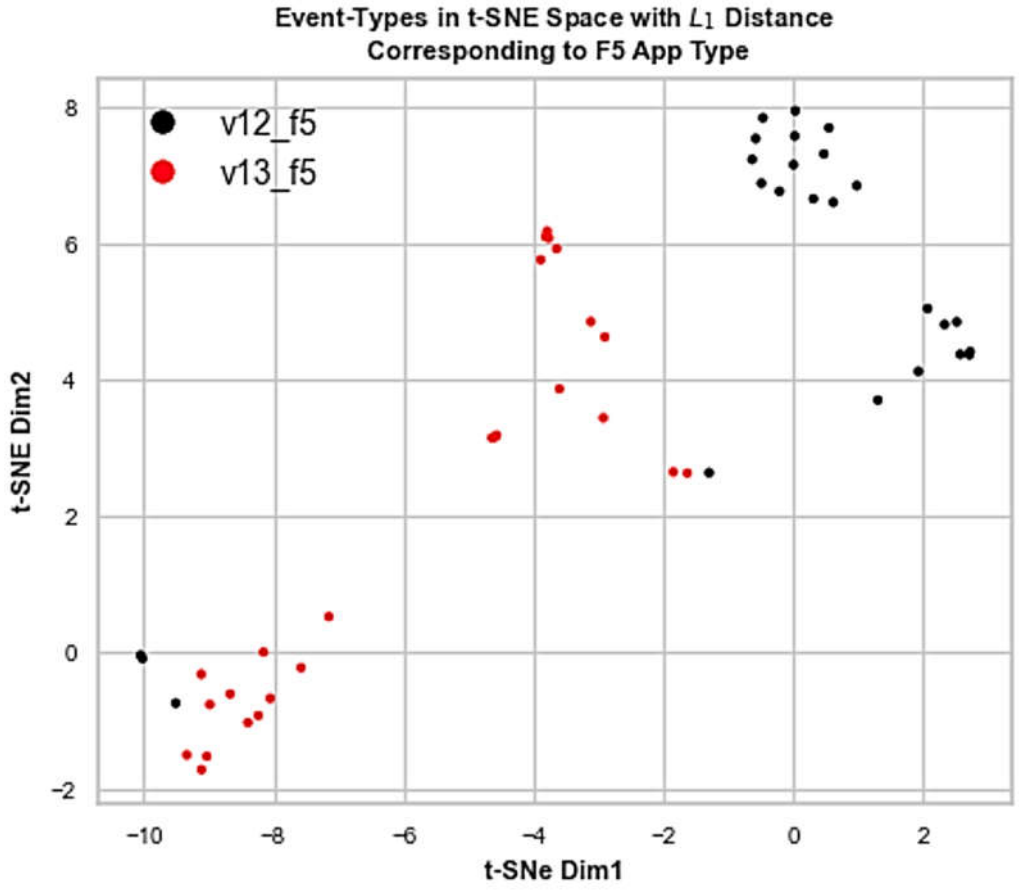


Figure 14. Not completely separable versions of F5 app kind in t-SNE space.

5. Discussion and Conclusions

Our initial belief is that the AD from logs approach remains a standalone way for contextually relevant and accurate identification of cloud resources, while also guiding other techniques to improve their recommendations and function as an ensemble method. At the same time, it is important to note that especially in the context of multi-cloud and native cloud management scenarios, ADLog might be a stronger choice for adoption with higher degree of observability power.

To enhance this work, we plan to conduct extensive experimental validation and comparison with various techniques based on available log data sets which we did not possess while working on this idea. Explainability aspect in application discovery and performance analysis is another important direction of research to pursue in our future studies.

An extended ADLog prototype will also include a fully developed recommender system for inferring relevant application kinds and indicative tags from messages behind ETs using NLP.

6. Patents

The study is supported by a filed US patent.

Author Contributions: Conceptualization, A. Harutyunyan and A.P.; methodology, A. Harutyunyan and A.P.; software, A.H., M.H., and L.H.; validation, A.H. and M.H.; formal analysis, A.P. and T.B.; investigation, A.P., T.B., and L.H.; resources, X.X.; data curation, A.H., M.H, and L.H.; writing—original draft preparation, A. Harutyunyan and A.P.; writing—review and editing, T.B. and N.B.; visualization, A.P. and L.H.; supervision, A. Harutyunyan.; project administration, A. Harutyunyan.; funding acquisition, N.B. All authors have read and agreed to the published version of the manuscript.

 doi:10.20944/preprints202404.0701.v1

Funding: The research was supported by ADVANCE Research Grants from the Foundation for Armenian Science and Technology.

Data Availability Statement: The data analyzed in this study are available on request from the corresponding author. The data are not publicly available due to their proprietary and business-sensitive nature.

Acknowledgments: Authors are thankful to the anonymous reviewers for their constructive feedback.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. What is application modernization? www.vmware.com/topics/glossary/content/application-modernization.html (accessed on 04/04/2024).
2. Microsoft Azure: What is application modernization? www.azure.microsoft.com/en-us/solutions/application-modernization/ (accessed on 04/04/2024).
3. Application Modernization: Application Discovery Services, www.deloitte.com/us/en/pages/technology/solutions/application-discovery-tool.html (accessed on 04/04/2024).
4. Smit, M. Using machine learning to discover applications, www.blogs.vmware.com/management/2020/05/using-machine-learning-to-discover-applications.html (accessed on 04/04/2024).
5. AWS Application Discovery Service, https://www.aws.amazon.com/application-discovery/faqs/ (accessed on 04/04/2024).
6. Introduction to AWS AD Service and its use cases, www.projectpro.io/recipes/introduction-aws-application-discovery-service-and-its-use-cases (accessed on 04/04/2024).
7. VMware Aria Operations for Logs, www.vmware.com/products/vrealize-log-insight.html (accessed on 04/04/2024).
8. Harutyunyan, A.; Poghosyan, A.; Grigoryan, N.; Kushmerick, N.; Beybutyan, H. Identifying changed or sick resources from logs, In Proceedings of 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Trento, Italy, pp. 86-91, Sept. 3-7, 2018. doi: 10.1109/FAS-W.2018.00030.
9. Poghosyan, A.; Harutyunyan, A.; Davtyan, E.; Petrosyan, K.; Baloian, N. A Study on automated problem troubleshooting in cloud environments with rule induction and verification. *Appl. Sci.* 2024, *14*, 1047. https://doi.org/10.3390/app14031047.
10. Working with blueprints/cloud templates, https://developer.vmware.com/docs/19299/GUID-976A01AB-CEEA-4A9A-B485-858012FA0DE1.html (accessed on 04/04/2024).
11. Service and Application Discovery, https://www.blogs.vmware.com/management/2020/12/an-overview-of-application-monitoring-with-vrealize-operations.html (accessed on 04/04/2024).
12. An overview of application monitoring with vRealize Operations, www.blogs.vmware.com/management/2020/12/an-overview-of-application-monitoring-with-vrealize-operations.html (accessed on 04/04/2024).
13. VMware Aira Operations for Networks, https://www.vmware.com/products/aria-operations-for-networks.html (accessed on 04/04/2024).
14. vSphere Distributed Switch, https://www.vmware.com/products/vsphere/distributed-switch.html (accessed on 04/04/2024).
15. VMware NSX, https://www.vmware.com/products/nsx.html (accessed on 04/04/2024).
16. van der Maaten, L.J.P.; Hinton, G.E. Visualizing data using t-SNE, *Journal of Machine Learning Research*, vol.9, pp. 2579-2605, 2008.
17. Campello, R.J.G.B.; Moulavi, D.; Sander, J. Density-based clustering based on hierarchical density estimates, *Lecture Notes in Computer Science, vol. 7819, Springer, Berlin, Heidelberg, 2013.*
18. Shafer, G. A Mathematical Theory of Evidence; Princeton University Press: Princeton, NJ, USA, 1976.