December 04, 2018

AUTOMATED CLIENT CLASSIFICATION AND POLICY ENFORCEMENT THROUGH CLOUD APPLICATION PROGRAMMING INTERFACE GATEWAY

Xueqiang Ma

Kumar Sankaralingam

Shubhojit Chattopadhyay

Kalyan Ghosh

AUTOMATED CLIENT CLASSIFICATION AND POLICY ENFORCEMENT THROUGH CLOUD APPLICATION PROGRAMMING INTERFACE GATEWAY

AUTHORS:
Xueqiang Ma
Kumar Sankaralingam
Shubhojit Chattopadhyay
Kalyan Ghosh

ABSTRACT

Techniques are described for identifying key parameters for classifying interactive clients and daemon clients. Machine learning algorithms based on Principal Component Analysis (PCA) and enhanced Decision Tree (DT) are utilized together with HyperText Transfer Protocol (HTTP) header analysis to accurately classify clients. Rate-limiting effects are monitored as a feedback to perform re-enforcement learning to improve accuracy of the algorithm. The Application Programming Interface (API) gateway is one of the most important components to any cloud service and it needs to apply different rate-limiting policies on API requests based on different requirements of different types of clients. Traditional client identification with source Internet Protocol (IP) addresses are no longer suitable when moving a networking infrastructure to a cloud platform.

DETAILED DESCRIPTION

When a service in the cloud is deployed, the Application Programming Interface (API) gateway provides the Representational State Transfer (REST) API interface to the rest of the world. It acts as the front gate to the service so it is strategically at the most important position to guarantee a secure and reliable tiered service to different types of clients. Each service's computing and networking resources are limited. Clients may access cloud services from anywhere in the cloud and in various different forms (e.g., from human users or automating crawling scripts) and from different types of devices. Since clients can be dynamically created and migrated, there is a need to differentiate the API accessing requirements from different types of clients.

In a typical example (e.g., a hybrid-cloud system), the REST API gateway implemented mainly receives API requests from, among others, the local site's user requests via a Graphical User Interface (GUI) application or remote Multi-Site Controller

(MSC) daemon process. MSC is a piece of software that makes periodical GET and POST REST calls to different sites to synchronize the configuration and operational data among sites. While the local user's request is interactive and has higher priority, the MSC's request to each site's API gateway is periodic. There are significantly more frequent requests to replicate configuration and operational data between network infrastructure sites. Hence, if one of the MSC API requests is lost, the same request will be subsequently received from the MSC again. Thus, it has lower real-time requirement than a local GUI and is more tolerable to REST request loss than user access requests from local GUI. It would be desirable to enforce a rate-limit on the MSC requests to prevent monopolization of the network infrastructure controller's API gateway so that local requests at each site cannot access the APIs. But it is not desirable to place the rate-limit on local applications such as a GUI to impact the user experience. Therefore, the API gateway's request throughput may be enhanced by performing rate-limiting on requests from the MSC, but not on requests from the local GUI.

For reasons described above, it is important to classify different types of clients and perform policy enforcement in a cloud environment. However, the source IP address may not be relied upon as an identification of the clients to enforce policy such as rate-limiting for at least two reasons. The first reason is that clients access a cloud service through various intermediate entities such as HyperText Transfer Protocol (HTTP) proxies and/or cloud load-balancers. As such, the traditional mechanism to use the client IP address to identify/classify the cloud clients is no longer suitable because the API gateway only sees the shared proxy IP address instead of the real client IP address. The second reason is that the client application may crash or migrate, and when it recovers or restarts, it can be assigned a new IP address, thereby rendering the old IP address obsolete.

Therefore, a dynamic and intelligent mechanism to identify clients and enforce policy automatically is called for to address these issues when a traditional data center networking infrastructure is moved to a cloud environment.

With the Decision Tree (DT) based client classification engine, the API gateway can classify requests from the MSC via its inter-request gap's mean and standard deviation parameters, and place the rate-limit only on the MSC requests but not on requests from local user interactive applications automatically. This is illustrated in Figure 1 below.

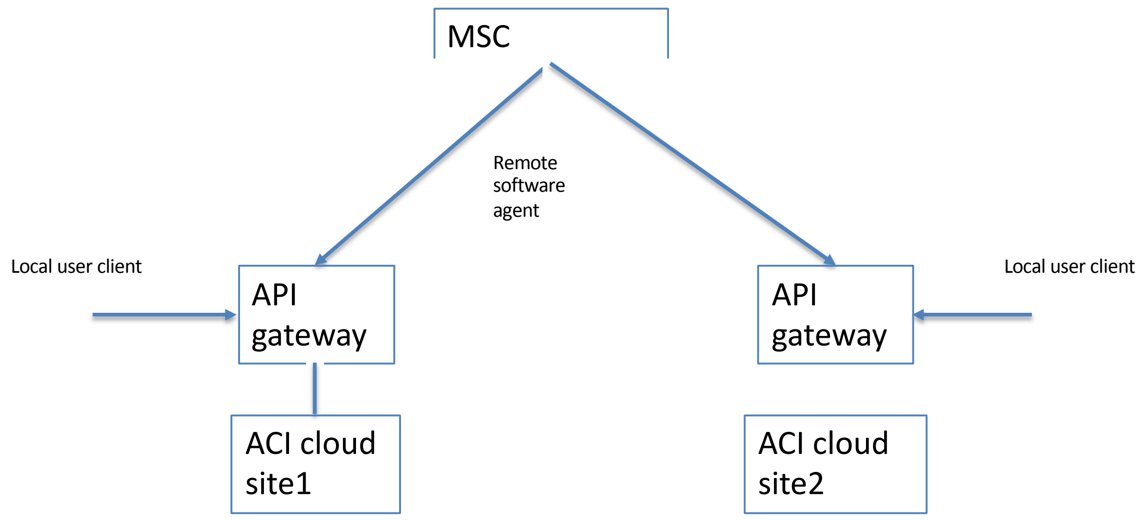

Fig.1 API Gateway in multi hybrid-cloud sites

Provided is a solution to use machine-leaning technology such as DT within a cloud service API gateway (e.g., a REST API service in a network infrastructure platform) to classify/identify clients and apply rate-limiting on API requests based on classification result to provide differentiated API services based on different requirements. The same mechanism can be utilized to fight against the most common Distributed Denial of Service (DDoS) attacks to cloud deployed services.

This is not a trivial application of generic machine learning algorithms because real-time performance is required for an API gateway, as the API gateway needs to process requests from clients and provide responses in real time. The specific domain knowledge in this case may be very useful to optimize and simplify the classification/regression issues, as a generic algorithm may not know which data is essential and may often rely on irrelevant data or even noise to perform a classification, which may cause problems such as over-fitting.

Several parameters have been experimentally determined to help efficiently distinguish human user interactive clients (I-clients), such as local GUIs, from software agent daemon clients (D-clients), such as the MSC.

One parameter is periodicity. D-clients tend to have a few characteristic queries and make those same characteristic queries periodically at fixed intervals. By contrast, I-clients tend to make relatively random queries without obvious periodicity. This property of D-clients may be exploited by performing dimension reduction and Principal Component

Analysis (PCA) to extract the characteristic queries and then identify their repeating intervals from the sequence of Uniform Resource Locators (URLs) received from clients.

Another parameter is clustering of queries. D-clients tend to make queries close to the beginning of each interval. This can be easily verified after extracting the interval value from the periodicity analysis mentioned above.

Yet another parameter is transaction size. I-clients tend to be smaller in terms of return HTTP payload size compared to D-clients.

Still another parameter is subscription duration. I-clients leverage more on (typically web-socket based) event subscriptions. The subscriptions are typically short-lived. They are either closed actively by clients (GUI page switching) or time out due to user inactivity. Meanwhile, D-clients either do not use subscriptions at all or they refresh the subscription periodically to maintain longer subscriptions.

Another parameter is Inter-Request Gap (IRG). This is a time gap between an API request and its subsequent API request from the same source. The inter-request gap and its mean and standard deviation value may be useful parameters in client type classification.

A final example parameter is the rate-limiting reinforcement response. Even before the classification result is obtained to decide whether to perform rate-limiting, rate-limiting may be attempted in order to observe how the clients would respond to the induced reinforcement. The D-clients that require rate-limiting may respond to the action better than I-clients. This may be due to I-clients' lower requests rates, and because interactive users tend to immediately make new requests when the previous ones were dropped.

Several different machine learning techniques may be based on findings mentioned above, such as PCA analysis, DT, and HTTP header analysis. The result may be combined with a weighted adjustment to obtain the final result. The DT may be pre-created with sample data and be dynamically updatable as new sample data are available. The DT is one of the most efficient artificial intelligence / machine learning algorithms that can make classifications accomplished in real-time, so policies can be enforced in real-time in a cloud environment.

For DT classification, effective parameters may include transaction size, subscription duration, and IRG mean, IRG standard deviation, request URL length, request body length, etc. A re-enforcement learning extension may be applied to DT by applying rate-limiting and then checking the effect of traffic reduction to confirm the effectiveness as a method of re-enforcement learning. This parameter may be used as node splitting criteria in boosting DT.

Other domain specific methods may also be used, such as HTTP header analysis on user agent and X-Forwarded-For (XFF) fields.

The techniques described herein enable automatically detecting client type and applying API policy enforcement based on the classification result. These techniques may also be used to address cloud service DDoS attack, a common cloud service attack.

In summary, techniques are described for identifying key parameters for classifying interactive clients and daemon clients. Machine learning algorithms based on PCA and enhanced DT are utilized together with HTTP header analysis to accurately classify clients. Rate-limiting effects are monitored as a feedback to perform re-enforcement learning to improve accuracy of the algorithm. The API gateway is one of the most important components to any cloud service and it needs to apply different rate-limiting policies on API requests based on different requirements of different types of clients. Traditional client identification with source IP addresses are no longer suitable when moving a networking infrastructure to a cloud platform.