Anomaly Detection Design

Authors: [Xing Yang](https://github.com/xing-yang)

This document proposes a design for Anomaly Detection.

Background

OpenSDS is aimed at addressing the storage integration challenges of both the cloud native environment and traditional IT environment. As part of this mission, Telemetry and Anomaly Detection are planned for the road map of Capri. In Telemetry, volume metrics will be collected to reflect the state of the storage systems. The next step is to interpret these metrics and figure out if something goes wrong and trigger alerts. Once users are notified with the alerts, they can take actions to fix the problems and keep the storage systems healthy.

Objectives

In this project, a few areas will be identified to do Anomaly Detection to limit the scope of the work. Algorithms on Anomaly Detection will be researched. A few algorithms will be chosen to do experiments in the selected Anomaly Detection areas.

Analysis will be done using open source tools.

Goals

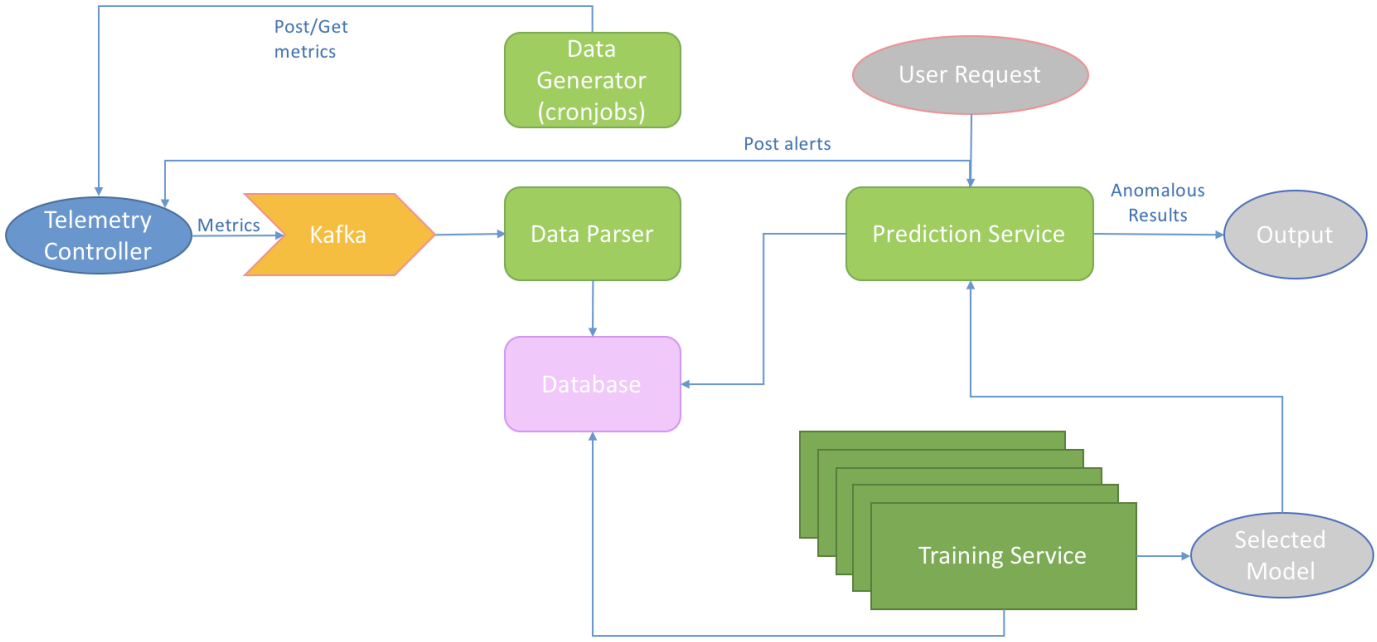
* Analyze data collected from storage and identify anomalous behavior using existing open source tools.
* Integrate with Prometheus alert manager.
* Show alerts in OpenSDS dashboard.

Non Goals

* Manual action is required to handle the alerts. Automatically correcting the problem is not covered in this release.
* Log analysis will be delayed to the next release for trouble shooting problems. Take a look of https://www.fluentd.org for potential integration.
* Kubeflow integration can be investigated in the future.

Architecture Diagram

The following diagram shows the architecture of the Anomaly Detection design and its relationship with the Telemetry project.



As shown above, there are several components in the Anomaly Detect project:

* **Data Generator**: Data Generator runs a cron-job that sends POST API requests to the Telemetry Controller periodically to generate new metrics data such as IOPs, latency, and bandwidth. The Telemetry Controller will in-turn invoke the Metrics driver to collect these requested data. There will also be existing data needed by the ML module that can be retrieved by sending a GET API request.

* **Kafka**: Metrics generated by Telemetry will be sent to Kafka through the Kafka Adapter in Telemetry.

* **Data Parser**: Data Parser retrieves metrics from Kafka, parses it, and saves them to MongoDB.

* **Training Service**: Training Service retrieves data from MongoDB, prepares data, runs several ML algorithms, picks the best algorithm which is called Selected Model, and releases it to Prediction Service.

* **Prediction Service**: Prediction Service is the Anomaly Detection module that runs in production. It picks up Selected Model for each ML model release and uses the newly released model to run prediction. Output is sent back to user after it is ready. The request to run Prediction Service is triggered by user request. Alerts will be sent back to the Telemetry module when anomalous data points are detected.

Training Service

Training Service contains several components:

* Data Preparation is a component that prepares data, i.e., it labels which data point is anomalous and saves it to the database.
* Algorithm Execution is a component that runs algorithms, compares the results, and picks the best algorithm as the Selected Model.
* Training Service uses historical data so it will be a very large data set.

Labeling data manually is a labor intensive task. For performance data, for example, we can look at the range of normal data points published for a particular storage system if it exists and label data points automatically. If there are data points that cannot be marked automatically, they will be sent back to the data scientist to be labeled manually.

Prediction Service

The Prediction Service contains REST APIs that can be called by an user. The Prediction Service will be triggered by user request to perform an analysis. This will use more recent data set which is smaller than what is used by the Training Service. It only uses the Selected Model released by the Training Service to run prediction. After the analysis is done, a report will be generated that contains the results of the prediction, i.e., anomalous data points will be detected.

MongoDB

For simplicity, one MongoDB database will be shared by both the Training Service and the Prediction Service. In the future, we may need two separate databases, one to store long term historical data for training purpose, and the other one to store current short term data for prediction.

Anomaly Detection Algorithms

There are many algorithms used for Anomaly Detection. They are in the following categories [1]:

* Classification based
* Nearest neighbor based
* Clustering based
* Statistical models
  * Gaussian model
  * Regression model
* Information theoretic
* Spectral

Design details

Areas for Anomaly Detection:

Monitor and analyze performance data for abnormal change:
* IOPs
* Latency
* Bandwidth

Performance data will be the focus for Anomaly Detection in the Capri release.

Analyze capacity usage of the system to detect sudden spike and understand the reason behind the change:
* Used capacity

Analyze the duration of an operation and detect sudden increase which is inconsistent with the normal pattern:
* Create volume
* Attach volume
* Detach volume
* Delete volume

Analyze the failure rate of an operation to detect sudden increase to find underlying problems:
* Create volume
* Attach volume
* Delete volume

Algorithms

The following algorithms will be used for Anomaly Detection:

* Gaussian distribution
* MAD
* DBSCAN
* Numenta

Algorithms will be continuously evaluated for its effectiveness and news ones may be used when appropriate.

Open source tools will be leveraged for the analysis.

Gaussian distribution

The Gaussian model is a statistical model that assumes the pattern of the dataset follows the gaussian distribution. A threshold needs to be specified to differentiate between normal and abnormal data points.

Here is an Python-based example we can follow to apply this model:

http://aqibsaeed.github.io/2016-07-17-anomaly-detection/

MAD

MAD is defined as the median absolute deviation from the median.

MAD(D) = median({|di - median(D)|})

Here's a Python library we can use to calculate MAD:

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.mad.html

DBSCAN

DBSCAN refers to Density-Based Spatial Clustering Applications with Noise. Clustering is used to group similar data instances into clusters. DBSCAN is a clustering model designed to discover clusters of arbitrary shape based on density.

The algorithm has two parameters that require user's input:

- ε: The radius of our neighborhoods around a data point p.
- minPts: The minimum number of data points we want in a neighborhood to define a cluster.

Here's a Python library we can use to apply the DBSCAN model:

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

Numenta Anomaly Detection Algorithm

There is also an anomaly detection algorithm provided by Numenta for real-time streaming analytics. Details are as follows:

https://arxiv.org/pdf/1607.02480.pdf
http://nupic.docs.numenta.org/stable/guides/anomaly-detection.html

The above algorithm is available open source in both Python and Golang.

Other Algorithms to Consider

We can take a look of the following algorithms too but need to verify if the license is compatible with Apache2.

Facebook's prophet
https://facebook.github.io/prophet/docs/quick_start.html

Twitter's algorithm
https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2
https://github.com/twitter/AnomalyDetection
This is also an implementation of MAD which is discussed earlier.

References

[1] Anomaly Detection : A Survey
http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf

[2] Density-Based Clustering

https://blog.dominodatalab.com/topology-and-density-based-clustering/

[3] Real-time anomaly detection for streaming analytics
https://arxiv.org/pdf/1607.02480.pdf

[4] Time Series Anomaly Detection
https://storage.googleapis.com/pub-tools-public-publication-data/pdf/dfd834facc9460163438b94d53b36f51bb5ea952.pdf

[5] AIOps: Anomaly detection with Prometheus
https://events.linuxfoundation.org/wp-content/uploads/2017/12/AIOps-Anomaly-Detection-with-Prometheus-Marcel-Hild-Red-Hat.pdf

[6] Detecting outliers and anomalies in realtime at Datadog
https://www.youtube.com/watch?v=mG4ZpEhRKHA