April 12, 2019

AUTOMATED ROOT CAUSE ANALYSIS FOR MITIGATING RISK TO APPLICATION PROGRAMMING INTERFACES IN INTERNET OF THINGS INFRASTRUCTURE

Sai Kiran Reddy Malikireddy

Lalit Jain

Bipin Algubelli

Chandra Veluru

Vivek Parekh

See next page for additional authors

Inventor(s)
Sai Kiran Reddy Malikireddy, Lalit Jain, Bipin Algubelli, Chandra Veluru, Vivek Parekh, Dileep Kumar, and Vimal Suba

AUTOMATED ROOT CAUSE ANALYSIS FOR MITIGATING RISK TO APPLICATION PROGRAMMING INTERFACES IN INTERNET OF THINGS INFRASTRUCTURE

AUTHORS:
Sai Kiran Reddy Malikireddy
Lalit Jain
Bipin Algubelli
Chandra Veluru
Vivek Parekh
Dileep Kumar
Vimal Suba

ABSTRACT

Techniques are described herein for detecting anomalies in Application Programming Interface (API) request/response/payloads and determining probable root causes by analyzing the log, request payload, response payload, and metric data of the hosts serving the API in the Internet of Things (IoT) infrastructure. The API response time is captured in the log messages and the anomalies are detected using machine learning and statistical techniques. Given the anomalies in API response time and the probable root cause in real time, the IoT administrator may identify the root cause without manual effort and remediate the anomaly by taking appropriate action. Thus, the techniques described herein enable identification of treatments for API abuse from users or system failures because of a chain reaction on a host.

DETAILED DESCRIPTION

In the absence of an automated anomaly detection and root cause analysis module, it is very difficult or not feasible to observe several million or even billions of log messages each day in real time in an Internet of Things (IoT) infrastructure. Often the problem might be known/detected in the event of a catastrophic occurrence but it can nonetheless require several hours or days for a human to determine the problem and perform a root cause analysis.

The log data comprises very rich and abundant data captured and indexed frequently (e.g., every minute) regarding information about the system and processes. The client (e.g., edge devices or customer) may access Application Programming Interfaces

(APIs) in the IoT infrastructure to obtain and/or process information. Among the log data, API response time is the metric that may be used to determine whether the client is able to obtain the authorized access and results in the appropriate time to avoid Service Level Agreement (SLA) violations. If the API response time for any API increases or does not fall within the normal window, this may indicate an anomaly that requires investigation. Possible reasons for the increase in API response time include that a given host is experiencing resource contention, or that there exists some anomalous behavior for the API host serving a specific API or anomalous behavior for host responding to the API host. The root cause may be derived from the log and metric data of the API hosts.

Figure 1 below illustrates an example client request API gateway. The client interacts with the system using several different API calls that are passed to the load balancer. The load balancer uses the appropriate policy and sends the API request to the relevant API host (e.g., a web service application). These API hosts may interact with the system and obtain the API query result. The log and metric data are collected for these API hosts in the Log Aggregator and Time Series DB module, respectively.

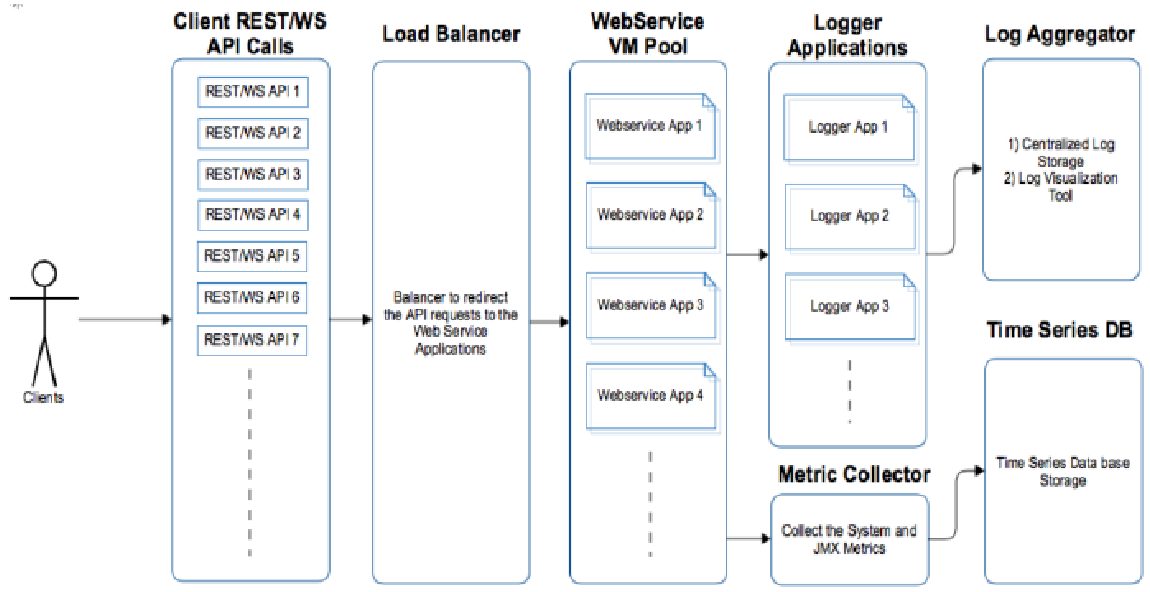


Figure 1: Client request API gateway

As illustrated in Figure 2 below, these logs and metric data are later used to detect anomalies in the API response. The system may thus perform automated root cause analysis.

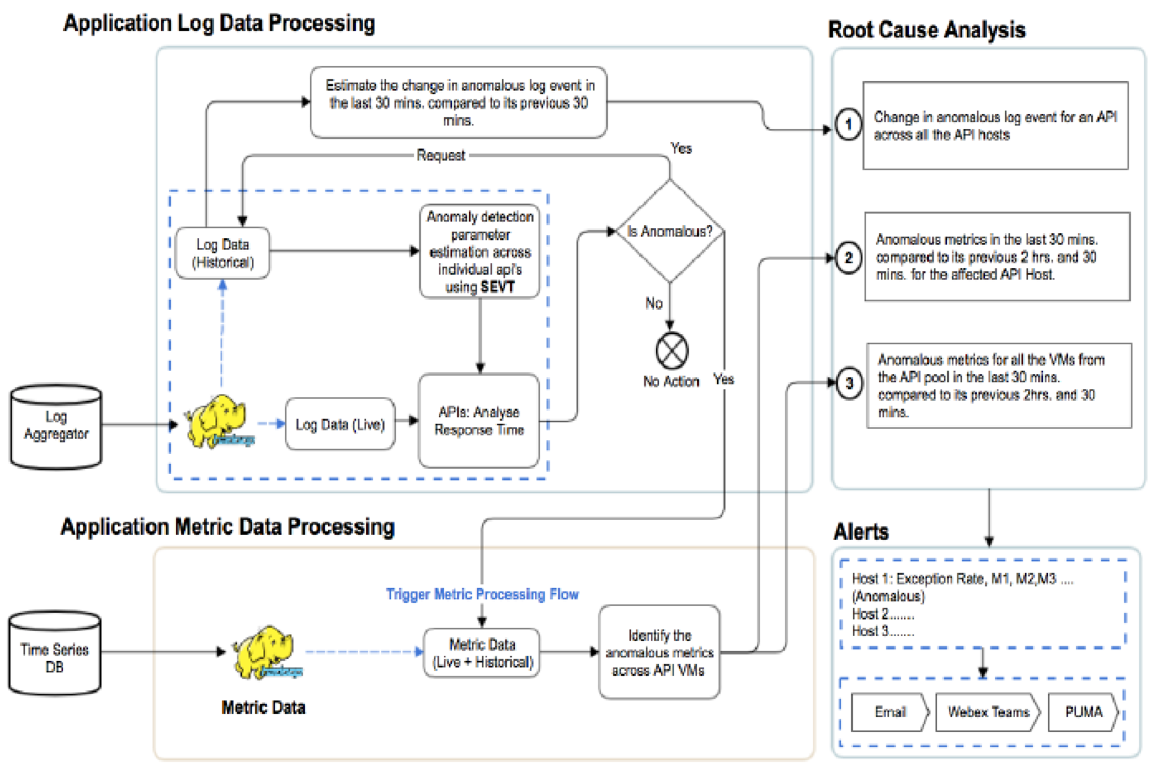

Application Log Data Processing
Root Cause Analysis
Estimate the change in anomalous log event in the last 30 mins. compared to its previous 30 mins.
Request
Yes
Log Data (Historical)
Anomaly detection parameter estimation across individual api's using SEVT
Is Anomalous?
No
No Action
Yes
Log Aggregator
Log Data (Live)
APIs: Analyse Response Time
Change in anomalous log event for an API across all the API hosts
Anomalous metrics in the last 30 mins. compared to its previous 2 hrs. and 30 mins. for the affected API Host.
Anomalous metrics for all the VMs from the API pool in the last 30 mins. compared to its previous 2hrs. and 30 mins.
Application Metric Data Processing
Alerts
Trigger Metric Processing Flow
Time Series DB
Metric Data (Live + Historical)
Identify the anomalous metrics across API VMs
Metric Data
Host 1: Exception Rate, M1, M2,M3 .... (Anomalous)
Host 2.......
Host 3.......
Email
Webex Teams
PUMA

Figure 2: API anomaly detection and root cause analysis using log and metric data

The live log and metric data is collected across all API hosts and stored in a Hadoop File System (HDFS). Thus, after a few days, both historical and live data may be available for both the log and metric data. The log data may be unstructured, but any subset thereof may be structured. Log message with an API response may have a common internal structure, and the following attributes may be derived using a regular expression such as:

timestamp, host_name, api_name, user_name, response time (in msec)

This information may be fetched for historical data (e.g., 15 days) for each API and parameters may be estimated daily for anomaly detection using statistical Extreme Value Theory (EVT), which may be suitable for anomaly/failure detection. According to EVT, regardless of the original distribution of function *f*, the normalized sample extrema (maxima/minima) will converge to one of the three extreme value distributions, namely Gumbel, Frechet, and Weibull. Among these three distribution types, Weibull applies to bounded distributions. As values are bounded (i.e., greater than or equal to zero), the extreme values of a metric may be modeled by the Weibull distribution. The Weibull probability distribution function is shown in Equation 1 below.

$$f(x;\lambda,\kappa) = \begin{cases} \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1} e^{-(x/\lambda)^{\kappa}} & x \geq 0, \\ 0 & x < 0, \end{cases} \tag{1}$$

where κ > 0 and λ > 0.

Once the shape (κ) and location (λ) parameters are estimated over the maxima (extrema) of historical data, the current metric value (x) anomalous probability may be computed using the Weibull Cumulative Distribution Function (CDF) as shown in Equation 2 below.

$$F(x;k,\lambda) = 1 - e^{-(x/\lambda)^{k}} \tag{2}$$

Applying a threshold (e.g., 0.9) on the probability value estimated in Equation 2 may help determine whether the metric value (here, response time) is an anomaly. Rather than applying a static ad-hoc threshold on any metric value to detect an anomaly, a threshold is dynamically chosen to detect an anomaly given its historical data.

Given the live log messages with API response time, the estimated EVT parameters may be used to estimate whether the response time is anomalous (e.g., in scale and shape) based on historical data for that API using Equation 2. If the API response is anomalous, then root cause analysis may be performed as shown in Figure 2 using log and metric data.

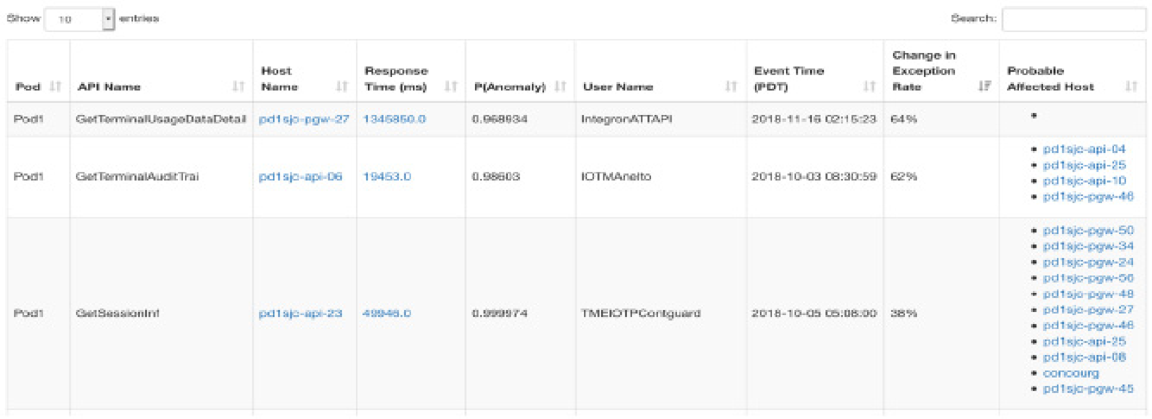
Root cause analysis may be performed using log data. The log data contains a rich set of information such as log messages which contain keywords such as "exception," "error," etc. For example, the anomalous log may be a log message with a different type of exception or error message than the API host (or the host responsible for serving the API host) has previously raised. One key metric that may be used for root cause analysis is the change in anomalous log message count in the near past compared to the historical past (e.g., the change in the anomalous log event count for all the API hosts in the last thirty minutes compared to the previous thirty minutes). If the rate of change is high for the estimated anomalous log count, the log message may provide insights into the problem.

Root cause analysis may also be performed using metric data. An abnormal response for the API may be caused by a specific API host (or API hosts) being slowed due to resource contention (e.g., many requests) or some other reason. Thus, several system-level metrics (e.g., Central Processing Unit (CPU), memory, network, Input/Output (I/O), Java Virtual Machine (JVM), etc.) and application level metrics (e.g., datasource_numbusyconnections, datasource_numconnections, scrape_duration _seconds, etc.) may be analyzed if their behavior has changed in the recent past for root cause analysis.

In one example, the system determines whether a specific metric in the API host is anomalous/extreme in the more recent past compared to historical behavior using statistical EVT. For instance, it may be determined whether any metric behavior is anomalous in the most recent thirty minutes compared to its previous two and a half hours for the affected API host (or all API hosts). First, the EVT distribution may be fit using Equation 1 for data regarding a given metric for the two and a half hour period and compute its scale and shape parameter. The probability of extremes for all the metric values in the most recent thirty minutes may subsequently be computed. Anomalous behavior metrics may be estimated using Equation 3 below.

$$P_{MetricExtreme} = \frac{\sum_{i=1}^{n} \; 1 \; if \; P_{extreme} > th_{metric} \; else \; 0}{n} \tag{3}$$

If $P_{\text{MetricExtreme}} > th_{\text{counter}}$ then the metric is considered to be anomalous. If the anomaly is API response time, for example, once the root cause analysis is performed for both the log and metric data, all the information may be provided to the IoT administrator, as shown in Figure 3 below. Figure 3 provides the P(Anomaly) for the affected API call, the change in the exception rate for the anomalous log event, and a list of hosts that are likely affected.



Pod	API Name	Host Name	Response Time (ms)	P(Anomaly)	User Name	Event Time (PDT)	Change in Exception Rate	Probable Affected Host
Pod1	GetTerminalUsageDataDetail	pd1sjc-pgw-27	1345850.0	0.968934	IntegronATTAPI	2018-11-16 02:15:23	64%	•
Pod1	GetTerminalAuditTrai	pd1sjc-api-06	19453.0	0.98603	IOTMAnelto	2018-10-03 08:30:59	62%	• pd1sjc-api-04 • pd1sjc-api-25 • pd1sjc-api-10 • pd1sjc-pgw-46
Pod1	GetSessionInf	pd1sjc-api-23	49946.0	0.999974	TMEIOTPContguard	2018-10-05 05:08:00	38%	• pd1sjc-pgw-50 • pd1sjc-pgw-34 • pd1sjc-pgw-24 • pd1sjc-pgw-56 • pd1sjc-pgw-48 • pd1sjc-pgw-27 • pd1sjc-pgw-46 • pd1sjc-api-25 • pd1sjc-api-08 • concourg • pd1sjc-pgw-45

Figure 3: Anomaly detection and root cause analysis in API response

As illustrated in Figure 4, when the IoT administrator selects any of the affected hosts, the anomalous metrics are plotted. Figure 4 shows the example of an affected host with affected metrics when there is an anomaly in API response time.

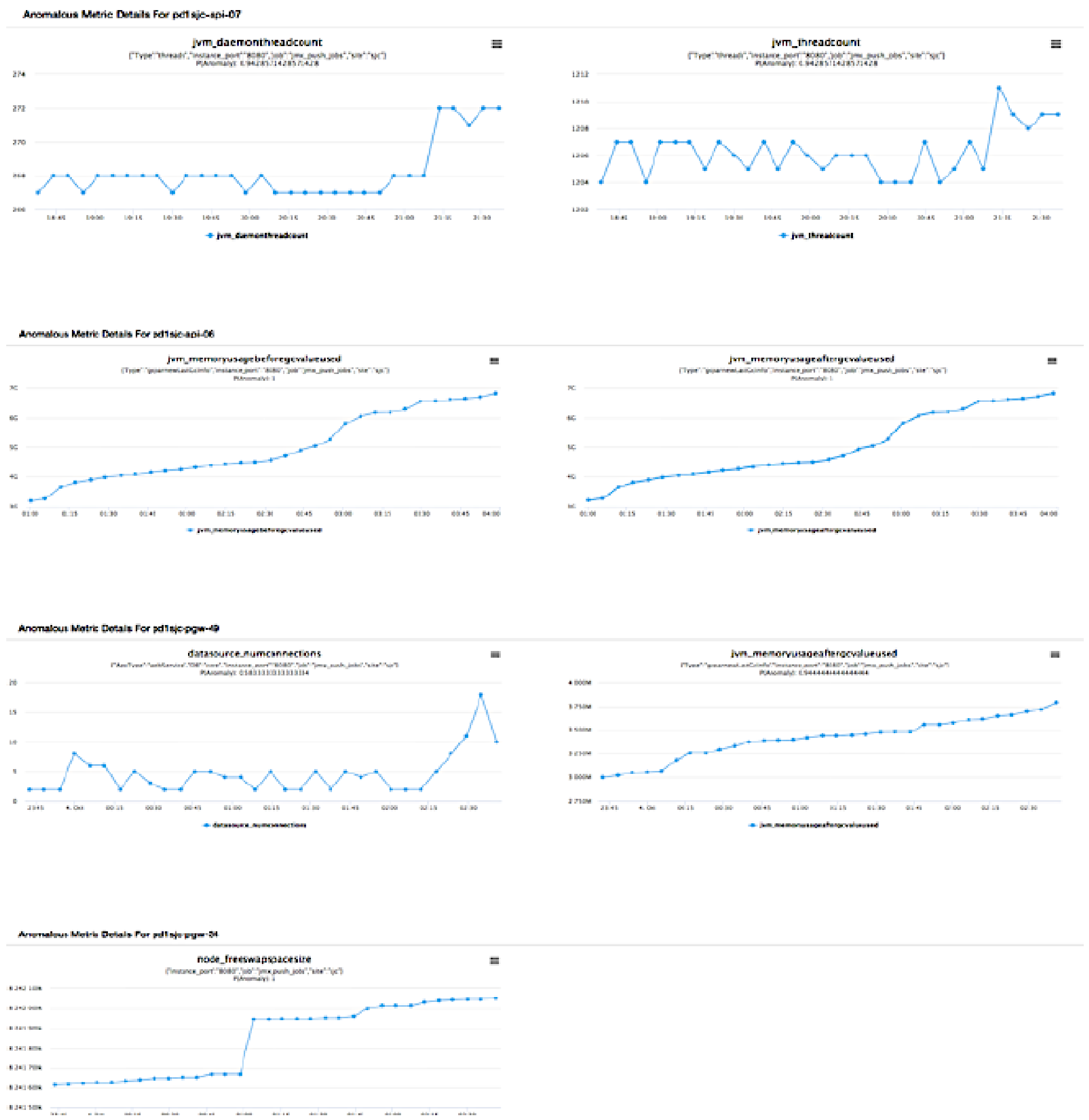


Figure 4: Example of affected host with affected metrics when there is anomaly in API response time.

It will be appreciated that the techniques presented herein may involve correlation of any anomaly, not just response time. The overall system may collectively examine the dependent systems and identify the root cause of the specific slowness. The metrics may be selected automatically and machine learning models may be updated in real-time based on the feature set selected for analysis. An anomaly at one point in time may not be an anomaly at a future point in time as the system evolves with complexity. Additionally, the system may provide a possible mitigation path to minimize the impact and contain the anomaly. An event may be triggered (or an anomaly detected) in the system for many reasons, such as slowness in infrastructure, backend systems being overloaded/unavailable, abusive behavior of certain users, JVM processing delays due to garbage collection, system overload, cascading impact to other components in an IoT-based infrastructure, application code, etc. The system described herein may provide mitigation steps based on prior learnings.

In summary, techniques are described herein for detecting anomalies in API request/response/payloads and determining probable root causes by analyzing the log, request payload, response payload, and metric data of the hosts serving the API in the IoT infrastructure. The API response time is captured in the log messages and the anomalies are detected using machine learning and statistical techniques. Given the anomalies in API response time and the probable root cause in real time, the IoT administrator may identify the root cause without manual effort and remediate the anomaly by taking appropriate action. Thus, the techniques described herein enable identification of treatments for API abuse from users or system failures because of a chain reaction on a host.