Enhancing Structural Health Monitoring and Management Through Edge, Fog and Cloud Computing Architectures

VENKAT SURENDAR TALARI, VISVESH NARAHARISETTY, PRAFULLA KALAPATAPU
and VENKATA DILIP KUMAR PASUPULETI

ABSTRACT

The aim of Structural Health Monitoring (SHM) is to detect, locate, and quantify structural damage through data acquisition, which not only enhances human safety and minimizes infrastructure maintenance costs but can also be performed in real-time. However, environmental factors and operational risks can limit the effectiveness of SHM techniques. Fortunately, the emergence of smart connected devices and internet connectivity has enabled remote monitoring of various structures such as buildings and bridges from any location and at any time. Despite numerous advancements in Internet of Things (IoT) technology in Structural Health Monitoring (SHM), it still struggles with latency issues when analyzing and visualizing real-time data from various structures. Current cloud-based architectures are not efficient in displaying various structure metrics, such as acceleration, humidity, and temperature. Additionally, utilizing state-of-the-art machine learning techniques for predictions is time-consuming as data must travel long distances to reach the cloud. To address these challenges, emerging technologies like edge computing and fog computing can be implemented. Edge computing aims to bring processing and storage as close as possible to the application, while fog computing complements cloud computing by handling fewer intensive analytics and processing tasks. The proposed work showcases a multilayer system architecture comprising edge, fog, and cloud computing, designed to collect, analyze, and visualize sensor data via Amazon Web Services (AWS). This approach involves connecting multiple sensors to an edge device (Raspberry Pi) to develop a standalone web monitoring interface. Furthermore, machine learning algorithms are deployed on the edge to predict the local behavior of a structure using its local data. The methodology of monitoring structures using the architecture presented in this paper exhibits great potential. The effectiveness of the current approach is demonstrated through a comparative analysis of its performance and latency. Besides the software could potentially be packaged and offered as Software as a Service (SaaS) product in the future.

Venkat Surendar Talari, Research Assistant, Email: t.venkatsurendar@gmail.com. Center for Sustainable Infrastructure and Systems, Ecole Centrale School of Engineering, Mahindra University, Hyderabad, India
Visvesh Naraharisetty, Research Assistant, Center for Sustainable Infrastructure and Systems, Ecole Centrale School of Engineering, Mahindra University, Hyderabad, India

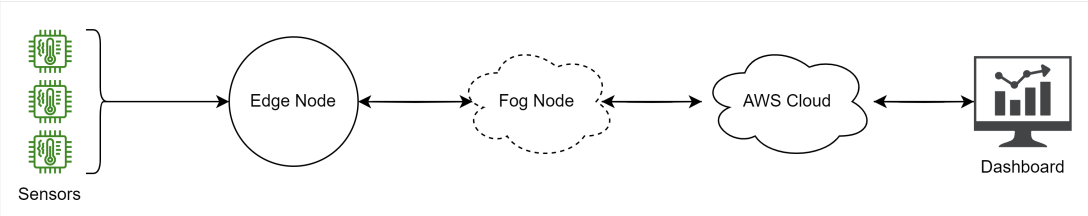

Figure 1. Overview of Architecture

INTRODUCTION AND BACKGROUND

Structural Health Monitoring (SHM) is the process of continuously monitoring structures to detect any damage or degradation, and to assess the overall condition of the structure. It is a highly effective system that helps detect structural damage early on, improves safety, and reduces maintenance costs. [1] This process involves using sensors, data acquisition systems, and analysis tools to collect and study data about how the structure responds to different loads and environmental conditions. SHM systems generate a substantial amount of data, and analysing and managing that data can be quite a challenge. It is required to extract meaningful data to identify patterns and trends that indicate damage or degradation, hence advanced algorithms are used to extract meaningful information from the data. [2] SHM is increasingly using machine learning (ML) to analyse massive amounts of sensor data and detect anomalies or potential failures in structures. The development of ML-based SHM applications has been made possible by edge computing in recent years. [3] Data is processed and analysed closer to the source, often at the edge of the network, rather than being sent to a centralised cloud or data acquisition system (DAQ). Amazon Web Services (AWS) is one of the most popular cloud platforms for edge computing.

By utilizing these advanced technologies, the entire system can be packaged as a Software as a Service (SaaS) and provides a user-friendly as well as adaptable solution for implementing SHM applications. With this, SHM professionals can improve the efficiency and accuracy of monitoring and analysis processes, ultimately improving the outcome of various structures and infrastructure projects. As a result, this solution is flexible, scalable, and efficient.

IOT AND CLOUD ARCHITECTURE

Overview

Edge, Fog, and Cloud computing form an interconnected network that facilitates distributed computing and data processing as depicted in Figure 1 [4]. By creating a harmonious architecture, the system is capable of making decisions in real time, thus enhancing its scale, capability and ability to handle failure gracefully. A structured framework is created which optimises the allocation of computing and storage resources, taking into account the particular needs of each application and its attributes.

Edge Computing

In the field of Structural Health Monitoring (SHM), edge computing has emerged as a promising solution for real-time monitoring and analysis of structures. Edge computing [5] refers to the practice of bringing computing power closer to the source location. Instead of relying on far-away data sources, edge computing puts small but smart devices, like sensors or gateways, right at the edge of the network. These devices collect and process data right where it's generated, helping to make quick decisions without sending everything to the cloud. []

By deploying sensors at critical locations, SHM systems can sense the behaviour and health of structures. However, processing and analyzing this data in real-time can be a significant challenge, especially for large-scale structures.

AWS offers a range of services that can help overcome this challenge. Here AWS IoT Greengrass allows for the deployment of custom code to edge devices, enabling real-time compute environments for data processing and analysis. The Greengrass framework consists of a core component that runs on edge devices, and a cloud component that provides management and deployment capabilities. The core component includes a local message broker, which enables devices to communicate with each other, and a run-time environment such as running Lambda functions. The core component can also store data locally and provide access to AWS services through the AWS IoT Device SDK. ML models are implemented on node edge devices as one of their primary functions. These models are used for analyzing sensor data and extracting insights. It is possible to make real-time decisions and analyses based on ML models by executing them locally without having to send data to a centralized cloud or server. It improves response times and reduces latency, thereby allowing networks to utilize resources more efficiently.

Fog Computing

Fog computing is a computing paradigm that is useful when there are more sensors interconnected in which the fog layer helps as intermediate between edges. [6]It aims to overcome some of the challenges associated with cloud computing, such as high latency and bandwidth limitations. This is done by bringing computing resources closer to the source of data. But there are also few trade-offs for the system such as a lot of interconnections between devices which may arise a chance of getting data vulnerable to attacks while flowing and also makes practical implementation a bit complex [7]. However taking good measures can reduce drawbacks mentioned above and help in efficient transfer and processing of data. Fog computing can also help address some challenges associated with sensor placement and data interpretation.

One of the practical applications in Figure 2 used in smart buildings where data collected from one edge is used in taking decisions and perform actions on a nearby another edge.

Cloud Architecture

In Figure 3, The work focuses on a basic AWS cloud architecture [8, 9]that involves the transfer of data from sensors to a device at the edge, which has Greengrass software. The software integrates seamlessly with the AWS cloud services. Even in low-

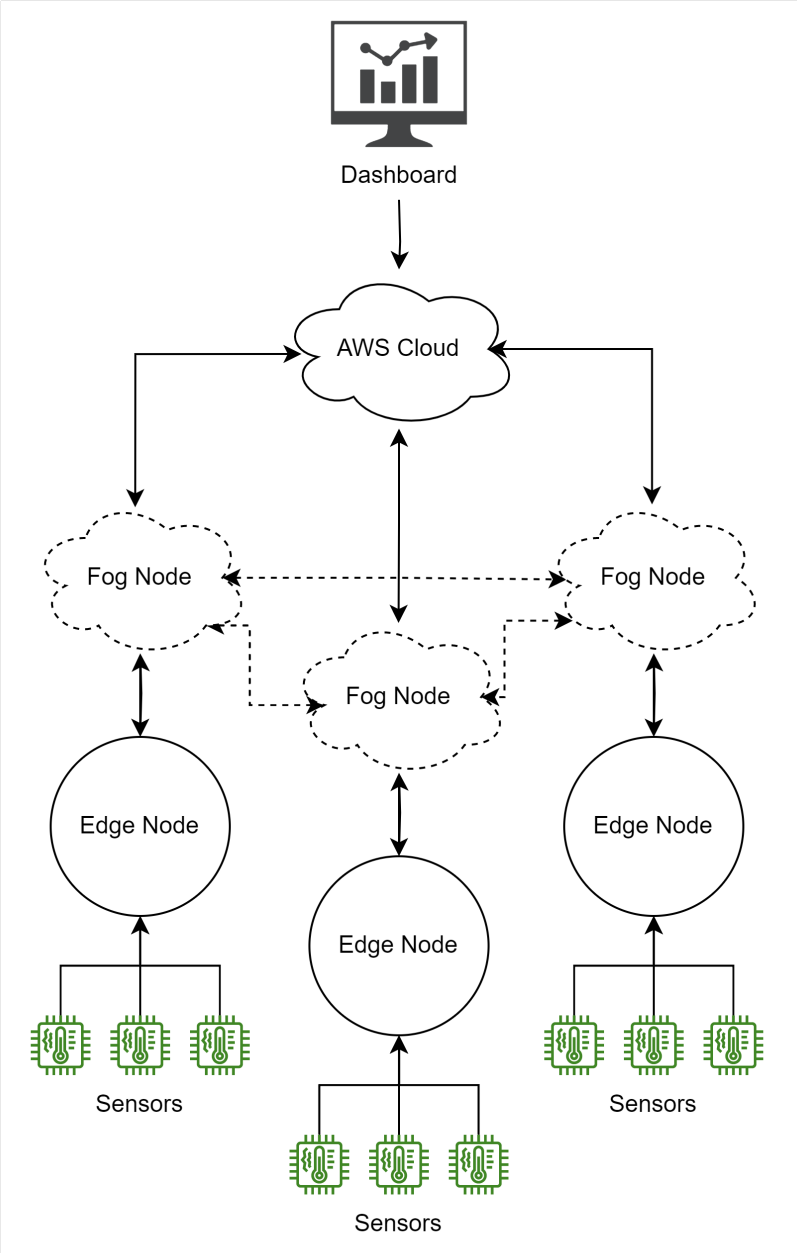

Figure 2. Fog Computing Architecture

connectivity or offline scenarios, Greengrass provides a lightweight message broker that makes it possible for devices to exchange messages locally. After receiving data, it is processed by the IoT Core service, which sends the data to the appropriate AWS location for immediate processing or storage. In addition, the AWS Lambda function is used to generate notifications based on specific events or conditions, and Simple Notification Service (SNS) can be set up to inform subscribed endpoints like email, SMS, or mobile push notifications. This setup allows relevant operators to receive real-time notifications in the case of critical incidents. Furthermore, the collected data is stored in DynamoDB, a solution for storing non-relational data, and it is presented on a dashboard

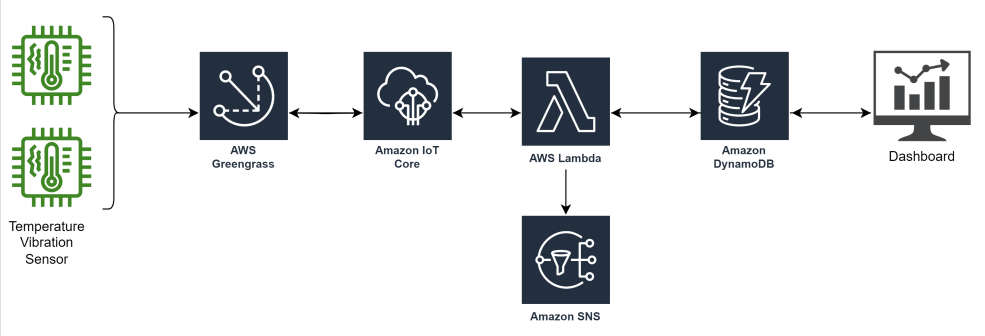

Figure 3. Proposed AWS cloud Architecture

for visualisation purposes.

DATA PROCESSING AND ANALYTICS

A continuous stream of data from various sensors are gathered at a Jetson Nano edge placed at the remote location. Now the data is used as input for the ML model running on the edge device. The result is now sent to AWS S3 target location for storage. This result is used for graphical visualisation.
Once data is acquired from sensors, data is processed and run through inference rules which are formed by a model deployed on run time. Based on the prediction, the system may perform actions such as controlling an actuator or send it to the backend for visualisation and one of the advantages is the entire system is always controllable and configurable at any point of time.

The life cycle of machine learning (ML) at the edge involves four crucial steps that must be followed to ensure successful implementation. Firstly, the process begins with model preparation, which involves selecting and preparing the appropriate dataset, designing and training the ML model, and testing it thoroughly to ensure optimal performance. The second step is model and inference application deployment, which involves deploying the model and the inference application to the edge device where it will be used. This step is crucial because it determines the speed and efficiency of the ML model's inference process. After deployment, the third step is running the inference, which involves utilizing the model to make predictions on new data and evaluating the results. Finally, the fourth step is monitoring the model, which involves continuously monitoring the model's performance and making adjustments as necessary to ensure that it continues to perform optimally. By following these steps, the ML life cycle at the edge can be successfully implemented, providing numerous benefits such as improved performance, reduced latency, and increased privacy and security.

MLOps in monitoring systems helps in building more robust ML models as data to be trained is increasing day by day . Figure 4 explains how MLOps is performed on AWS as follows. Model is developed and staged in Code Commit(1). Once, model is ready, it is pushed to Sagemaker edge manager through code pipeline(2) for building the model. Model is converted to inference rules(3) On greengrass(4), with the help of Sagemaker(5) predictions are formed and results are stored in S3(6). Predicted data is

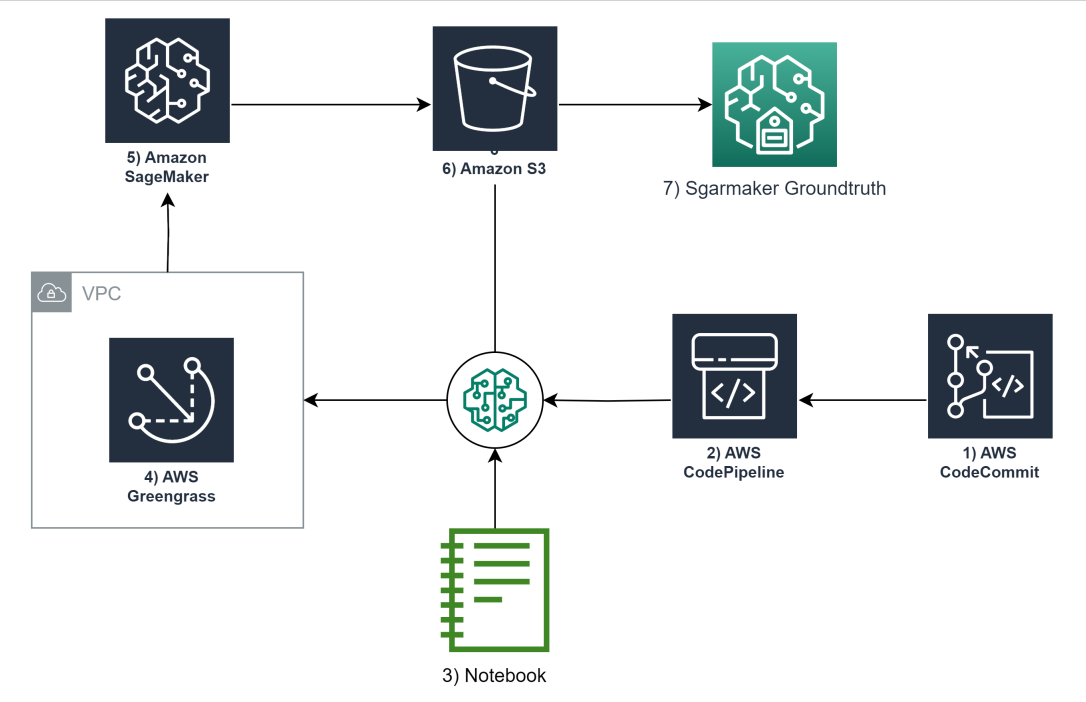

Figure 4. Machine Learning Operations (MLOps) pipeline

sent to Sagemaker Groundtruth(7) and the model is rebuilt . Thus , it completes one MLOps cycle.

Anomaly detection

Isolation Forest is an unsupervised machine learning algorithm that is commonly used for anomaly detection in datasets. [10]It works by randomly selecting features from the dataset and then randomly splitting the selected feature values into two groups, thus creating a tree structure. This process is repeated recursively until each instance is isolated in its own terminal node. The algorithm is especially effective when dealing with large datasets since it does not require any labels or prior knowledge about the data. It is also efficient in its computations, since it does not traverse the entire dataset and only focuses on a few select features for each split. It can also be used for outlier detection, by measuring the path length from each instance to the root node. Instances that have a significantly longer path length than the rest may be considered outliers. Isolation Forest is a powerful tool for anomaly and outlier detection in large datasets. Isolation Forest [11] has several advantages over other anomaly detection methods. It is able to handle high-dimensional data and is efficient at detecting anomalies in large datasets. It also does not require prior knowledge of normal or anomalous instances, making it a useful tool in situations where anomalies may be unknown or constantly changing.

Here, Isolation forest is used in detecting anomalies in stream data of sensors viz accelerometers, temperature sensors, strain gauge sensors.

For the training model , raw data is taken from S3 bucket and the model is developed in sage maker and built into greengrass using sagemaker edge manager . Once deployed

with application ,it imports ML model as a package and uses device data to make predictions and sends data to backend via AWS code commit for retraining purposes thus making the model more robust.

The major advantage of using this kind of MLOps is, it enables working on scalable devices (thousands to millions), deploy /update devices simultaneously.

TRAINING AND RESULTS

In order to train the model, tri-axial accelerometers were used, which include features such as time and z-axis vibration data. The preprocessed data is fed into the Isolation Forest model with parameters viz. N - estimators, max samples, max features.

Isolation Forest measures model performance and accuracy using the Anomaly score. Tree anomaly scores are the mean anomaly score of trees in a forest. A data-set with 3000 rows was used to train this model. Train and test splits are 70% and 30%, respectively. It has an accuracy of about 85%.

CONCLUSION

Vibration-based live monitoring systems play a vital role in Structural Health Monitoring (SHM) by providing continuous remote monitoring of structures. To handle data effectively, a smooth and uninterrupted medium is necessary. This paper presents a method for implementing a live monitoring system that uses a combination of fog computing, edge computing, and cloud computing. Using this approach, predictive analysis can be performed closer to the data source, improving system efficiency and latency. Additionally, this system can be standardized, making it easier to implement across a variety of edge devices. A standardized SHM system can be used in a variety of configurations and environments, enabling it to scale more effectively.

REFERENCES

1. Song, G., C. Wang, and B. Wang. 2017, "Structural health monitoring (SHM) of civil structures," .
2. Li, S., L. D. Xu, and S. Zhao. 2015, "The internet of things: a survey," .
3. Malekloo, A., E. Ozer, M. AlHamaydeh, and M. Girolami. 2022, "Machine learning and structural health monitoring overview with emerging technology and high-dimensional data source highlights," .
4. Martín, C., D. Garrido, L. Llopis, B. Rubio, and M. Díaz. 2022. "Facilitating the monitoring and management of structural health in civil infrastructures with an Edge/Fog/Cloud architecture," *Computer Standards & Interfaces*, 81:103600.
5. Verma, R. K., K. Pattanaik, P. Dissanayake, A. Dammika, H. Buddika, and M. R. Kaloop. 2020. "Damage detection in bridge structures: An edge computing approach," *arXiv preprint arXiv:2008.06724*.
6. Yi, S., C. Li, and Q. Li. 2015. "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, pp. 37–42.

7. Yi, S., Z. Hao, Z. Qin, and Q. Li. 2015. “Fog computing: Platform and applications,” in *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*, IEEE, pp. 73–78.
8. Naraharisetty, V., V. S. Talari, S. Neridu, P. Kalapatapu, and V. D. K. Pasupuleti. 2021. “Cloud architecture for IoT based bridge monitoring applications,” in *2021 International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, IEEE, pp. 39–42.
9. Naraharisetty, V., V. S. Talari, S. Neridu, P. Kalapatapu, and V. D. K. Pasupuleti. 2022. “Proposed Cloud Architecture for Real-Time Bridge Monitoring Using IOT,” in *European Workshop on Structural Health Monitoring: EWSHM 2022-Volume 2*, Springer, pp. 195–203.
10. Goldstein, M. and S. Uchida. 2016. “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” *PloS one*, 11(4):e0152173.
11. Xu, D., Y. Wang, Y. Meng, and Z. Zhang. 2017. “An improved data anomaly detection method based on isolation forest,” in *2017 10th international symposium on computational intelligence and design (ISCID)*, IEEE, vol. 2, pp. 287–291.