ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Machine Learning and Artificial Intelligence Techniques Using Observability Data in Distributed Systems

Amreth Chandrasehar

Director of Engineering - ML, Infrastructure, SRE and Observability, AWS Community Builder San Jose, California, United States

ABSTRACT

Organizations depends on Observability data to reliably operate systems and develop applications. The amount of data generated in the systems exponentially grows with business demand with new and existing customers, more products and features being released and because developers and operators require more data to analyze and debug applications. As the data grows, the complexity, mean time to detect (MTTD) and mean time to resolve (MTTR) grows as well. Using Machine Learning (ML) and Artificial Intelligence (AI) techniques discussed in this paper will improve MTTR, MTTD, reduce complexity of debugging issues using Observability data collected from the various distributed systems.

***Corresponding author**

Amreth Chandrasehar, Director of Engineering - ML, Infrastructure, SRE and Observability, AWS Community Builder San Jose, California, United States.

Received: September 14, 2022; **Accepted:** October 11, 2022; **Published:** November 07, 2022

Keywords: Machine Learning, Artificial Intelligence, Distributed systems, Observability, SRE

Introduction

Observability is the level of visibility that the system grants to an outside observer. It's a property of a system, just like usability, availability, and scalability. Monitoring is for operating software/ systems Instrumentation is for writing software Observability is for understanding systems Investing in observability means to be prepared to spend the time on instrumenting systems, cope for the unknowns that come in production. It can be very simple at the start, such as some basic health-checks. With metrics, tracing, logging, correlations, structured logging, events; combined together it just brings a really powerful solution. When things may go wrong, Observability helps to react and recover faster.

Organizations usually have different data stores, each for logs, metrics, traces, events using multiple different tools. At times even federation of each tools may be challenging causing one massive monolith cluster having a lot of this data. This can be challenging for correlating data across different data sources with different formats and also for Data Scientists to create a model data to use in ML model development. This paper discusses how to solve these challenges, improve visibility and build successful AI and ML models using Observability data.

Introduction to Machine Learning Models

A machine learning model is a statistical model that is trained on data to make predictions or decisions. Machine learning models are used in a wide variety of applications, such as Predicting customer behavior, Fraud detection, medical diagnosis, Self-driving cars

There are two main types of machine learning models: supervised learning and unsupervised learning.

- **Supervised Learning:** In supervised learning, the model is trained on data that has been labeled, meaning that the output for each input is known. For example, a supervised learning model could be trained on a dataset of images of cats and dogs, with each image labeled as either a cat or a dog.
- **Unsupervised Learning:** In unsupervised learning, the model is trained on data that has not been labeled. The model must learn to find patterns in the data without any guidance. For example, an unsupervised learning model could be trained on a dataset of customer purchase history, with the goal of finding groups of customers with similar purchasing habits.

Machine learning models can be used to analyze observability data to improve the reliability, performance, and security of systems. Below are the use cases of machine learning models using observability data:

- **Anomaly Detection:** Machine learning models can be used to detect anomalies in observability data, such as sudden spikes in CPU usage or memory usage. This can help to identify potential problems before they cause outages or performance problems.
- **Root Cause Analysis:** Machine learning models can be used to identify the root cause of problems using observability data to speed up the troubleshooting process and prevent problems from recurring.
- **Predictive Maintenance:** ML models can predict when equipment is likely to fail. This can help to schedule maintenance before problems occur, which can prevent outages and downtime.

- **Fraud Detection:** ML models can be used to detect fraud by analyzing data to protect systems from malicious activity.
- **Security Monitoring:** Machine learning models can monitor systems for security threats using observability data to identify and respond to threats quickly.

Machine learning models developed using observability data can be challenging. Below are some of the challenges while building ML models:

- **Data Quality:** The quality of the observability data is critical for the accuracy of machine learning models. The data scientists must clean and remove any errors before training ML models.
- **Data Quantity:** Machine learning models require large amounts of data to train. This is a huge challenge for large enterprises, distributed system-based applications that generate lot of data.
- **Model Selection:** There are many different machine learning models available, and it can be difficult to choose the right model for the task at hand. ML Engineers spend a lot of time on choosing the right models, Auto-ML can be helpful for model developers.
- **Model Tuning:** The parameters of machine learning models need to be tuned to achieve the best accuracy. It can be a time-consuming process due to large unstructured and distributed data.
- **Explainability:** It can be difficult to explain how machine learning models make predictions. This can make it difficult to trust the predictions and to use the models to make decisions.
- **Bias:** Machine learning models can be biased, meaning that they may not be accurate for all populations. This can cause huge issues to organizations in scenarios such as recruitment, advertisements, etc.

Despite the challenges, machine learning models can be a powerful tool for improving the reliability, performance, and security of systems. By carefully considering the challenges and selecting the right machine learning model, organizations can use machine learning to improve their observability and make their systems more resilient.

Machine Learning models when rightly developed, it can transform organizations by improving reliability, security, performance efficiency, operational excellence and reduce cost. The diagram below is from AWS Well-architected framework that combines Observability, ML development and aligning organizational goals.

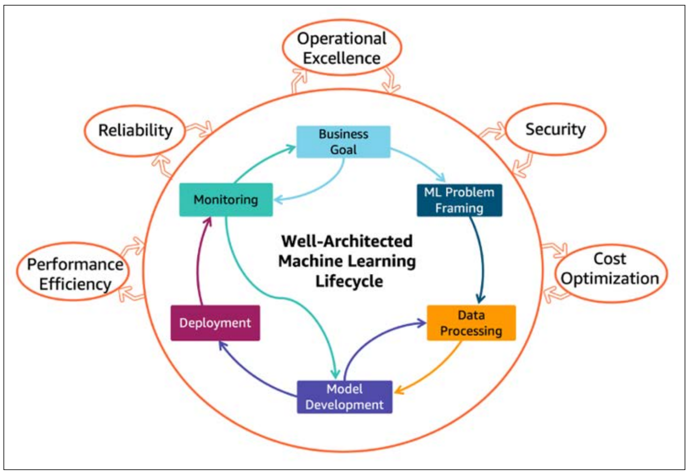


Figure 1: The AWS well-architected ML lifecycle [1]

Observability Data Collection

Observabilities 4 pillars – Logs, Metrics, Traces and Events are collected by agents and sent to a streaming layer. The data is then ingested into a unified Observability platform to process.

Observability standards are very important to standardize the data creation from applications across all services and platforms. This will help developers and operators of applications to easily correlate and have data traceability from UI to the DB layer. It is one of the most important step the organizations need to enforce and keep track frequently. This can also help the Data scientists to clean the data and for the ML engineers to develop ML models.

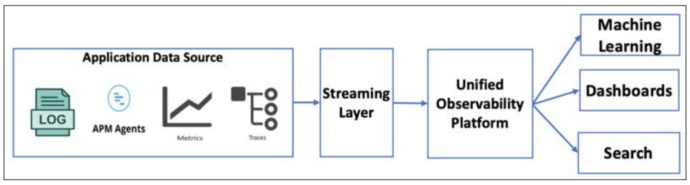


Figure 2: Observability Data Collection

Observability Data Types and Correlations

Observability is based on three main pillars: logging, metrics, and traces.

- **Metrics:** Metrics are quantitative measurements of the state of a system and used to track CPU utilization, memory usage, disk space usage, and response times.
- **Logs:** Logs are textual records of events that occur in a system and used to troubleshoot problems, identify performance bottlenecks, and audit system activity.
- **Traces:** Traces are records of the interactions between different components of a system and used to identify root cause of issues in distributed systems.

The correlations between observability data types can be used to gain a deeper understanding of the system. For example, by correlating metrics and logs, ML models can identify the events that led to a spike in CPU usage. Also by correlating metrics and traces, systems can identify the specific components of a system that are interacting with one and other.

Here are some examples of correlations between observability data types:

- **CPU Usage and Memory Usage:** A sudden spike in CPU usage may be correlated with a sudden increase in memory usage indicates a memory leak or a problem with the application.
- **Log Messages and Metrics:** A log message that indicates an error may be correlated with a metric that shows a drop in performance could indicate that the error is causing the performance problem.
- **Traces and Metrics:** A trace that shows a long-running request may be correlated with a metric that shows high CPU usage indicates that the request is CPU-intensive.

By understanding the correlations between observability data types, a deeper understanding of the system can be gained and to identify potential problems before it causes outages or performance problems.

Below are some tips for identifying correlations between observability data types:

- **Use A Centralized Observability Platform:** A centralized observability platform can help to collect and store

observability data from different sources. This will make it easier to correlate the data.
- **Visualization Tools:** Visualization tools can help to see the relationships between different Observability data types. This can make it easier to identify correlations.
- **Machine Learning:** Machine learning can be used to identify correlations between data. Transaction IDs are smartly added to all traces and during data creation phase in Observability data collection agents.

Using ML Models with Observability Data
Applying Machine learning models on Observability (monitoring, telemetry) data provides actional insights and can be leveraged to proactively surface threats, identify anomalies, identify behavioral trends, accelerate problem resolution.

Anomaly Detection for Trends and Seasonality
Anomaly detection addresses one of the core challenges in monitoring dynamic, responsive, ever-scaling infrastructure: How to define normal versus abnormal performance. Setting static thresholds often leads to false alarms due to normal variations in key metrics like website traffic and customer checkouts, which tend to rise and fall depending on the time of day, day of the week, or day of the month. Anomaly detection accounts for those expected variations, as well as long-term trends, to intelligently flag behavior that is truly unexpected. Datadog's anomaly detection algorithms are rooted in established statistical models but have been heavily adapted for the domain of high-scale infrastructure and application monitoring.

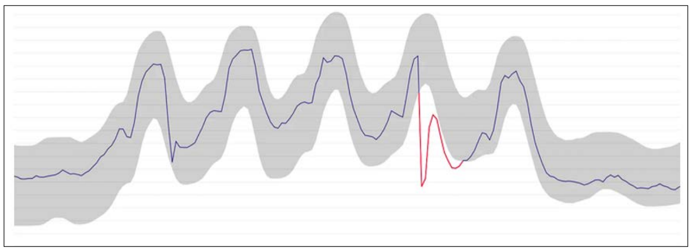


Figure 3: Sample Anomaly detection graph

Outlier Detection for Monitoring Large, Dynamic Fleets
Monitoring large fleets of servers, containers, IoT devices, or application instances makes it difficult to keep tabs on the health and performance of any individual member of the fleet. Datadog's outlier detection algorithms constantly evaluate large fleets or groups to identify if any member of the fleet starts behaving abnormally, as compared to its peers. Using outlier detection, engineers can automatically identify unhealthy application servers, databases, or other systems in need of maintenance, without having to define ahead of time what normal, healthy behavior looks like.

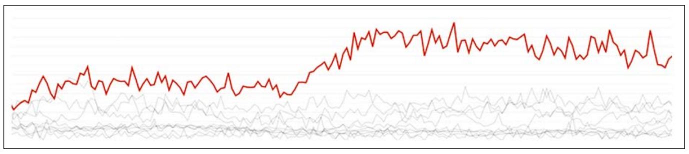


Figure 4: Sample Outliner Detection ML Model Graph

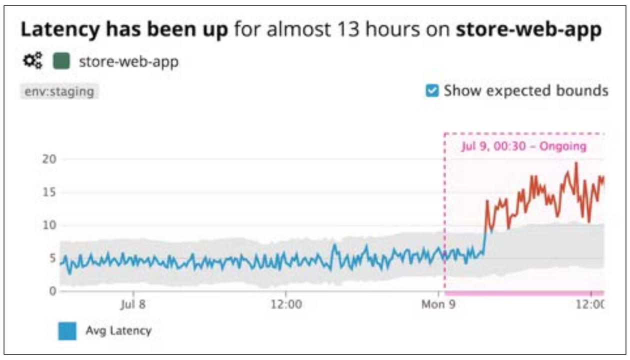


Figure 5: Store Web-App Latency Anomaly Detection

Forecasting to Prevent Bottlenecks
Even in dynamic systems, some limits are fixed, and breaching them can have severe consequences. When an application runs out of memory, or a database server runs out of disk space, the resulting crash can trigger a cascading failure and cause a user-facing outage. For resource constraints such as these, forecasting algorithms can be used to alert engineering teams with sufficient time to address the problem and avoid issues altogether. For instance, forecast alerts can notify teams a week before disk space is predicted to run out, based on recent trends and seasonal patterns in that system's disk usage. With Webhooks integration and monitoring APIs, teams can build automated AIOps (artificial intelligence for IT operations) workflows, such as archiving or deleting logs to reclaim disk space or provisioning more instances of an application to reduce the memory pressure on app servers.

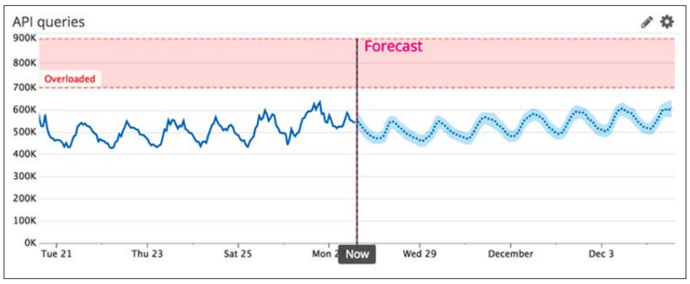


Figure 6: Sample Forecasting Graph

Root Cause Analysis for Immediate Diagnosis
When a system goes down or is affected by an unexpected issue, it can take hours to find the root cause of the problem and fix it, leading to prolonged service interruption and possibly loss of revenue. Watchdog RCA automatically detects anomalous behavior from across your applications and infrastructure, identifies the causal relationships among different symptoms, and clearly pinpoints the root cause. This approach enables you to resolve problems anywhere in your stack faster than ever, significantly reducing your mean time to resolution.

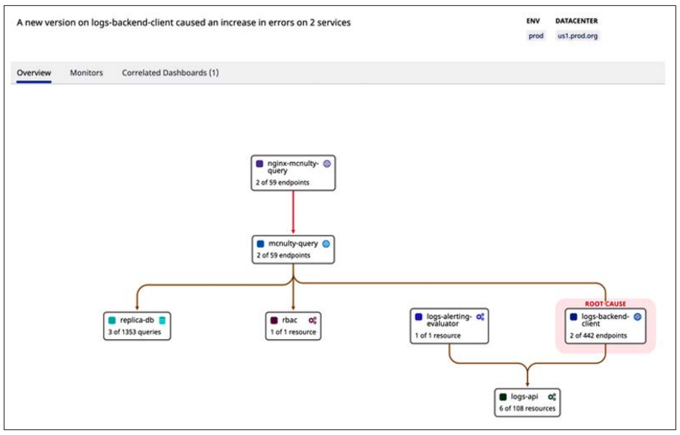


Figure 7: Sample automated root cause analysis by ML model using monitoring data

End to End Machine Learning Platform on Observability Data

The end-to-end Machine Learning on Observability data contains data collection, data transformation, data storage, mode inferencing, action and resolution components. Below diagram shows the flow graph of data generation to an action taken to resolve an issue.

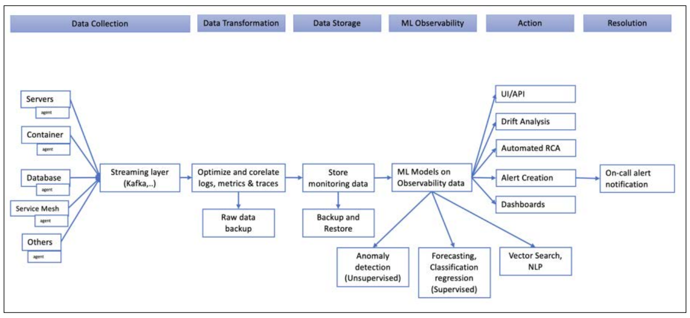


Figure 8: End to 2nd ML Observability Platform

Various machine learning models can be used on observability data such as XGBoost (Extreme Gradient Boosting), Recurrent Neural Network (RNN), Auto-regressive integrated moving average (ARIMA), Support Vector Regression (SVR), Multi-layer Perceptron (MLP), and Long Short-Term Memory Recurrent Neural Network (LSTM RNN) for predictions. In next section, using SARIMA model, CPU utilization of database will be used based on time series metrics.

Applying ML on CPU Utilization Time Series Data to Forecast DB Usage

Inefficient allocation of CPU resources and lack of proactive monitoring can lead to service failures and suboptimal resource utilization in RDS instances across different regions. There is a need for an advanced Time Series model that can accurately forecast CPU utilization, provide real-time insights, and enable proactive resource allocation to prevent service disruptions and optimize performance.

Implementing this CPU Utilization Forecasting model will result in optimized resource allocation, reduced service failures, and improved performance. The real-time insights provided by the model will allow teams to monitor CPU utilization and identify instances that are either overutilized or underutilized for longer durations. This information will enable them to optimize resource allocation, maintaining high performance levels while ensuring a buffer for sudden increases in CPU utilization. Ultimately, this proactive approach will prevent service failures, enhance resource optimization, and improve overall service availability and performance.

The model will consist of three pipelines: a data collection pipeline, a static threshold pipeline, and a forecasting model pipeline. The data collection pipeline will run every 30 minutes, collecting and storing CPU utilization data for all RDS instances in an S3 bucket. The static threshold pipeline will run every 30 days to update the static threshold for all instances. Finally, the forecasting model pipeline will run every hour, predicting CPU utilization for the next 30 minutes for all RDS instances. In the forecasting process, the model will compare the predicted CPU utilization with the static threshold if it is available for the respective instance. For instances without a static threshold, the model will compare the forecasted value with a predefined threshold of 80%. If the predicted value exceeds the threshold, an alert will be triggered for the respective team, allowing them to take corrective measures and proactively reallocate CPU resources.

Below pipeline can be run on MLOps tool to first calculate dynamic threshold and also to forecast database failures based on CPU usage data [2, 3].

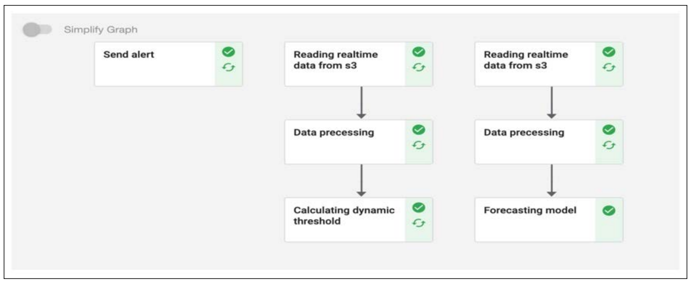

Simplify Graph
Send alert
Reading realtime data from s3
Data precessing
Calculating dynamic threshold
Reading realtime data from s3
Data precessing
Forecasting model

Figure 9: ML pipeline to calculate dynamic threshold and forecast DB failure

Results

The model is successfully able to forecast failures based on CPU usage and DB connections. The graph below shows the DB model almost predicting the failure points. An alert can be set to prevent database failures.

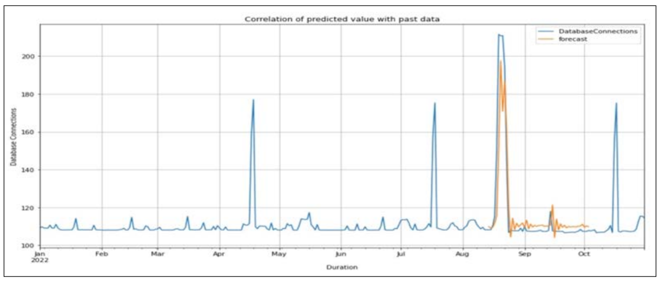
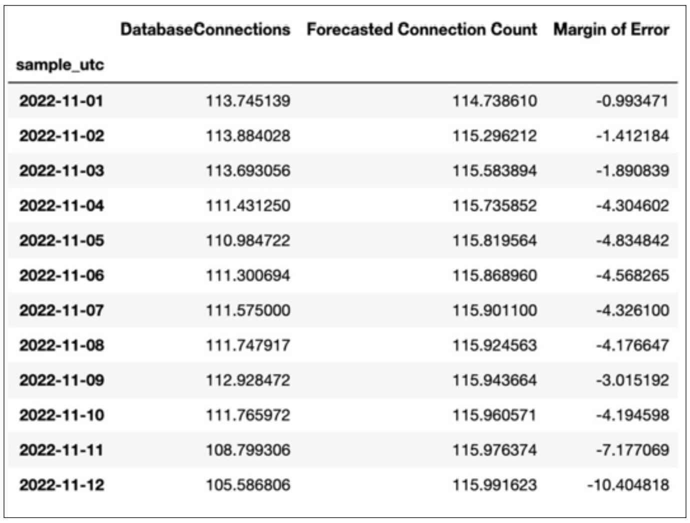


Figure 10: SARIMA ML Model Predicting DB Connections

Results in table below shows the value of forecasted connection count and the actual database connections during the model run.



sample_utc	DatabaseConnections	Forecasted Connection Count	Margin of Error
2022-11-01	113.745139	114.738610	-0.993471
2022-11-02	113.884028	115.296212	-1.412184
2022-11-03	113.693056	115.583894	-1.890839
2022-11-04	111.431250	115.735852	-4.304602
2022-11-05	110.984722	115.819564	-4.834842
2022-11-06	111.300694	115.868960	-4.568265
2022-11-07	111.575000	115.901100	-4.326100
2022-11-08	111.747917	115.924563	-4.176647
2022-11-09	112.928472	115.943664	-3.015192
2022-11-10	111.765972	115.960571	-4.194598
2022-11-11	108.799306	115.976374	-7.177069
2022-11-12	105.586806	115.991623	-10.404818

Figure 10: DB connections

Implications

The models have successfully predicted the failures using Observability data. This is significant as teams can proactively get alerted and take counter measures to protect the database and the application from failures. Database failures can be too common due to improper sizing, long running jobs, logs taking up disk storage, etc. Having ML models to predict these failures can prevent customers being impacted.

Conclusion

The paper provides various ML techniques to be applied on Observability data and also a practical example of using SARIMA model to predict DB failures. Organizations generate a lot of Observability data, but very few proactively put the data to use by applying Machine Learning models. ML models can help teams to predict failures to efficiently run operations as shown in DB forecasting example. The MTTR, MTTD can be vastly improve with these models and also reducing outages in the first place. The limitation of the model is, they are required to be retrained frequently to ensure the dynamic limits set in place are updated with changing metrics every minute. The model needs to be performant, run quickly at less cost, this can take several weeks to fine tune the models to run at scale, especially for 1000s of databases at large enterprise organizations.

There are various tools like Elasticsearch, Datadog that provides ML capabilities for non-ML engineers to quickly configure with built-in features. As a future direction, AIOps can also be used with Observability data or alerts created from Observability tools to correlate data across various tool and ML should be part of every organization's strategy to efficiently operate teams, incidents and innovate to be successful.

References

1. Machine Learning Lens (2023) AWS. https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/machine-learning-lens.html.
2. Machine Learning. Data Dog. https://www.datadoghq.com/solutions/machine-learning/.
3. Rinkal Jaina, Minal Rohit, Anand Kumar, Ayush Bakliwal, Ashwinkumar Makwana, et al. (2022) IJERT 11: 58-64.