POLITECNICO
MILANO 1863

Usability research over the FogFlow framework

Professor : Elisabetta Di Nitto

Teacher assistant : Mehrnoosh Askarpour

-Matteo secco(10529606)

(matteo.secco@mail.polimi.it)

-Mohammad rahbari solout

(10688365)

(Mohammad.rahbari@mail.polimi.it)

POLITECNICO
MILANO 1863

Abstract - **At the heart of smart factories is the issue of intelligence, and spatial intelligence is an important part of these factories : If something moves and the process is very important, you must first see where it happened. And what is it doing ? Industry 4.0 describes an environment in which flexible and intelligent processes use data from sensors connected to all parts of a value chain at the same time as the event to optimize business processes . Nowadays, as the Internet of Things (IoT) provides high-volume , wide - ranging data , it is possible that when data is transferred to the Cloud for analysis, the opportunity to perform any activity on it is eliminated . Fog Computing technology provides Cloud implementation conditions near equipment that generates and operates IoT data . FOGFLOW is a fog computing-based framework for IoT smart city platform which by taking advantages of some features such as dynamically orchestrate IoT services over cloud and edges , in order to reduce internal bandwidth consumption and offer low latency and fast response time . A holding company has several factories and they want to implement "Industry 4.0" on their subgroups, using FOGFLOW framework by adding some devices such as robots and human bracelets.**

I. INTRODUCTION

Collecting massive amounts of data in everyday life poses huge challenges for the user to keep control of his data in terms of managing access , sharing and protection . The industrial process requires most of the tasks to be performed locally because of delay and security requirements and structured data to be communicated over the Internet to web services and the cloud . To achieve this task, middleware support is required between the industrial . Collecting massive amounts of data in everyday life poses huge challenges for the user to keep control of his data in terms of managing access , sharing and protection . The industrial process requires most of the tasks to be performed locally because of delay and security requirements and structured data to be communicated over the Internet to web services and the cloud . To achieve this task , middleware support is required between the industrial environment and the cloud/web services. In this context , fog is a potential middleware that can be very useful for different industrial scenarios . Fog can provide local processing support with acceptable latency to actuators and robots in a manufacturing industry . Additionally , as industrial big data are often unstructured , it can be trimmed and refined by the fog locally , before sending it to the cloud . One of the new platform which applies this kind of computations is FOGFLOW and here are some of its specifications :

a. Context-driven orchestration mechanism

b. Serverless edge computing

c. Based on Docker

d. Dynamic data orchestrator

II. ANALYSIS

In the following paragraphs we are going to explain our experiences of working with FOGFLOW .

At the first step , we faced lack of well-defined documentation and implemented instances which put users and developers in a risky situation even if they already know all the advantages of this framework and these are rooted in not being "an actual product" , instead , it comes from a research lab so the number of compassionate developers to improve it , is handful .

In the next one, we tried to define a large scale scenario but restricted to comprehend its utmost capability of geo - distributed data control with low latency over the cloud .

A. Scenario

Smart factory use case : The earlier referred holding company has the intention to apply some robots and human bracelets in their factories by using FOGFLOW to control and compute the data .

A robot moves around and does some specific jobs then sends the requested data to the system. We designed two distinct tasks (two separate fog function) for a robot as follows :

1. Idel _ notifier

2. state _ detector

Idel _ notifier :

Idle notifier on a Result update:

- If the status recorded in the result is Idle and last_interval is grater than a fixed threshold, publishes an Error entity

State _ detector :

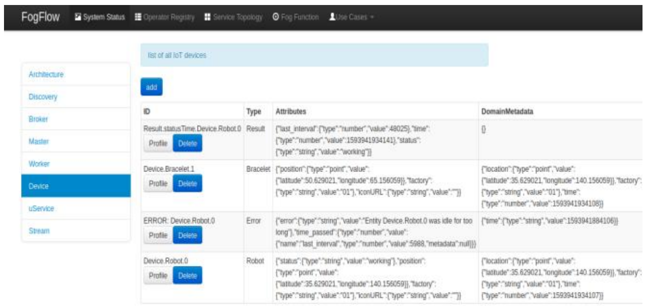
State detector on a Robot update:

- If no result entities exists with the same numeric suffix in the id than the Robot, it creates one with { last _ interval : 0 , time : *currentTime* , status : *receivedRobotStatus* }
- If a result exists but the recorded status is different from the one of the robot update, the existing result is replaced with the one described at point 1
- If a result exists and the statuses are the same, the result is replaced with one having last _ interval = last interval$_{old}$ + *currTime* – time$_{old}$

State detector assumes that all Results existing have a different numeric suffix in the id, so that if a Result exists related to a robot , no other result exists related to the same robot .

B. Implementation

As Fogflow is using Docker so the compatibility of designing with every OS (Linux , Mac , Windows) would be guaranteed . At early steps lack of a series of facilities was felt such a powerful analytical and visualization tool that has solved later on by integrating this framework with Grafana which is a known and well designed tool for the needs mentioned . Then we decide to programming two devices and trying to connect them with the dashboard (Figure.1) . next step was defining some so-called Fog function (based on serveless edge computing) to specify the interaction between devices and the system .



ID	Type	Attributes	DomainMetadata
Result.statusTime.Device.Robot.0	Result	{"last_interval":{"type":"number","value":48025},"time":{"type":"number","value":1593941934141},"status":{"type":"string","value":"working"}}	{}
Device.Bracelet.1	Bracelet	{"position":{"type":"point","value":{"latitude":50.629021,"longitude":65.156059}},"factory":{"type":"string","value":"01"},"iconURL":{"type":"string","value":""}}	{"location":{"type":"point","value":{"latitude":35.629021,"longitude":140.156059}},"factory":{"type":"string","value":"01"},"time":{"type":"number","value":1593941934108}}
ERROR: Device.Robot.0	Error	{"error":{"type":"string","value":"Entity Device.Robot.0 was idle for too long"},"time_passed":{"type":"number","value":{"name":"last_interval","type":"number","value":5988,"metadata":null}}}	{"time":{"type":"string","value":1593941884106}}
Device.Robot.0	Robot	{"status":{"type":"string","value":"working"},"position":{"type":"point","value":{"latitude":35.629021,"longitude":140.156059}},"factory":{"type":"string","value":"01"},"iconURL":{"type":"string","value":""}}	{"location":{"type":"point","value":{"latitude":35.629021,"longitude":140.156059}},"factory":{"type":"string","value":"01"},"time":{"type":"number","value":1593941934107}}

Figure.1

- Robot and Bracelet are actual devices

- Result is an in–the–middle data (the outcome of state detector)

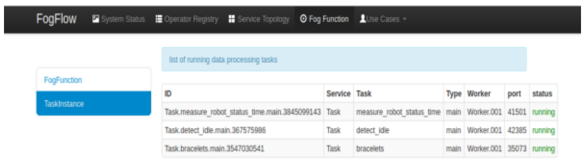
- Error is the outcome of detect idle.

By designing the fog function the rest will be done automatically by fogflow, including :

- Triggering the submitted fog function when its input data are available .
- Deciding how many instances to be created according to its defined granularity .
- Deciding where to deploy the created instances .

There are two separate way to registering the Fog function , docker image and the related operator on the Fogflow . for making them you can use the fogflow dashboard's form wizard for having a easier registration but manually so each time you need to configure them , also you can program three separate file for register and automate them.

Here in the Figure.2 you can see some result of running fog functions on the dashboard .



list of running data processing tasks

ID	Service	Task	Type	Worker	port	status
Task.measure_robot_status_time.main.3845099143	Task	measure_robot_status_time	main	Worker.001	41501	running
Task.detect_idle.main.367575986	Task	detect_idle	main	Worker.001	42385	running
Task.bracelets.main.3547030541	Task	bracelets	main	Worker.001	35073	running

Figure.2

C. Technical challenges

Although we went through all the steps according to the instructions, we encountered a series of problems as bellow :

- The platform doesn't work properly on all networks (for some unknown reason we need to use mobile wifi hotspot)
- The code running in fogflow is not immediately updated if changed on docker hub. It is necessary to record it again by hand (or handcrafted scripts)
- The registration of functions can be automated (useful for , in example , replication) but the json structure to use is not intuitive
- It is possible to add edge nodes at runtime (Dynamically) . However , the devices are not automatically (by fogflow) moved to the new edge if it is spatially closer .
- There are some difficulties for having a deep understanding for programming model , it is not easy.

D. Fogflow ability

As you may know normal operator defined as a function with specific APIs but for fog computing they have some limitation such a flexibility so fogflow provide their operator as a dockerized application based on NGSI (Figure.3)[1] .

(Dockerizing an application is the process of converting an application to run within a Docker container)

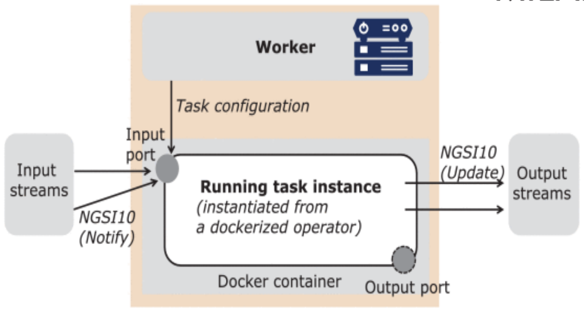


Figure.3

E. Conclusion

The main idea of implementing Fogflow is appreciable and also it has bright future perspective if and only if the number of develovers who are interesting with working an Iot platform which is scalable and capable to handle large amount of data and processing them , become more and more (being open source provide this situation) .

The framework itself is well suited for microservice-oriented applications: deploying such applications would consist in making a function out of each service. However, the more monolithic the application is, the harder it would be to deploy it as a fogflow application.

F. Refrence

[1] "FogFlow : Easy Programming of IoT Services Over Cloud and Edges for Smart Cities"

Bin Cheng , Gürkan Solmaz , Flavio Cirillo , Ernö Kovacs , Kazuyuki Terasawa , and Atsushi Kitazawa