Intelligent Troubleshooting in Data Centers with Mining Evidence of Performance Problems

Ashot Harutyunyan, Naira Grigoryan, Arnak Poghosyan, Sunny Dua, Hovhannes Antonyan, Karen Aghajanyan, and Bonnie Zhang

Cloud Management BU, VMware

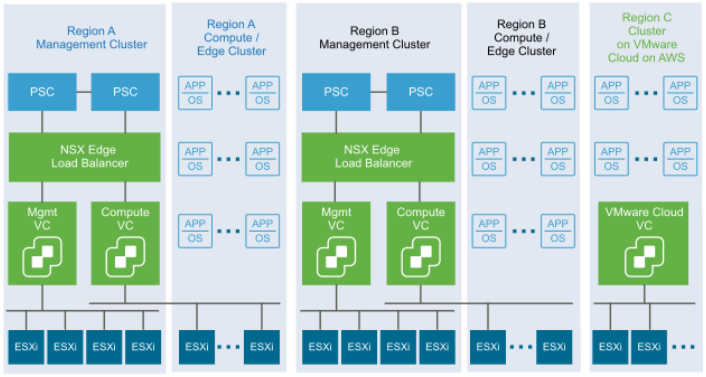
CODASSCA 2020

AUA, Yerevan, Armenia
September 14-17, 2020

vmware

Vision of Self-Driving Data Centers

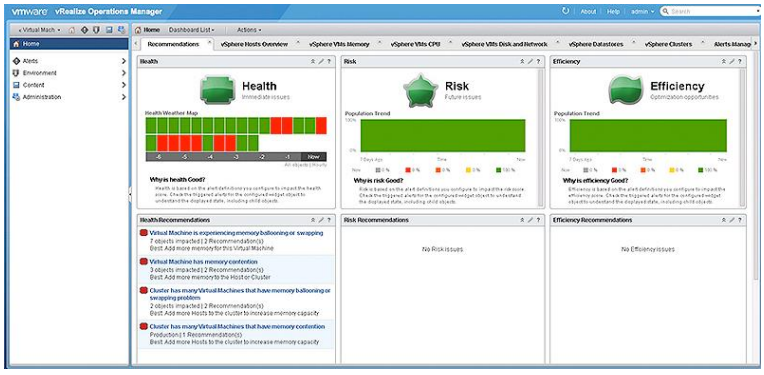
- Cloud infrastructures and applications are highly complex and dynamic systems



Public, Private, and Hybrid clouds are the main models of cloud computing environments where most of the modern software applications are serviced. They are highly volatile, complex, and sophisticated systems with challenging management problems.

Vision of Self-Driving Data Centers

- For a full visibility into those systems and their reliable management
  - real-time monitoring of diverse parameters are vital
  - industry trending to apply AI approaches for making decisions instead of human operators



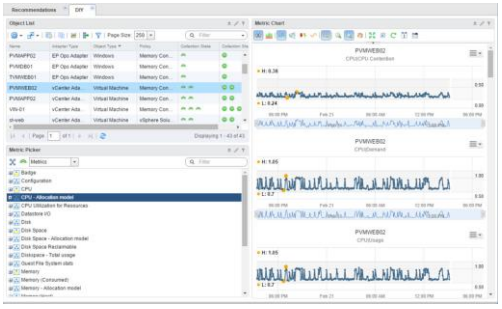
vmware

Such a management needs to be enough insightful, desirably proactive to maintain those systems healthy, optimal, with minimum impact on business delivery/end-users.

This inevitable leads to AI solutions leveraging huge amount of measured data from the entire infrastructure stack and applications processes.

Root Cause Analysis (RCA) is Challenging

- Hard to identify the problem cause(s) within a large volume of parameters/data
  - Compute/Network/Storage metrics (time series)



vmware

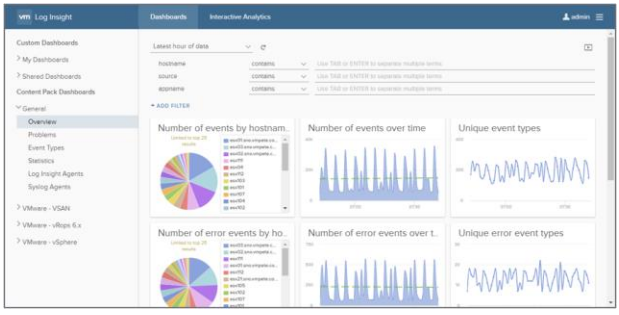
Causes of performance issues within modern cloud infrastructures are hard to identify. Because of high level of complexity of those systems. It is especially complicated to diagnose a service or infrastructure degradation of an unknown nature, when fault conditions (alerts) do not indicate the source of the problem or are only effects.

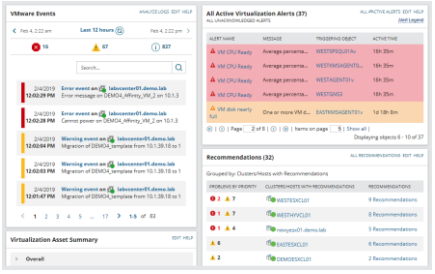
In such a situation, the data center administration is intuitively looking for changes in the system that might reveal the causative factors. This requires costly investigations and results in business-critical losses.

Cloud management vendors are building visions around AI Ops-enabled automation of the entire workflow of root cause analysis and troubleshooting.

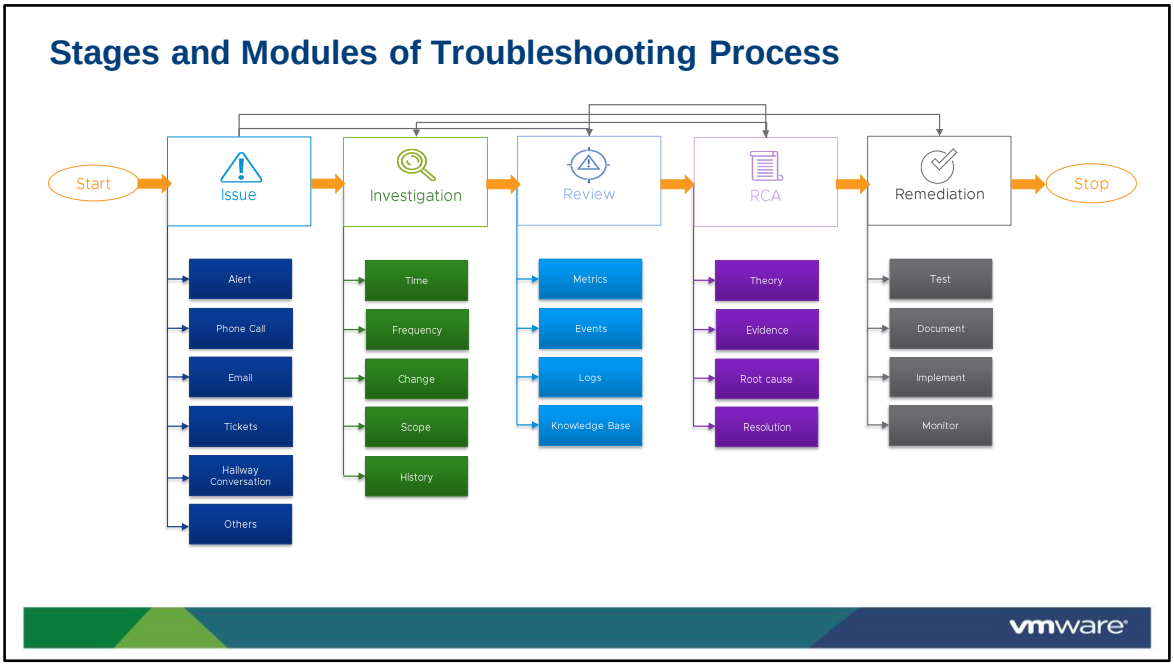
Root Cause Analysis (RCA) is Challenging

- Large volume of heterogeneous data reflecting cloud environments
  - Events
  - Logs
  - Configuration properties

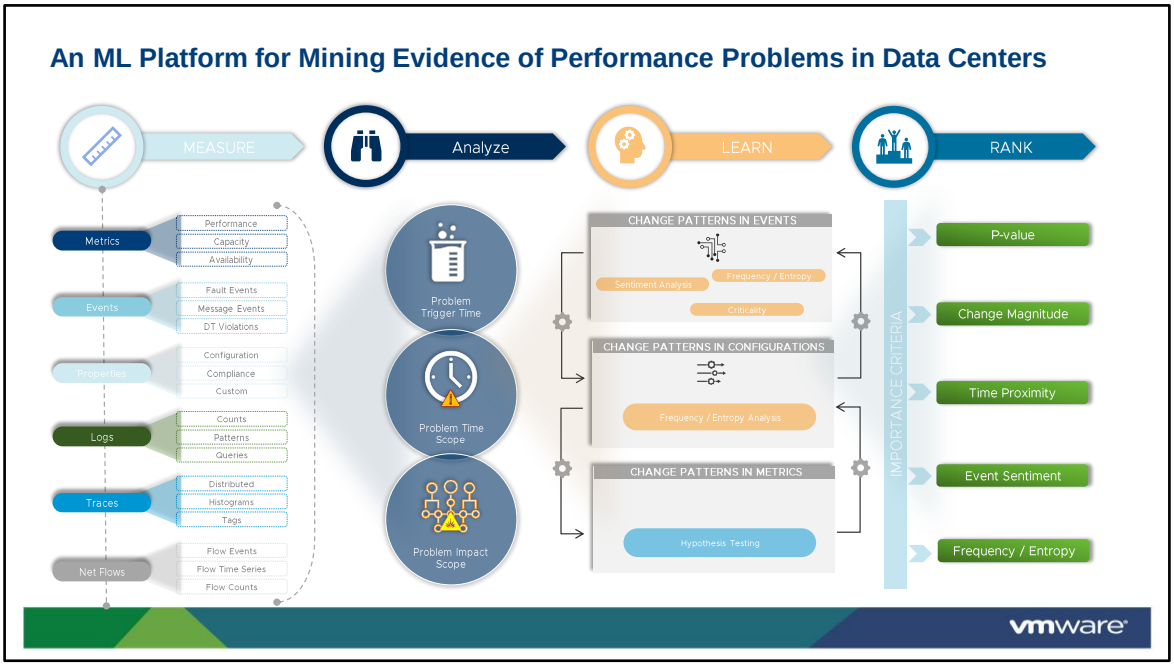




All kinds of data representing the underlying system are useful (capturing different aspects/dimensions of the cloud flows). The problem is how to make sense of such heterogeneous sources of information.



Signals about a problem come from various channels. They refer a service degradation or non-optimal performance which can originate both from the infrastructure and application layers of the cloud system. When it occurs, the user goes through a typical process of troubleshooting consisting of several stages. This process is subject to full automation within an intelligent troubleshooting. To adequately approach the problem, different information sources (obtained from monitoring of various aspects of the data center deployment – metrics, logs, properties, events, application traces, net flows, etc.) need to be combined within an intelligent analysis in an automatic manner. It means that all cloud monitoring solutions/products could bring their insights into such an analysis within an integrated cloud management platform.



We propose a solution towards such a vision which is based on hypothesis testing and machine learning approaches for automatic mining "important changes" of various kinds in behavior of data center objects across time and infrastructure topology. Those are the most relevant evidence patterns expected to explain the performance issue. Our current implementation which is integrated into vRealize Operations runs on the available three sorts of monitoring data – metrics, properties, and events. However, the full vision is to extensively include more observability provided by other cloud management tools vertically scaled to capture the depth of a specific dimension of the data center administration. The implemented module produces lists of recommended patterns across those three dimensions rank ordered subject to different criteria for each, such as confidence (p-value) provided by hypothesis testing and magnitude of change in the metric data, event's sentiment score or abnormality degree, unexpectedness/entropy of property variations, etc. We describe the main analytical concepts behind the solution and demonstrate its validation in an application troubleshooting scenario.

This demonstrates our AI Ops vision consisting of four main layers:
- measuring and selecting data for analysis,
- discovery of problem signals in time and topology scopes,
- learning importance of patterns,
- ranking those by various criteria.

Evidence Mining Approaches

- **Change** (point) **detection** across all sorts of data
  - for objects in Time and Topology Scopes
- Implementation includes
  - time series data
  - events
  - properties
- Hypothesis Testing (HT) is applied with
  - quantifying strength of evidence by
    - p-value
    - change magnitude

For time series, different tests can be applied, parametric/non-parametric, for instance, Pettitt's

$$U_{t,T} = \sum_{i=1}^{t} \sum_{j=t+1}^{T} D_{ij}$$

$$D_{ij} = sgn(x_i - x_j) = \begin{cases} 1 & x_i < x_j \\ 0 & x_i = x_j \\ -1 & x_i > x_j \end{cases}$$

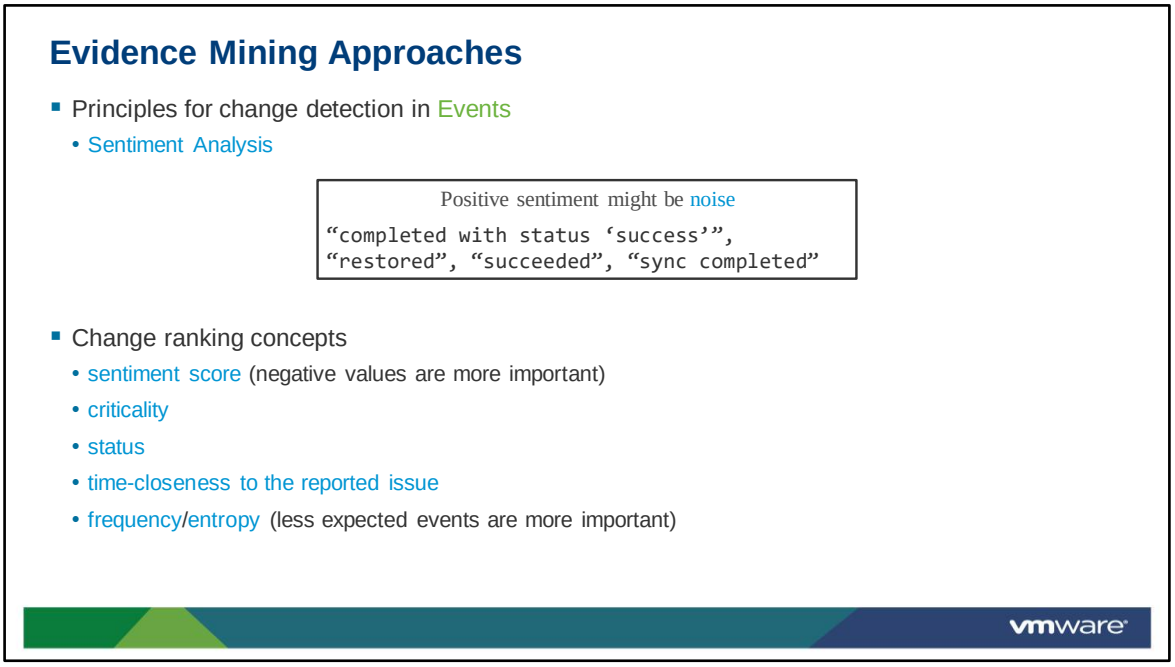
$$K_T = \max_{1 \le t < T} |U_{t,T}|$$

vmware

For the identified problem coverage zone (both learned or adjusted by the user), a change point detection algorithm is applied to all metrics of objects within the related time and topology scopes. We assume that when the user is troubleshooting an issue that is still active, spiky behaviors in the metrics are not very much interesting to the user. Instead, changes indicating distribution shifts (or any of its attributes like mean, variance, median, etc.) in the time series data are important.

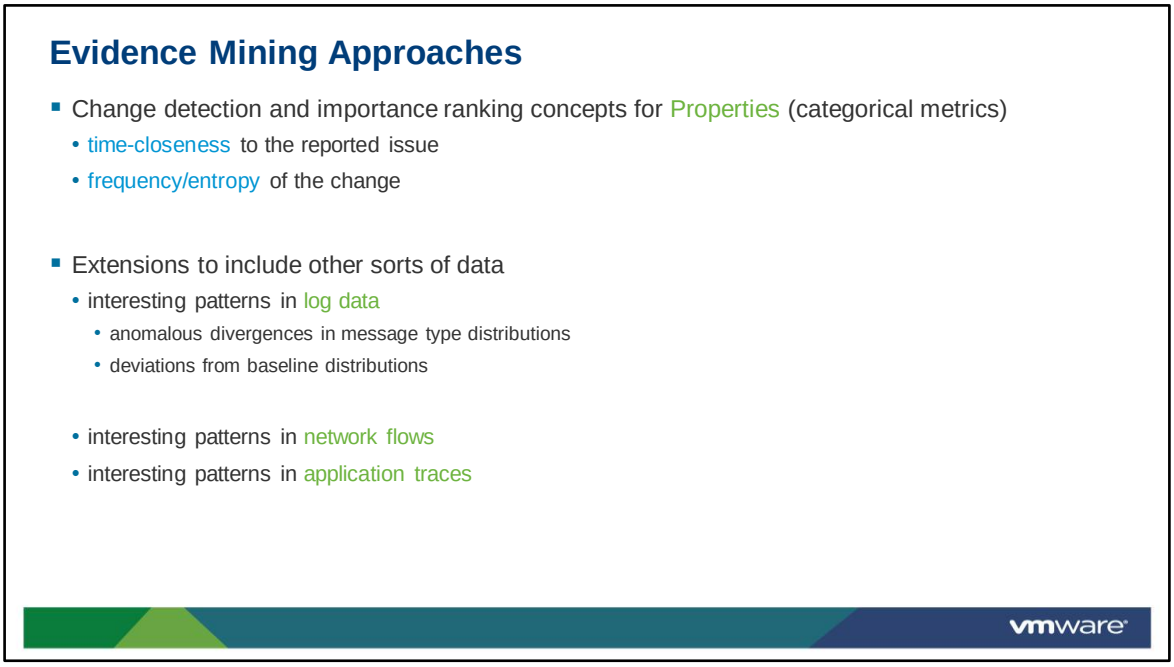
- we exclude those metrics that are "accumulative", trendy (e.g. uptime metrics) by checking whether change was declared on all steps when we perform HT;
- for each of the changed/anomalous metrics, we compute magnitude of the change within the time scope, measured by difference of medians between the left and right windows of the change point, normalized over the range of the whole data. This way we filter out those metrics which experienced a distribution change but not a significant shift in the data range.

A non-parametric test, reasonable in terms of implementation feasibility with randomization for short time series data and p-value estimation, is Permutation Test.



Querying all events that are active during the troubleshooting time frame. All types of events can be considered (Faults, Change Events, Notifications, Dynamic Threshold violations, etc.) except, for instance, those which are subject to symptoms included in alert definitions (for example, Hard Threshold violations). This is because in case of an "unknown" issue, the alerts appeared in the system are not self-explaining the cause of the problem. We apply sentiment analysis to narrow down the space of potential evidence patterns. The algorithm excludes the events with highly positive sentiments (from a predefined library), for instance: “completed with status ‘success’”, “restored”, “succeeded”, “sync completed”. Then, candidate evidence patterns are ranked according to the following criteria:

- *sentiment score* [-1,1] (from very negative to neutral (0) to very positive);
- *criticality* level
- *status* of the event (active or cancelled);
- *closeness* of the event to the problem start time;
- *frequency* of event: its occurrence during the troubleshooting time frame;
- *entropy* of event (how unexpected/rare/uncertain or usual/expected is the event) for the object or application component. Rare events get higher rank according to the entropy criterion, meaning uncertainty might imply higher risk or stronger evidence.



Across this dimension, all configuration/compliance changes in property data within the time frame of interest are discovered. These are Boolean metrics or counter metrics. For importance ranking of property changes the following criteria are applied:

- *time-closeness* to the reported issue;
- *frequency* of the property (or property type) change within the troubleshooting time window;
- *entropy* of the property change (how rare or usual is this change) for the object or application component. Rare changes get higher rank according to the entropy concept.

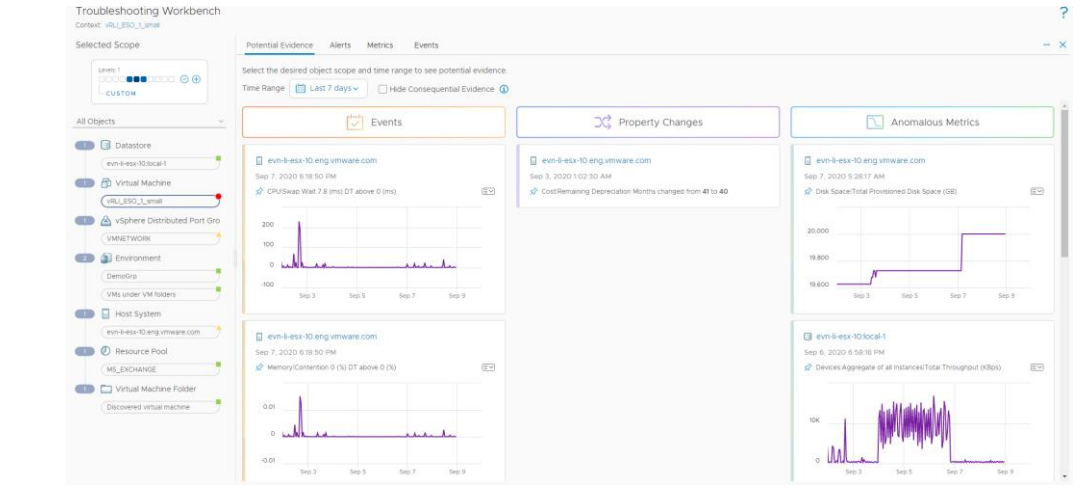
Interesting patterns in log data. Log data of infrastructure objects and application components within the investigation scope might contain important evidence about the performance issue. The following signals are of interest:

- *trending* error/warning/info messages from log stream of object or application;
- *topics* detected in log messages correlated to other evidence discovered;
- *anomalous divergences* in event type distributions by vR LI;
- *deviations* from baseline event type distributions.

*Interesting patterns in network flows***.** vR NI provides network-related diagnostics of the system. Here important patterns are bottleneck flows, change points in flows detected using our analysis for metric data.

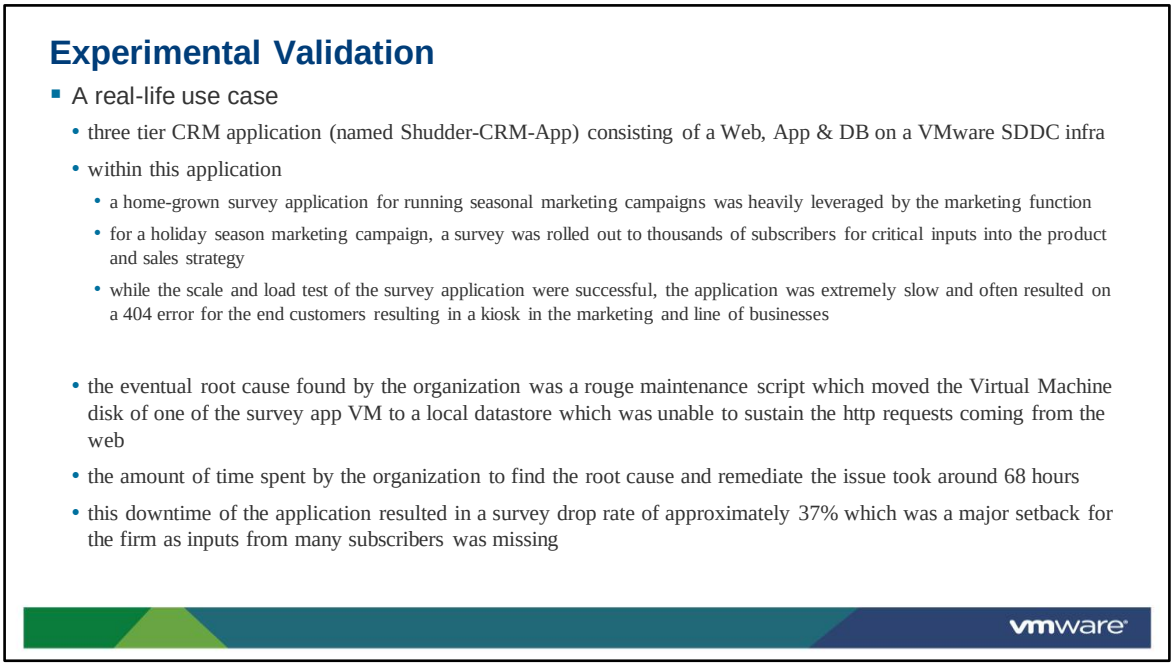
Interesting patterns in application traces. Wavefront's distributed application tracing provides us with an extra dimension for this evidence discovery analysis. Potential evidence patterns might include traces that are "atypical" or simply of low-frequency signatures (entropy concept).

Evidence Mining/Troubleshooting Workbench in Action



An IT object in the context under investigation for evidence of a misbehavior:

- Navigation through topology infra/neighborhood
- Navigation across timeline
- Change patterns across different data sources detected accordingly



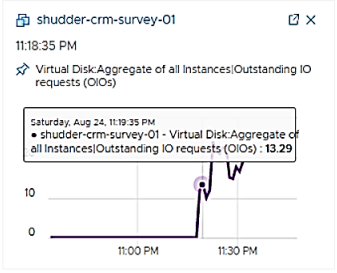
- SDDC – software-defined data center
- During the validation phase of the troubleshooting workbench inside vR Ops, multiple experiments with real-life use cases were conducted to arrive at the potential root cause for issues which customers face on a day to day basis. The simulation of such use cases leveraging a real application and building real life conditions helped mimic what customers go through. The goal of this exercise was to measure the effectiveness of the capabilities of the troubleshooting workbench including automated scope definition, time proximity and most importantly the relevance of the mined evidence by the system.

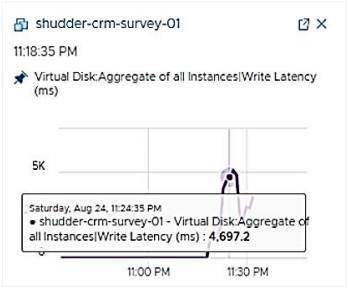
- In one such experiment a real-life use case associated to a media services provider was simulated. The company ran a three tier CRM application consisting of a Web, App & DB on a VMware SDDC infrastructure. Within this CRM application a home-grown survey application for running seasonal marketing campaigns was heavily leveraged by the marketing function. For a holiday season marketing campaign, a survey was rolled out to thousands of subscribers for critical inputs into the product and sales strategy. While the scale and load test of the survey application were successful, on eventual roll out in production, the application was extremely slow and often resulted on a 404 error for the end customers resulting in a kiosk in the marketing and line of businesses. The eventual root cause found by the organization was a rouge maintenance script which moved the Virtual Machine disk of one of the survey app VM to a local datastore which was unable to sustain the http requests coming from the web. The amount of time spent by the organization to find the root cause and remediate the issue took around 68 hours. This downtime of the application resulted in a survey drop rate of approximately 37% which was a major setback for the firm as inputs from many subscribers was missing.

Experimental Validation

- This use cases was simulated with some stress tools:
  - the VM as migrated over a local datastore when number of simulated users reached to 450
  - external load was generated using synthetic I/O on the local datastore (to introduce a bottleneck issue)
  - upon reaching close to 500 users, the web service hosting the survey crashed

- Then, our evidence mining algorithms were executed
  - detected patterns (effects not causes) in metrics





vmware

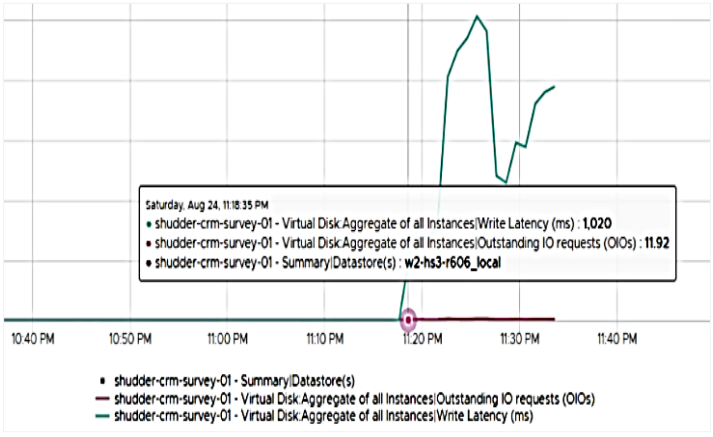
- in addition, external load was generated by using synthetic I/O on the local datastore using I/O Meter to introduce a bottleneck issue

- From this point on, to verify the evidence gathering capabilities of the troubleshooting workbench, the application in question was searched within vR Ops.
- Upon launching the troubleshooting workbench with the contextual application topology of the shudder application, potential evidence was presented along with signals of existing critical events which represented a high amount of storage read-write latency.

- The workbench was instrumental in pointing out at certain key evidence which helped validate the root cause by showcasing the key underlying changes resulting in a correlated event of storage performance degrading drastically. This was the root cause of the web application going down under drastic user pressure and underlying I/O bottlenecks. The first critical event which is a consequence of the issue, points at the storage outstanding I/O and Latency hitting the roof. This was detected automatically as an evidence by the workbench using the change point detection algorithm.

Experimental Validation

- Time correlation of evidences

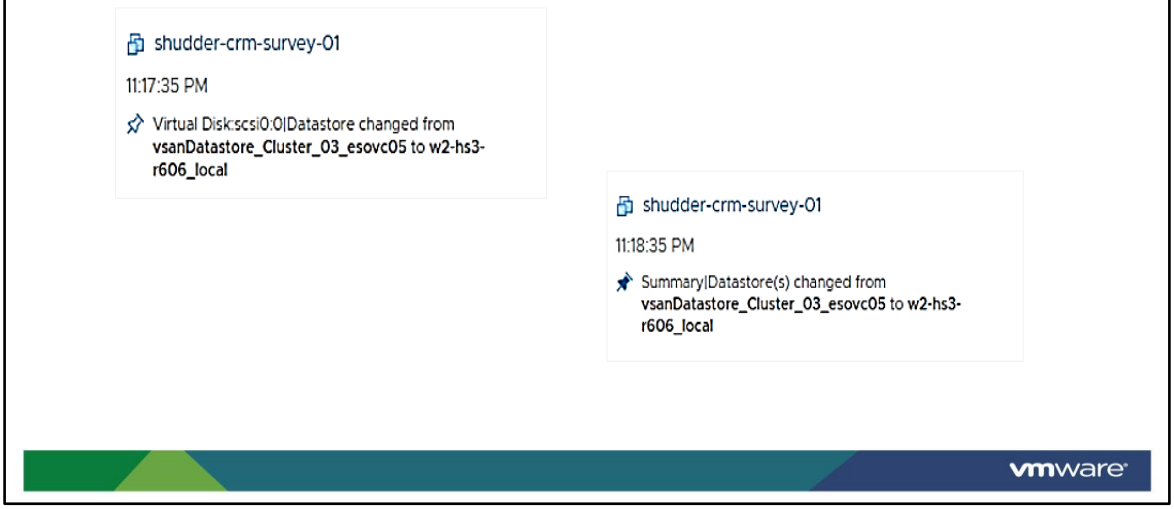


vmware

- Upon pinning the key evidences on a common scale, a perfect time and change pattern correlation was found across changes and causal evidences

Experimental Validation

- Other evidence detected pointing out the root cause



- This change was detected as a property change by the troubleshooting workbench with correlated timestamps for subsequent change points detected. Upon pinning the key evidence on a common scale, a perfect time and change pattern correlation was found across changes and causal evidence, hence solidifying the root cause of the problem

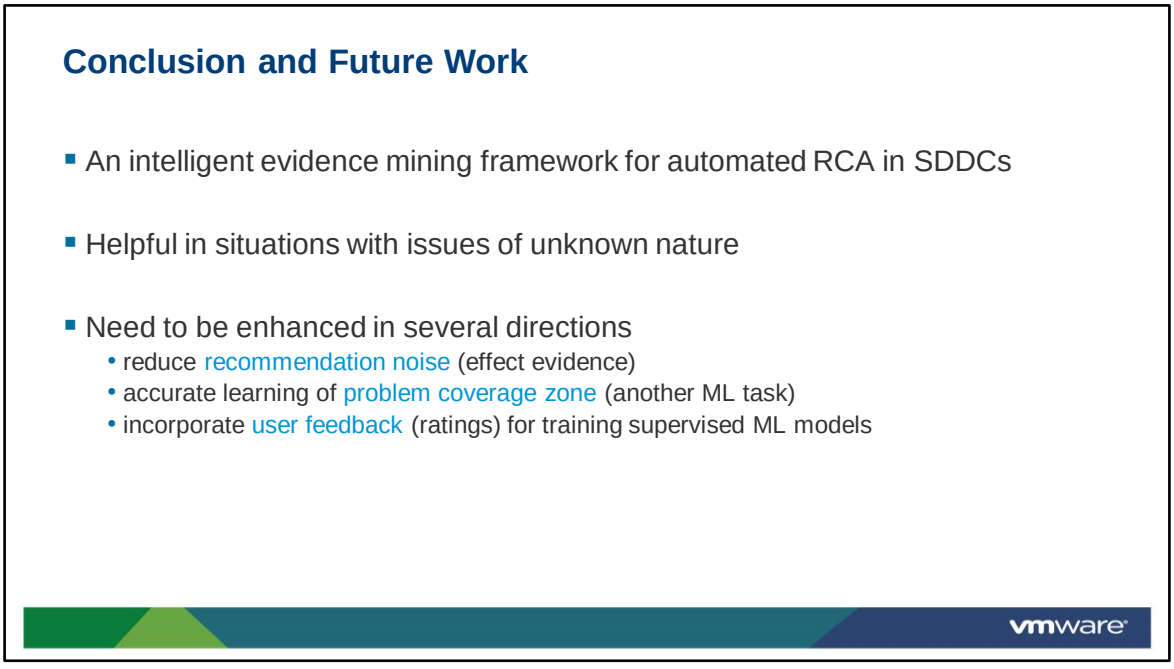
Experimental Validation

- Thus
  - the solution proved to be effective in detecting the root cause
  - within a huge number of metrics, events, and log changes
  - monitored from objects hosted on a complex SDDC
- Detection of helpful evidences and the rest of analysis with remediation took approx. 30 min
  - compare to 68-hour downtime occurred for the real-world environment
- Substantial reduction in MTTR and MTTI
- Based on a user survey
  - the functionality gained a satisfaction rating of 5.47/7 and a recommendation rating of 6.31/7

vmware

- The experiment proved the effectiveness of the troubleshooting workbench in detecting the root cause from thousands of metrics, events and log changes occurring in a dynamic environment over a large scope of objects hosted on a complex SDDC environment. The end to end issue detection, root cause analysis and remediation time reduced to a mere 30 minutes as compared to the 68-hour downtime faced by an equivalent application in real world environment, hence meeting the key objective of reducing the mean time to resolution (MTTR) and mean time to innocence (MTTI) with accurate and automated root cause analysis

- A survey with an open discussion forum was held with 22 existing users, screened for those using the latest release of the product that premiered the troubleshooting functionality. Several participants have successfully used the troubleshooting workbench to troubleshoot and resolve a real issue. The functionality gained a satisfaction rating of 5.47/7 and a recommendation rating of 6.31/7.



We introduced a novel intelligent troubleshooting framework for mining evidence of performance problems in data centres. It is based on combination of data science and ML algorithms to discover important patterns across various type of data that might explain the origin of the problem of an unknown nature. We also outlined how it can be further extended and enhanced. Initial implementation demonstrates significant power of the approach in automatically recommending accurate evidence in experimental application performance diagnostic and in real customer environments. Further plans include not only improving user experience on how indicatively we can organize the evidence patterns in terms of trend lining the evolution of the problem (their densities across time axis and across topology hierarchies), but also enhancing analytics power of the workbench in several directions:

- accurate learning of problem coverage zone is an important ML task, which will improve noise degree of our recommendations;
- incorporation of user feedback (ratings) on the recommended items collected over time would help us in filtering out non-indicative patterns (meaning, they are not likely to be causative);
- alternatively, ratings can be used in importance ranking. Moreover, ratings can be used for labelling data and training supervised ML models. Thus, we'll be able to identify/predict complex incidents composed of various type of evidence in the data. In this way, the algorithms will be tuned to the customer environment and application nature.