Reference Model for Large Scale Intranet of Things Middleware

Mohamed Musthafa Fathima Naja
Department of Information and Communication Technology
South Eastern University of Sri Lanaka
Oluvil, Sri lanka
mmfnaja@seu.ac.lk

RK Ahmad Rifai Kariapper
Department of Information and Communication Technology
South Eastern University of Sri Lanaka
Oluvil, Sri Lanka
rk@seu.ac.lk

Mohamed Iqbal Isham Mohamed
Kloudynet Technologies
Kula Lumpur, Malaysia
isham@kloudynet.com

***Abstract* — Internet of Things (IoT) have been adopted by industries and enterprises in order to utilize the maximum of the facilities it supports. However, security is the biggest challenge in IoT implementation and deployment due to it residing in public internet. With the aim of addressing this issue, this research work focuses on developing a middleware that could be decoupled with the public internet and leverage big data for large scale enterprises and could be entirely hosted at on-premise intranet. For this purpose, a middle ware model is proposed, and a prototype was developed based on the proposed model and was tested for performance with identified evaluation. The proposed middleware model shall address the security concern of enterprises which uses Internet of Things in their cooperate network.**

Keywords —Enterprise IoT, Apache Middleware Offerings, IoT Hardware

I. Introduction

One of the most trending topics in the field of information technology which has emerged in the recent past is the Internet of Things (IoT). Although the application of IoT in various fields has drawn the attention of many stakeholders, some of the drawbacks and inefficient products used for these application purposes have brought up dissatisfaction too. Since the phrase itself suggests, the IoT is tightly coupled with Internet, by means of hardware such as sensors, devices and servers, middleware such as service buses and message ingesters and software such as enterprise application which are needed to be connected together [1]. Hence, they need to be exposed to the public internet too.

Main problem of IoT deployments and implementations residing on public internet is security. Although it is definitely going to be advantageous, especially for enterprises, most enterprises do not like to expose their IoT deployments to the public internet. The main concern here is the unauthorized access. One best solution to this issue is to deploy it with in their intranet.

The most important component in the IoT architecture is the middleware[2]. In the current IoT architecture, the middle-ware service is provided by cloud vendors such as Microsoft Azure, Amazon Web Service, IBM Bluemix etc. [3]. All these middle-ware offerings are hosted in a public cloud in a multitenancy manner. These cloud vendors normally provide Internet of Things deployments on public internet as it is the nature by the name "Internet of things". Also, the enterprise level IoT implementations are done on public internet, as the middleware which can process big data and which is reliable and fast in processing telemetries and messages are only available in the public internet. Even though Microsoft provide Azure Stack, which enables enterprises to host the Azure cloud services on-premise of their local corporate network, the Azure stack requires a very high-end hardware. On the other hand, Microsoft Azure provides Azure IoT Edge computing, which is basically an Edge computing instance supported by Microservice architecture that could be used in the proposed architecture as it does not totally decouple the IoT deployment from public internet [4]. If the IoT middleware is made to work with intranet connection or corporate network, it would easily address the concern on security and can be easily adopted by enterprises which concern much about security.

II. Objectives

This research focuses on how a middleware can be decoupled with the public internet and can leverage big data for large scale enterprises by developing a IoT middleware architecture which can entirely be hosted at on-premise in the network latency, which means, the intranet, as data does not need to pass through data centers hosted in different regions. The main objective is to build an edge computing architecture that would be highly scalable for enterprises by reusing the existing software. Hence, it will definitely improve the efficiency of the real-time decision making. The outcome is a middle-ware service that runs on linux platform as most of the IoT hardware support Linux[5].

III. Methodology

The proposed reference architecture is specially designed for large scale use like enterprises which focus much about the security concern. Apache NiFi is chosen for Data Ingestion and Device Management. For streaming the message and analytics, Kafka or Spark Cluster is chosen. For the storage purpose, HBase is used. Based on the proposed reference model, a prototype was developed, such that it incorporates the expected features using the existing software and components in Apache.

A. Proposed Middleware Model

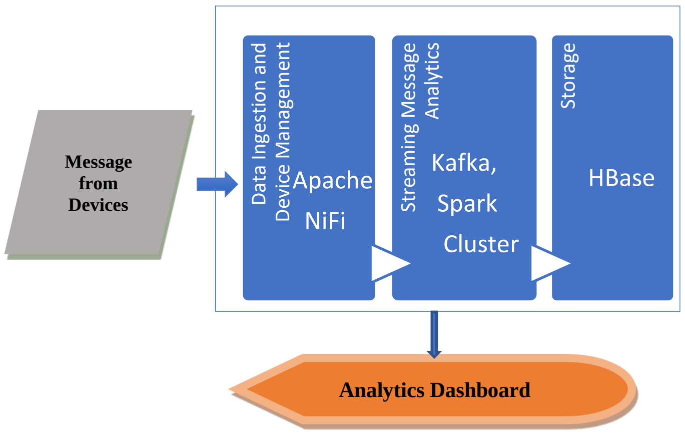

Fig. 1. Reference Model of the Proposed Middleware for Large Scale Intranet of Things

B. Prototype Implementation

a. Data Ingestion and Device Management

Data ingestion, which means receiving data from sensors is the starting point in the architecture. From the beginning onwards, the devices are managed and the next most important is the development of Digital Twins of the devices as it is easier to control all the devices from the software itself rather than physically setting the devices on or off or other properties. When it comes to current IoT offerings from famous vendors, each of them uses such tools like IoTHub (viz. in Microsoft Azure for example, via cloud offering). Since it is built as the on-premise architecture based on Linux, Apache NiFi was chosen. Apache NiFi does not have a dependency on a specific data format, and it has the capability to consume and publish MQTT[1] protocol as well. Since Apache NiFi offers the ability to create custom components, the device twins can be built as custom components [6]. On the other hand, NiFi has a robust authentication support as it uses TLS certificates by default for user authentication which has been configured for Kerberos[2] or LDAP[3] as well.

b. Streaming Message Analytics

Once the messages are ingested to the system, then it is stored. But since the architecture has been built for large scale domains, the number of messages would introduce bigdata to the system. Thus, there should be a way to opt-out which message to store. Apache Kafka is the ideal candidate for the previously mentioned purpose. The reason for choosing Kafka was that it could be used for real time streaming and data-pipeline applications. Kafka provides facility to partition the data into different topics which can be indexed and be read by the consumers. This can be also pipelined to either Apache Spark cluster for near-real-time analytics or can be batch processed.

c. Storage

Once the messages are processed, they are either stored or ignored. This architecture provides a way to store the messages as well using HBase which is schema-less, built for horizontally scalable wide tables and ideal for storing both semi-structured data and structured data.

C. Prototype Evaluation

The prototype developed based on the middleware model was tested for performance with the following metrics and the results were noted accordingly.

a. Number of devices connected concurrently: To evaluate the number of devices connected concurrently, a number of emulated devices with a unique ID were connected to Apache NiFi endpoint and the results was noted.

b. Number of Messages processed concurrently: To evaluate the number of messages processed concurrently, telemetry messages in a given format produced from each device connected to Apache NiFi endpoint was sent through the streaming message analytics service and results on the number of concurrently processed messages was noted.

c. Number of Records saved Concurrently: The telemetry messages thus sent to the streaming message analytics service was checked for whether it is saved to the HBase 3storage and the number of similar messages saved concurrently to the storage was evaluated to find the number of records saved concurrently.

d. Number of Messages Processed in a time window: With the aim of evaluating the performance fo the prototype, number of messages processed in a time window was calculated by identifying the number of messages processed in a given time (60s-time window).

e. Number of Messages successfully saved in a time window: The number of messages saved in the HBase storage was evaluated to find the number of messages saved to it in a given time (60s-time window).

IV. Results and Discussion

The prototype developed based on the proposed reference model was evaluated for expected functionalities. Since the prototype was developed with the aim of implementation at large scale intranet of things environment, the developed prototype was tested for performance based on the identified metrics.

At the first glance, the middleware's performance was tested for the number of devices being connected concurrently with three servers of different specifications and the results shows that all the emulated devices which were connected were running concurrently on all the servers with different specifications like 2GB RAM and 4 Core CPU, 4GB RAM and 8 Core CPU and 8GB RAM and 8 Core CPU.

Fig.2 , Fig.3 and Fig.4 shows the analyzed results of the four performance evaluation metrices as tested with a server of specifications 2 GB RAM and 4 Core CPU , 4GB RAM and 8 Core CPU and 8GB RAM and 8 Core CPU respectively with 100, 1000, 100000 etc. number of messages as the input which is given in the “x” axis of each graph.

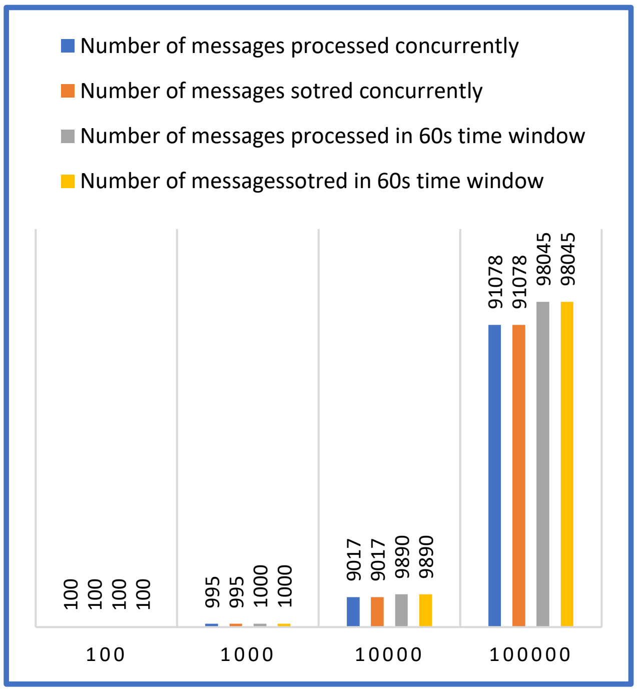

Fig. 2. Performance Evaluation of the Middleware Developed with Server of Specification - 2GB RAM and 4 Core CPU

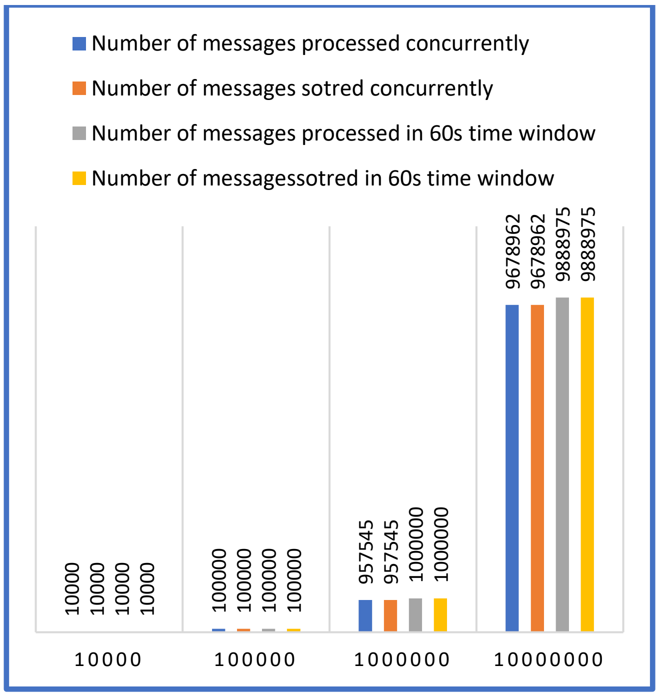

Fig. 3. Performance Evaluation of the Middleware Developed with Server of Specification - 4GB RAM and 8 Core CPU

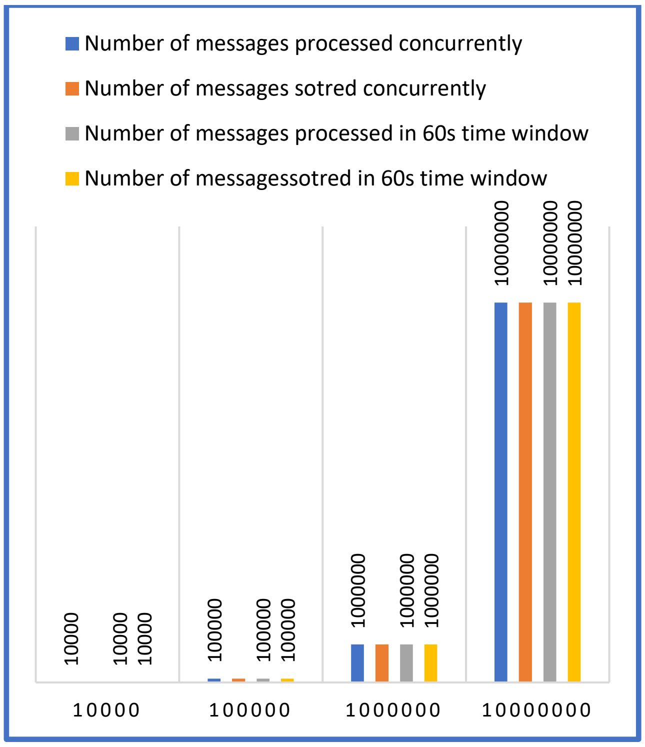

Fig. 4 Performance Evaluation of the Middleware Developed with Server of Specification - 8GB RAM and 8 Core CPU

Based on the results of the perofrmance evaluation, it is obvious that when the specificaton of the server is higher, the expected results for each metric is better compared to the one ith low specifications. Hence, it could be suggested that one way to improve the performance on enterprise babis to process large vloume of messages is to improve the hardware epcifications as needed. Figure 5 shows the graphical reprentation of the comparision of messages processed in a time window by different servers with different hardware specifications. It highlights the fact of improved performance with improved hardware specification.

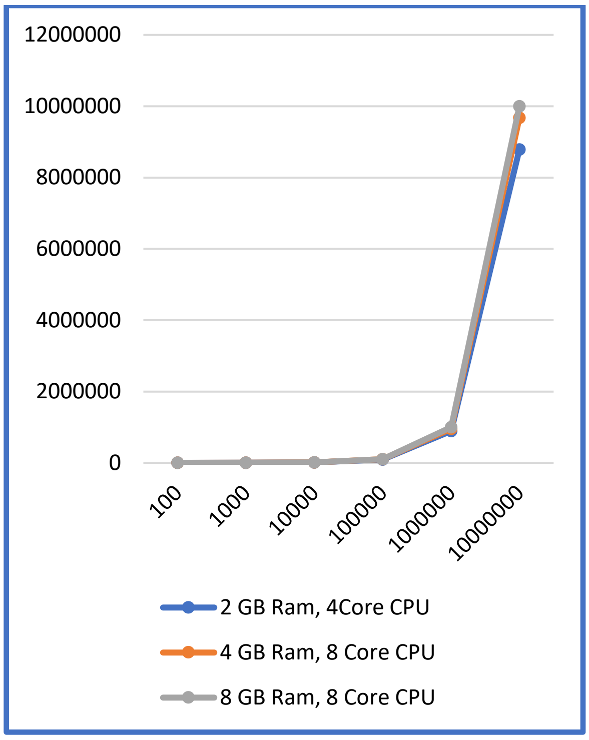

Fig. 5. Graphical Reprensentation of the number of messages proceesd in a time window (60s) by tdifferent servers

V. Conclusion

This research was mainly intended on building an edge computing architecture with the proposed reference model that is highly scalable for enterprises that uses huge amount of data which is said to be "Big Data". The main idea was to reuse existing software for the research instead of reinventing a wheel as it saves time and avoid errors. Hence, Apache was chosen as it provides several middleware offerings that can run on Linux with minimal configuration. The main reason for selecting a middleware service that can run on Linux is that most of the IoT hardware have Linux support at the first place.

References

[1] P. Desai, A. Sheth and P. Anantharam, "Semantic Gateway as a Service architecture for IoT Interoperability". IEEE International Conference on Mobile Services, 2015.

[2] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev and V. H. C. de Albuquerque, "A Reference Model for Internet of Things Middleware," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 871-883, April 2018, doi: 10.1109/JIOT.2018.2796561.

[3] Soumya Kanti Datta, Christian Bonnet and Navid Nikaein, An IoT Gateway Centric Architecture to Provide Novel M2M Services. IEEE World Forum on Internet of Things (WF-IoT), 2014.

[4] Jiehan Zhou ,Teemu Leppanen and Erkki Harjula , CloudThings: A common architecture for integrating the Internet of Things with Cloud Computing. 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2013.

[5] M. Weyrich and C. Ebert, "Reference Architectures for the Internet of Things," in IEEE Software, vol. 33, no. 1, pp. 112-116, Jan.-Feb. 2016, doi: 10.1109/MS.2016.20.

[6] [Available Online]: https://nifi.apache.org/docs.html