Utilization of Edge Computing to Improve the Lambda Architecture

Jonathan Lwowski
Department of Electrical and
Computer Engineering
The University of Texas at San Antonio
San Antonio, TX 78249

***Abstract*—Cloud computing has been used and studied for many years now. Less frequently researched is the use of cloud and edge computing to optimize the performance of a system as a whole. One important aspect of cloud and edge computing is to easily and compactly distribute the nodes of system on the cloud, edge and end devices, which accomplished with containerization. This results in a lightweight and compact applications for each of the algorithms in the system. These applications, that are hosted in Linux containers, and can easily and quickly be deployed on any layer of the cloud and edge computing architectures. Container management software, such as Kubernetes, is employed to manage these containerized applications and provide them with fault-tolerant and scalable management. In this research, a cloud-edge model is developed to improve the performance of a cloud-based temperature monitoring application by integrating edge computing into the Lambda architecture.**

***Index Terms*—cloud computing, edge computing, cloud-edge model, container orchestration, Kubernetes, containerization, Docker**

I. Introduction

During the last decade, Cloud Computing has provided new architectures to make the processing faster and more efficient. The term "Cloud" comes from when the user leverages the power of remote machines to do computations. By the year 2025, scientists have estimated that there will be one trillion IoT devices which are interconnected to each other, impacting the economics of approximately eleven trillion dollars per year. A recent study on IoT device networks has addressed some issues such as scalability, network processing, and latency in communications. As an extension of Cloud Services, Fog Computing [1] is an architecture in which there are three tiers for processing. The first layer is end devices such as sensors, actuators, etc. The second layer which is called "Edge" is a mid-level processor/communicator device such as routers, modems, access points, etc. The third level is Cloud processor which has the highest amount of computational power. The reason to add a mid-level layer to the traditional Cloud Computing map is that in many problems computation on the cloud layer can experience large amount of latency and delay, but the mid-level will not. The mid-level layer can distribute the work loads in an optimum way to the cloud in order to make the computation faster and the total network latency lower. As one of the most paradigms in Cloud/Fog Computing, there has always been the challenge to distribute the load between the edge and cloud levels. The edge level [2] which can be considered close to the end device can implement simple processing on the received data from the end device with lowest delay, but the computational power can be low. This is why in most Fog-Cloud scenarios, the edge level is being used for data pre-processing and filtering. On the other hand, the Cloud level which has the most computation power can be utilized for more complex computations. As one of the solutions for the paradigms of computation/network latency, Lambda Architecture was introduced for data-processing to manage the huge amount of data by dividing them into Speed Layer, Batch Layer, and Serving Layer for balancing the latency and fault-tolerance [3] of data. The batch layer can compute the results of the distributed data for increasing the accuracy. Speed layer processes data in a limited way, since it has the minimum latency and is limited in computation power. The serving layer can store the output from both speed and batch layers. In this paper, we propose a Lambda-Edge Architecture which places the speed layer on the edge device and modified the serving layer to minimize the latency issue in the traditional Lambda architecture. The end device for implementing this architecture is a temperature sensor, the edge layer is a router, and the Cloud layer is the main processor. Also, there is a time-series prediction layer using ARIMA [4] forecasting models in this architecture located in batch layer, which is used to query data and predict the the future data for evaluation.

* This work was supported by Grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD through a contract with North Carolina Agricultural and Technical State University.

II. Proposed Cloud Architecture

A. Lambda Architecture

The original Lambda Architecture, seen in Figure 1, consists of three layers; the batch layer, the speed layer, and the serving layer. The batch layer is designed to perform calculations on large quantities of data. These batch processes usually are very time consuming, and for many real-time systems, these calculations are too slow. To address this problem, the speed layer is used to perform calculations on smaller quantities of data. These algorithms will operate much faster, but only operate on a small subset of the data. Therefore the speed

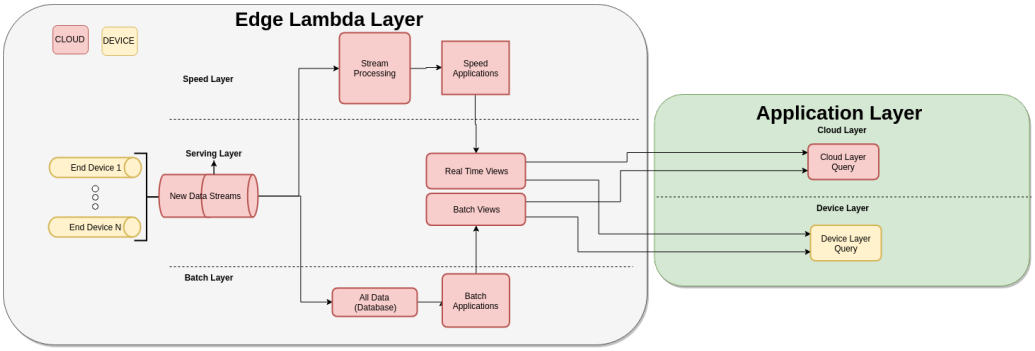

Fig. 1: Original Lambda Architecture

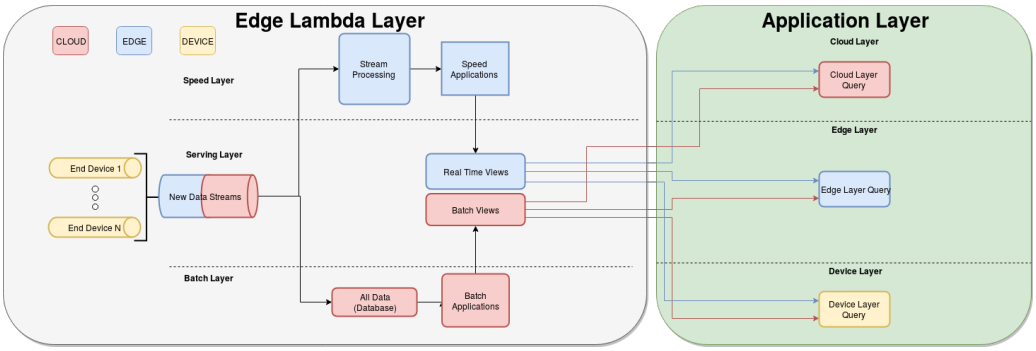

Fig. 2: Modified Lambda Architecture

layer may be less accurate but with the benefit of speed. The combination of the batch and speed layers can allow some real-time systems to have a good balance between speed and accuracy which is usually a drawback to traditional cloud computing. The last layer, the serving layer, is where the outputs of the speed and batch layers are stored, and can easily be queried by the end devices.

B. Lambda-Edge Architecture

Our proposed cloud architecture is an extensions of the original Lambda Architecture. One major change, is that the speed layer is no longer hosted on the cloud. Instead, as seen in Figure 2, the speed layer is moved to an edge device such as a router. The benefit of moving the speed layer to the edge device is to decrease the networking latency allowing, because the latency between end devices and a router in much smaller than the end devices and the cloud. Another major change to the original Lambda Architecture is modifying the serving layer to exist across all three layers of cloud-edge model. By allowing queries in the serving layer to be hosted on the cloud, edge, or end device, the performance of these optimized by placing the containers in an optimal layer of this model.

III. Experimental Results

To test the effectiveness of our proposed Edge-Lambda model, we have utilized temperature monitoring and prediction as an application. The temperature monitoring application is designed to show the user the last 100 temperatures along with a prediction, seen in Section III-A, of the future temperatures.

A. Temperature Prediction Algorithm

In this paper, the prediction model for forecasting the future temperatures is an Autoregressive Integrated Moving Average (ARIMA) model in the batch layer. Using a set of time-series information for training and another set for evaluation, this model has two sets of parameters. The first ones (p) are the order of the system which represent the time-lag for the autoregressive model, the second one (d) is the degree of differenciating, and the last one (q) represents the order of moving-average model. Figure 5 shows a sample of actual temperatures and predicted ones. This samples has been selected from first 1000 recorded datapoints (shown in blue). The training dataset has the 66 percent of the total datapoints and the evaluation/test dataset has the rest 33 percent. The calculated root mean squared error (RMSE) of the test datapoints and the predicted ones is 0.386712 which shows how the prediction model can follow the features of the data accurately.

B. Temperature Monitoring and Prediction using Lambda Architecture

Traditionally in this application, the end device would retrieve the last 100 temperatures, and the predicted temperatures from a database hosted on the cloud using the traditional lambda architecture. As seen in Figure 3, the retrieval of the last 100 temperatures is placed in the speed layer, and the temperature prediction algorithm is place in the batch layer. This allows for the monitoring device to quickly retrieve the past

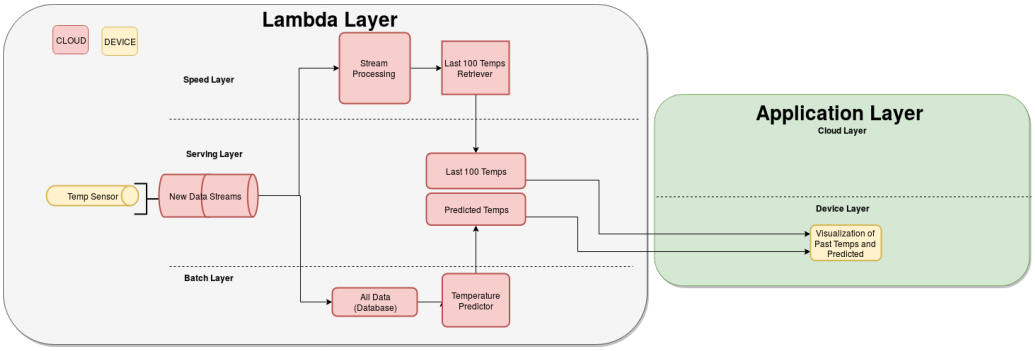

Fig. 3: Original Lambda Architecture for Temperature Monitoring and Prediction

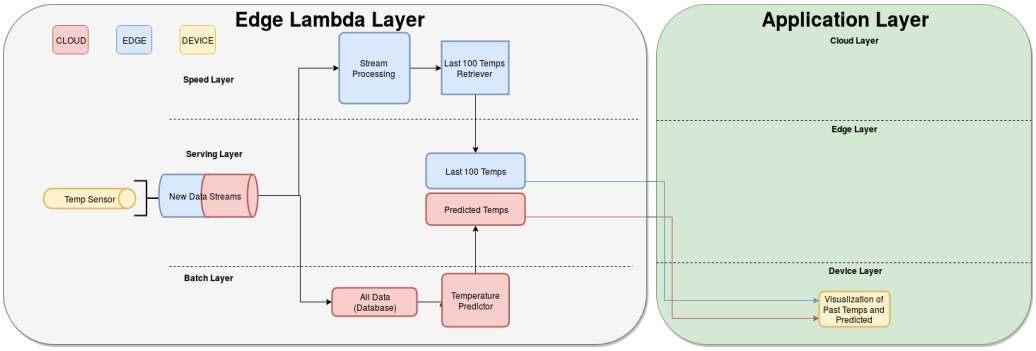

Fig. 4: Edge-Lambda Architecture for Temperature Monitoring and Prediction

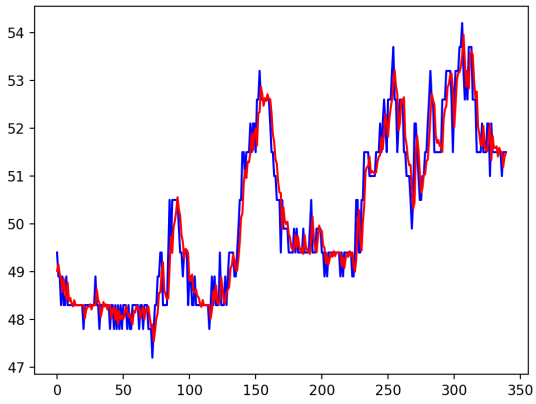

Fig. 5: Output of ARIMA prediction where the red line is the predicted temperature and the blue line is the actual temperature

temperatures, while the prediction algorithm is still running. Once the prediction algorithm is updated, the application and retrieve the most recent predicted temperatures.

C. Temperature Monitoring and Prediction using Edge-Lambda Architecture

The traditional lambda architecture works well, however, it does not make sense for the end device to go all the way to cloud for 100 most recent temperatures if the end device is connected to the same edge device as the temperature sensor. Instead the Edge-Lambda architecture can be used, seen in Figure 4, to store the 100 most recent temperatures on the edge device. Then the temperature monitoring application can retrieve those temperatures much quicker. Similar to the traditional Lambda architecture, the Edge-Lambda architecture also places the retrieval of the last 100 temperatures on the speed layer, and the temperature prediction algorithm on the batch layer. However, the speed layer now exists on the edge device which will make the networking latency of sending the temperatures from the speed layer to the end device much lower.

D. Cloud Architecture Results

To test the effectiveness of using the Edge-Lambda architecture, the temperature monitoring architecture, seen in Figure 4 was implemented into hardware. The end device was a simple temperature sensor on the CPU of a computer, the edge device was a Rasberry Pi 3 with routing software installed on it, and the cloud is a VM hosted in Chameleon. The end device would read the temperature of the CPU and simply publish the current temperature to a RabbitMQ server on the both the cloud and the edge devices. The cloud would save all of the temperatures read to MySQL database and the edge device would only save the last 100 temperatures into a MySQL database. The prediction algorithm seen in Section III-A, is hosted on the cloud. The prediction algorithm will read the temperatures from the cloud MySQL database, predict the future temperatures, and the write the results back to the MySQL database.

A visualization application was also developed to visualize the results of the monitoring and prediction application. This visualization app would read the last 100 temperatures from the edge device's MySQL database, and the current predicted temperatures from the MySQL database on the cloud. As seen in Figure 6, the visualization app successfully able to display both the predicted and actual temperatures verifying that our system works effectively.

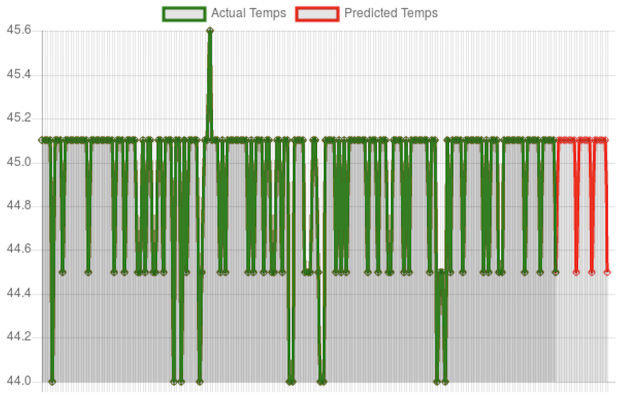

Fig. 6: Output of visualization application

IV. Conclusion

The innovation of Cloud Computing in the most recent years plays an inevitable role in solving complex problems. In this regard, one of the most challenging issues in the traditional Cloud architecture was the trade off between the latency in the communication between end devices and the cloud devices, and the computational power. To solve this issue a new architecture was introduced to divide the whole computation processes into both Cloud and new layer named Edge which can be the mid level in the communication such as router and modems. Fog-Cloud architecture has proposed an architecture using which there three layers in the network (end devices, edge devices, and cloud devices). The new challenge of this new platform is the distribution of the work loads between the edge devices, end devices, and cloud devices in order to minimize the network delay and latency, and maximize the computation processing power. In this paper, using a modified lambda architecture on the Fog-Cloud map of the network, we used a temperature sensor as the end device, the router as the edge device, and the Cloud servers. The purpose of this platform is to implement a real-time temperature predictor using the above-mentioned architecture. To fulfill this objective, we implemented the speed layer in the edge device and the serving layer in the three-layer devices. Applying the time-series Auto-regressive Integrated Moving Average prediction model in the batch layer, enabled us to minimize the network latency and optimize the utilization of computation power in all of the end devices, edge device, and the cloud. At the end, the results of the predictions in the End-device-only, Cloud-only, and Cloud-edge networks have been proposed and are showing that the new platform serves to optimize the challenges of cloud networks in order to minimize the delay and maximize the process power. As the future works in this project, this customized architecture can be used in order to solve more complex problems for distribution the work loads between all layers of the cloud.

References

[1] M. Taneja and A. Davy, "Resource aware placement of iot application modules in fog-cloud computing paradigm," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 1222–1228.

[2] J. Herrera, M. A. Demir, P. Yousefi, J. J. Prevost, and P. Rad, "Distributed edge cloud r-cnn for real time object detection," in *2018 World Automation Congress (WAC)*. IEEE, 2018, pp. 1–5.

[3] P. Yousefi, H. Fekriazgomi, M. A. Demir, J. J. Prevost, and M. Jamshidi, "Data-driven fault detection of un-manned aerial vehicles using supervised learning over cloud networks," in *2018 World Automation Congress (WAC)*. IEEE, 2018, pp. 1–6.

[4] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of arima and artificial neural networks models for stock price prediction," *Journal of Applied Mathematics*, vol. 2014, 2014.