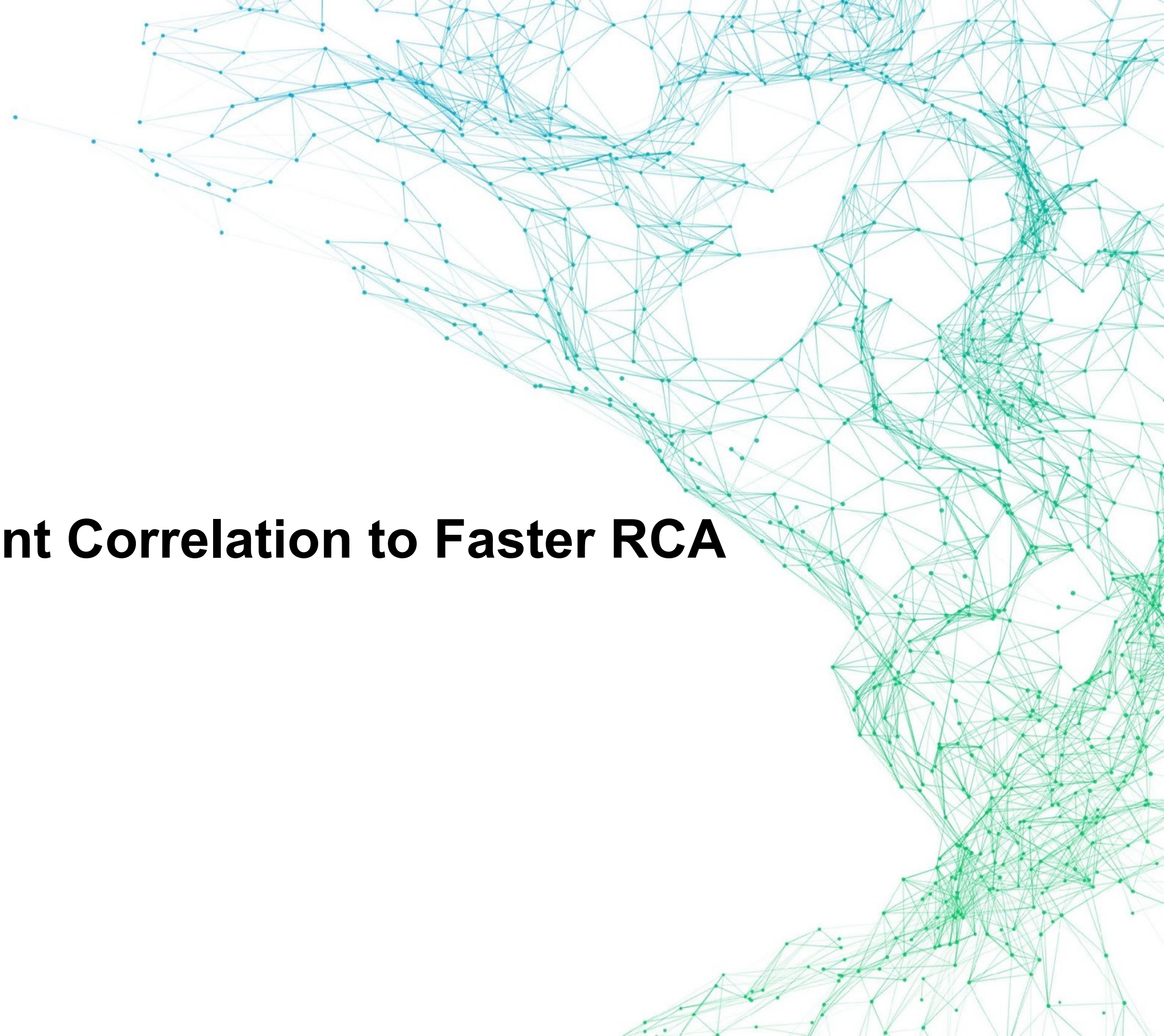
APPDYNAMICS

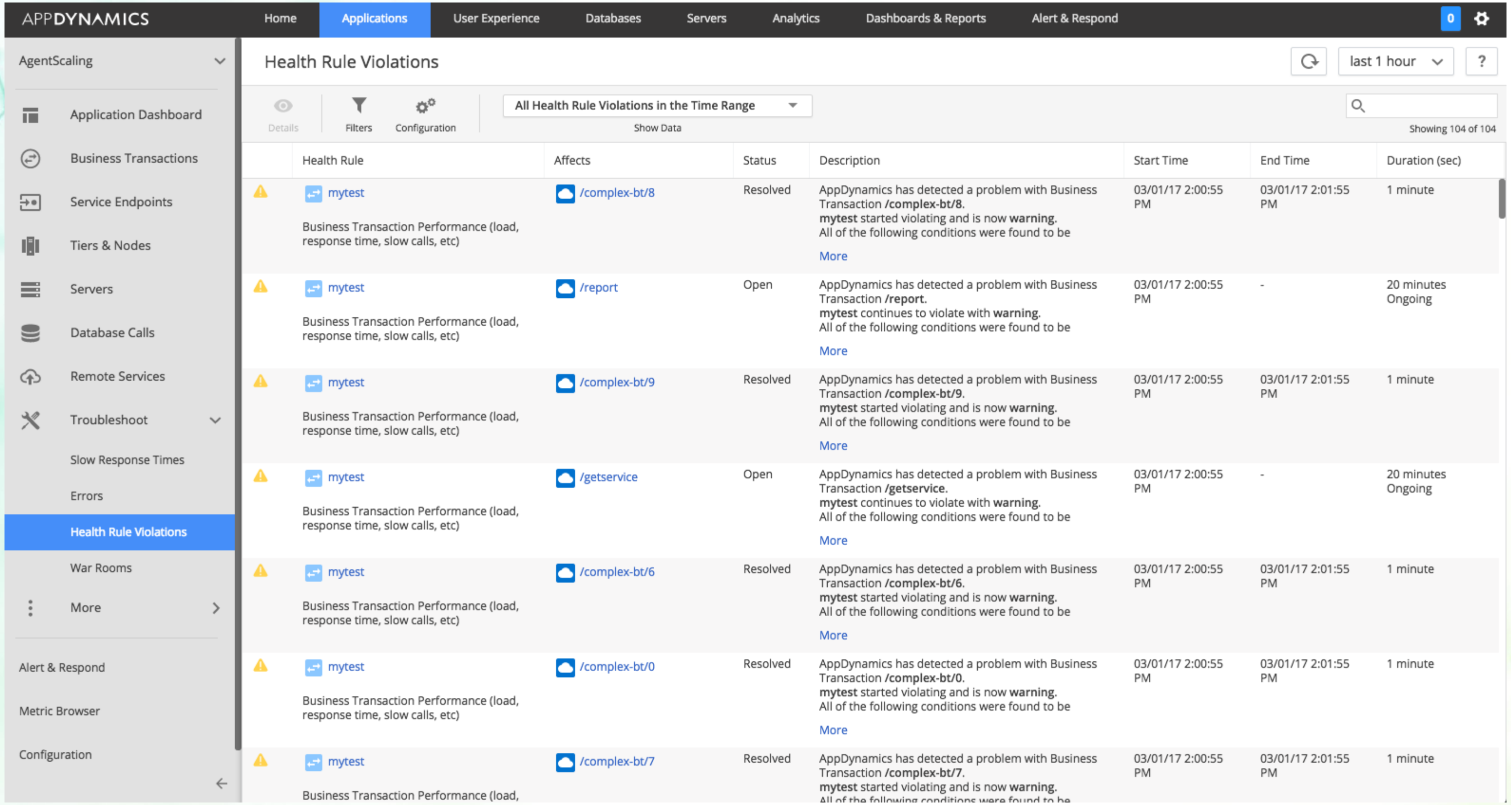
Incident Analytics

— Intelligent Incident Correlation to Faster RCA

Mar 10, 2017



The Status Quo

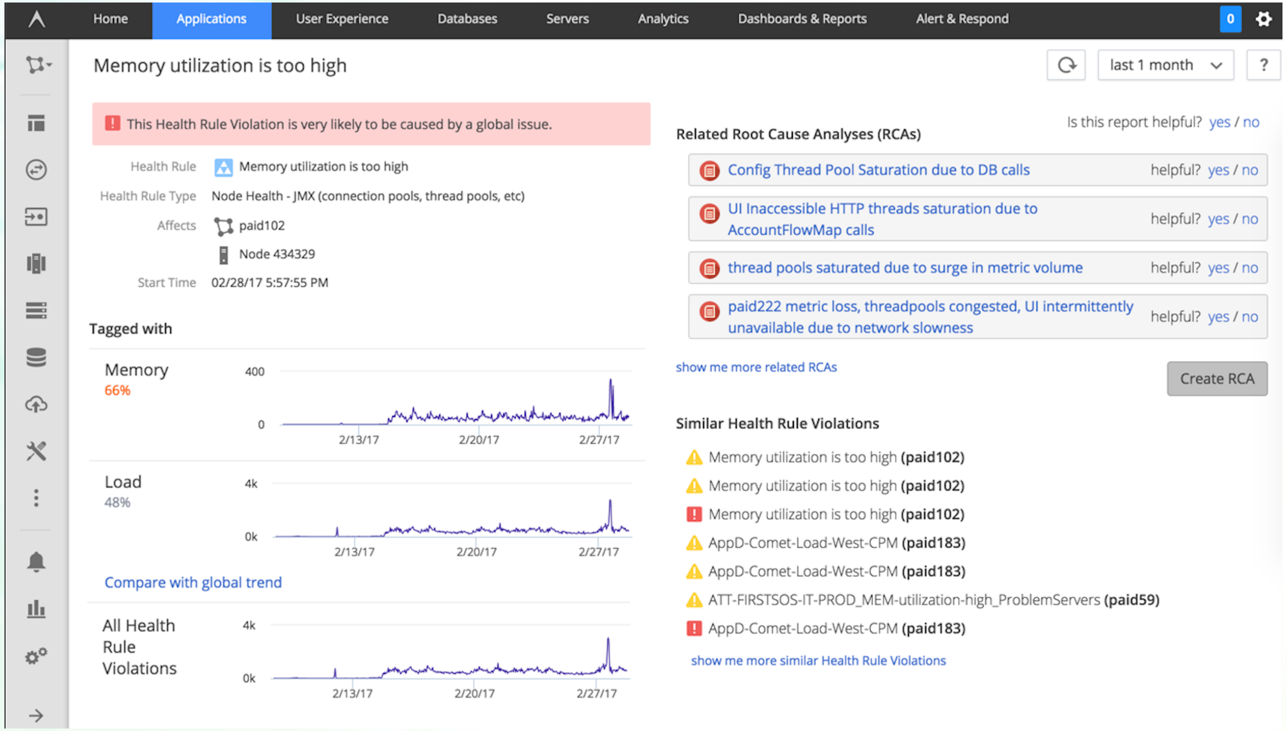


Health Rule Violations

Showing 104 of 104

	Health Rule	Affects	Status	Description	Start Time	End Time	Duration (sec)
⚠	mytest Business Transaction Performance (load, response time, slow calls, etc)	/complex-bt/8	Resolved	AppDynamics has detected a problem with Business Transaction **/complex-bt/8.** **mytest** started violating and is now **warning.** All of the following conditions were found to be More	03/01/17 2:00:55 PM	03/01/17 2:01:55 PM	1 minute
⚠	mytest Business Transaction Performance (load, response time, slow calls, etc)	/report	Open	AppDynamics has detected a problem with Business Transaction **/report.** **mytest** continues to violate with **warning.** All of the following conditions were found to be More	03/01/17 2:00:55 PM	-	20 minutes Ongoing
⚠	mytest Business Transaction Performance (load, response time, slow calls, etc)	/complex-bt/9	Resolved	AppDynamics has detected a problem with Business Transaction **/complex-bt/9.** **mytest** started violating and is now **warning.** All of the following conditions were found to be More	03/01/17 2:00:55 PM	03/01/17 2:01:55 PM	1 minute
⚠	mytest Business Transaction Performance (load, response time, slow calls, etc)	/getservice	Open	AppDynamics has detected a problem with Business Transaction **/getservice.** **mytest** continues to violate with **warning.** All of the following conditions were found to be More	03/01/17 2:00:55 PM	-	20 minutes Ongoing
⚠	mytest Business Transaction Performance (load, response time, slow calls, etc)	/complex-bt/6	Resolved	AppDynamics has detected a problem with Business Transaction **/complex-bt/6.** **mytest** started violating and is now **warning.** All of the following conditions were found to be More	03/01/17 2:00:55 PM	03/01/17 2:01:55 PM	1 minute
⚠	mytest Business Transaction Performance (load, response time, slow calls, etc)	/complex-bt/0	Resolved	AppDynamics has detected a problem with Business Transaction **/complex-bt/0.** **mytest** started violating and is now **warning.** All of the following conditions were found to be More	03/01/17 2:00:55 PM	03/01/17 2:01:55 PM	1 minute
⚠	mytest Business Transaction Performance (load,	/complex-bt/7	Resolved	AppDynamics has detected a problem with Business Transaction **/complex-bt/7.** **mytest** started violating and is now **warning.** All of the following conditions were found to be	03/01/17 2:00:55 PM	03/01/17 2:01:55 PM	1 minute

Why analyze when you can act?



Memory utilization is too high

This Health Rule Violation is very likely to be caused by a global issue.

Health Rule	Memory utilization is too high
Health Rule Type	Node Health - JMX (connection pools, thread pools, etc)
Affects	paid102
	Node 434329
Start Time	02/28/17 5:57:55 PM

Tagged with

Memory 66%

Load 48%

Compare with global trend

All Health Rule Violations

Related Root Cause Analyses (RCAs)

Is this report helpful? yes / no

- Config Thread Pool Saturation due to DB calls — helpful? yes / no
- UI Inaccessible HTTP threads saturation due to AccountFlowMap calls — helpful? yes / no
- thread pools saturated due to surge in metric volume — helpful? yes / no
- paid222 metric loss, threadpools congested, UI intermittently unavailable due to network slowness — helpful? yes / no

show me more related RCAs

Create RCA

Similar Health Rule Violations

- Memory utilization is too high **(paid102)**
- Memory utilization is too high **(paid102)**
- Memory utilization is too high **(paid102)**
- AppD-Comet-Load-West-CPM **(paid183)**
- AppD-Comet-Load-West-CPM **(paid183)**
- ATT-FIRSTSOS-IT-PROD_MEM-utilization-high_ProblemServers **(paid59)**
- AppD-Comet-Load-West-CPM **(paid183)**

show me more similar Health Rule Violations

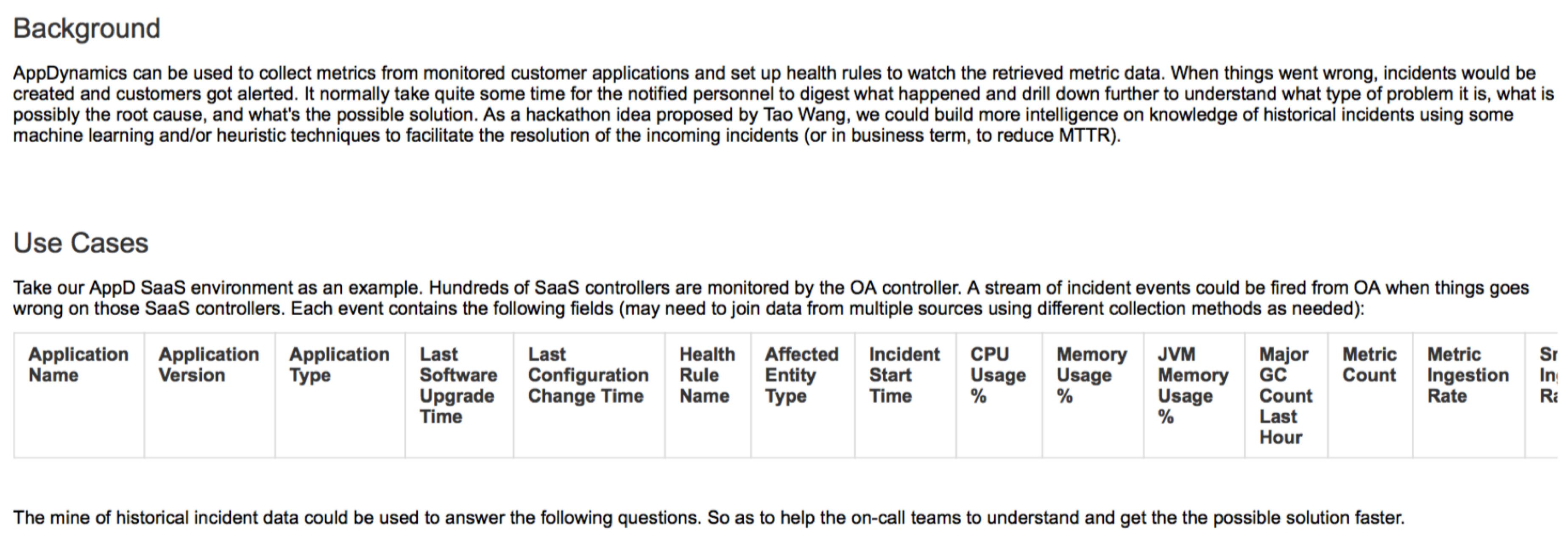
Thank You

Incident Analytics

— Intelligent Incident Management

APPDYNAMICS

Appendix



Background

AppDynamics can be used to collect metrics from monitored customer applications and set up health rules to watch the retrieved metric data. When things went wrong, incidents would be created and customers got alerted. It normally take quite some time for the notified personnel to digest what happened and drill down further to understand what type of problem it is, what is possibly the root cause, and what's the possible solution. As a hackathon idea proposed by Tao Wang, we could build more intelligence on knowledge of historical incidents using some machine learning and/or heuristic techniques to facilitate the resolution of the incoming incidents (or in business term, to reduce MTTR).

Use Cases

Take our AppD SaaS environment as an example. Hundreds of SaaS controllers are monitored by the OA controller. A stream of incident events could be fired from OA when things goes wrong on those SaaS controllers. Each event contains the following fields (may need to join data from multiple sources using different collection methods as needed):

Application Name	Application Version	Application Type	Last Software Upgrade Time	Last Configuration Change Time	Health Rule Name	Affected Entity Type	Incident Start Time	CPU Usage %	Memory Usage %	JVM Memory Usage %	Major GC Count Last Hour	Metric Count	Metric Ingestion Rate	Sr In Ra

The mine of historical incident data could be used to answer the following questions. So as to help the on-call teams to understand and get the the possible solution faster.

APPDYNAMICS

What's the incident trend?

Static historical incident graph can show some trend. Compared with the deployed controller version data overtime, It could show which version of controller is more problematic. For example, there are usually more issues upon the roll out of the first major release (E.g. 4.2.0), the number of incidents went steadily lower in each 4.2.x release rolled out. E.g. the following data could be helpful for analyzing the overall DevOps situation:

- Overall incident count over last 3 months.
- Top 5 applications with incident in last 3 months
- Top 5 Incident categories in last 3 months
- Incidents resolved with RCA vs open Incidents without RCA

Are there a global issue?

Are the incidents global across controllers, e.g. due to data center network outage, load balancer misconfiguration etc.? Issues are global if the incidents started in similar time frame, across multiple controllers, and different controller versions/types, load. In one word, very diversified types of incidents in a short time frame, let's say within last 10 minutes rolling time window. Some streaming analytics tool like Spark could be used to analyze the incident stream and predict if there is a global issue or not.

What is the category of the new incident?

Classifying incoming incidents with incident categories, so as to properly grouping/tagging each of them. The following could be the categories that incidents can be grouped into by heuristic knowledge. More sophisticated machine learning technics can be applied to discover subtle way of categorization/clustering incidents.

- Upgrade related: incident happens after an upgrade to a more defective software version.
- Configuration change related: Some software configuration change can affect the runtime behavior.
- Load related: Some problem happens when load exceed certain threshold.
- Memory problem: E.g. High GC activity and high JVM memory usage. Could be memory leak, or low memory configuration etc.
- High user load: High number of concurrent users logged in controller UI and doing work.
- Recurring intermittent: The problem happens intermittently and happened before, no root cause identified yet. But it happened multiple times with the same application/tier/node before.

Sometimes an incident could fall into multiple categories. Each category should have a playbook for diagnosing, fixing or alleviating the problems. Each type/category could have a graph itself with historical data which could show the trend.

What are the similar incidents with RCA?

Incident similarity between current incident and previous incidents that already have a RCA (root cause analysis) tagged. E.g. Indexing incidents using Lucene with similarity search, similarity search with boosting on different fields can be done. A similarity score could be presented to show how similar they are...

Incident Clustering

When a new incident being detected by APM product like AppDynamics, the similarity of historical incidents from all controllers to this incident is very valuable context to provide insights for evaluating the status and guiding the next steps of investigation. The situation applies to analyzing any old incident too. However, determining the similarity of incidents is complex because an incident has many factors. We may categorize the factors in two buckets.

First bucket is relatively static:
- Whether controller is dedicated or multi-tenancy
- Incident name
- Associated health rule name
- Affected entity type
- Start time

Second bucket is dynamic:
- Controller version change due to upgrades
- Server monitoring enabled or not
- Blitz enabled or not
- Many feature flags or AB tests enabled or not individually

We propose to compute the similarity of incidents across all controllers automatically. To be more specific, we came out a modified version of K-Modes clustering algorithm based on all the incidents for it. Through that, we figure out the N mostly similar incidents for this interesting incident. This opens the doors to provide the context with answers to many interesting questions: Is this incident due to a global event such as network outage? We can determine it if N similar incidents are randomly distributed across multiple controllers in a short period of time around the interesting incident. What is the most significant factor that may cause this incident? By further analyzing the N similar incidents on each factor automatically, we can order the factors in order of significance.

Assume some incidents have been tagged with RCA(root cause analysis) documents. What is the most likely RCA for this incident? We can compute it by scaling up/down the N similar incidents until this cluster includes the desired count of tagged RCAs.
Simply display top N similar incidents to the interesting incident and see if some of those rings a bell.
This way, for an known issue, the lookup for RCA is almost instant. For unknown issue, the sophisticated context provides insights and good starting point for investigation, from machine learning of previous incidents.

How are Incidents correlated?

- Collect more context information when the incident happened.
  - About 10 fields/factors collected
  - Incidents can come from different sources/applications
- An algorithm to determine how ‘similar’ 2 given incidents are.
  - Machine learning to be used to improve the algorithm accuracy
- Most similar incidents can be retrieved based on similarity score
- Most relevant RCAs can be retrieved from most similar incidents that have RCA already.
- Classifiers used to classify each incident into specific category
- Each incident category can have historical data for charting, so as to visually spot a possible trend if there is any.
- Code deploy/configuration change event can potentially be overlaid on the incident trend chart for further correlation.

Script for the 90-seconds business pitch

Today most of our customers find it time-consuming to identify a root cause of incident, based on the minimal information we provide. For our hackathon project, we've built a tool that deep insights into the customers incident history. and get to resolution for an incident much faster.

Take one of our SaSS customer as example. Once I get an alert, instead of landing on a simple dialog showing basic information, I immediately get a list of relevant RCAs to take actions on. I can vote for the helpful ones so that our machine learning algorithm's further recommendation would be more accurate. If there's no relevant RCA, I can create one from our controller UI, and tag it with relevant categories for the root cause.

For each incident, we also automatically classify it based on its potential root causes and give you the explanation. We display the a chart of incidents in that category so customers can relate the incident with what's happening across apps, identify the global pattern or trend which can be helpful. Finally, users can drill down into list of similar incidents which may occur on different apps.

We're confident that we can fit this tool in product and customers will find it useful for managing and resolving their production problems faster and more easily than ever.