A Design of Data Processing Applications Based on Micro-service Architecture for Wearable Robot Platforms

Yoon-Jae Chae[1], Yoon-Young Park[*2], Kyung-Oh Lee[3], Se-Yeob Kim[4], Ho-Won Lee[5]

[1,*2,3,4,5]*Computer Engineering, Sunmoon Univ.,ASAN, KS002,Republic of Korea*
kodeep11@gmail.com[1], yypark@sunmoon.ac.kr[*2], leeko@sunmoon.ac.kr[3], ksy0789@naver.com[4], kck3156@gmail.com[5]

Article Info
Volume 81
Page Number: 106 - 113
Publication Issue:
November-December 2019

Abstract

Background/Objectives: A method for processing wearable robot sensor data is needed. So it needs a micro-service-based data processing application packages that can be applied to wearable robot platforms server architecture.

Methods/Statistical analysis: Design a micro-service based wearable robot platforms server architecture, develop applications so that servers perform functions for processing sensor data, and containerize those applications through docker. Containerized applications are packaged and easily served to users.

Findings: Many different types of platforms are now widely used. However, there are not many wearable robot platforms. It will design a wearable robot platforms server architecture and design a software structure based on micro-service so that it can process sensor data extracted form wearable robots. So we designed the wearable robot platform server architecture by utilizing Docker, which is a representative technology of micro-service. In addition, each application that operates on that server is containerized and packaged, making it easy for users to use.

Improvements/Applications: It can be used as a server for robot platforms that process various types of wearable robot sensor data.

Article History
Article Received: 3 Jan 2019
Revised: 25 March 2019
Accepted: 28 July 2019
Publication: 22 November 2019

Keywords: *Wearable Robot Platform, Data Processing Application, Packages, Micro-service, Docker, Container*

I. Introduction

There has been a growing number of platforms in recent years. In particular, the number of platforms that utilize data while storing large amounts of data has increased more than before. There are many services that utilize large amount of data in big data and machine learning, which are on the rise these days. Not only DB's ability to store data, but services that utilize data in real time have increased. And these types of services are often based on cloud systems. In particular, more and more clouds are utilizing public clouds. There has also been a shift in how to develop a cloud system-based platform. The platform currently listed provides a variety of services. Among other platforms, there are not many in Korea to store and utilize sensor data from wearable robots. So in this paper, we will design the server architecture that works on the wearable robot platform, design the data processing applications that work on the server, package those applications and provide them to users. In this paper, Chapter 2 is defined for the relevant research, and Chapter 3 is defined for the design of the wearable robot platform system architecture. Chapter 4 is defined for conclusion.

II. RelatedWork

In this paper, I would like to describe the technologies that I would like to apply to design a wearable robot platform server architecture that I want to design.

Amazon Web Service(AWS)

Amazon Web Services is an on demand cloud computing platform metering pay-as-you-go service that provides a comprehensive basic abstract technology infrastructure and distributed computing building blocks and tools. AWS technology is implemented in server farms worldwide and is maintained by Amazon subsidiaries. Fees are determined by the combination of usage, hardware / OS / software / network functions chosen by the subscriber, required availability, redundancy, security and service options.[1]

Amazon Elastic Compute Cloud(Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides scalable computing capacity in the cloud.Through Amazon's simple web service interface, it can secure and configure capacity while minimizing friction.It has complete control over computing resources and can run in Amazon's proven computing environment.Amazon ec2 can quickly scale up and down capacity as computing needs change, reducing the time it takes to acquire new server instances and boot them up to minutes.[2]

Amazon Simple Storage Service(Amazon S3)

Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface that lets you set up and retrieve large amounts of data from anywhere on the Web.It is designed to provide 99.99999999% durability and can extend over trillions of objects in the entire book world.The customer uses S3 as a bulk repository or "data rake" for analysis.Backup and recovery, disaster recovery, serverless computing, and many cloud-native applications also use S3 as their primary storage.[3]

SoftwareArchitecture for Data Platforms

Software architecture is the process of converting software characteristics such as flexibility, scalability, feasibility, reuse, and security into structured solutions that meet technical and business expectations.It should be noted that the software architecture is a key feature of performance and low fault tolerance, scalability, and reliability.[4]

Monolithic Architecture

Monolithic architecture is a traditional integrated model for software program design. Monolithic means it's all made up of one piece.Monolithic software was designed independently.The components of a program are interconnected and interdependent, not loosely coupled, as in the case of a modular software program.To execute or compile code in a closely connected architecture, each component and its associated components must be present.The advantages of a monolithic architecture can be easier to test and debug because the better throughput and the fewer components, the fewer variables.[5]

Microservice Architecture

The micro-service architecture delivers large, complex applications quickly and frequently reliably.And organizations can develop technology stacks.The characteristics of the micro-service architecture are high maintenance and testable and have loose coupling.It can be deployed independently and structured around business functions and owned by a small team.Ideal for large, complex applications is the micro-service architectural pattern.[6]Developers are easy to understand and IDE is faster, which improves

November-December 2019
ISSN: 0193-4120 Page No. 106 – 113

developer productivity, and applications start faster, improving developer productivity and speeding deployment.When using a micro-service architectural pattern, it determines when it is appropriate to use.In general, whether to develop applications quickly can be an important issue for startups, whose biggest challenge.Most companies using micro-service architectures have large Web sites.They have evolved from a single architecture to a micro-service architecture.[7]

VirtualizationTechnique for Data Platforms

Virtualization refers to the creation of a virtual version of a virtual machine hardware platform, storage devices, and computer network resources, rather than the actual version.Virtualization is a way of logically dividing system resources provided by mainframe computers among multiple applications.[8]

Virtual Machine(VM)

Virtual machines are emulation of computer systems.Virtual machines are based on computer architecture and provide the functionality of real computers.Essentially, virtual machines are divided into system virtual machines and process virtual machines, which provide the necessary functionality to replace the actual machines and to run the entire operating system.Hypervisors can share the native execution they use and be separated from each other to allow multiple environments, manage hardware, and still exist on the same physical system.The hypervisor uses hardware-assisted virtualization and virtualization-related hardware primarily on the host CPU.Process virtual machines are designed to run computer programs in a platform-independent environment.[9]

Docker

Docker is a software platform that can rapidly build, test, and deploy applications.[10]There is a concept called container in the docker.A container in software is a standard software unit that packages code and all dependencies, so applications run quickly and reliably from one compute environment to another.A container image is a lightweight, standalone software package that includes everything (code, runtime, system tools, system libraries, and settings) to run the application.The container image becomes the container at run-time, and for the docker container, the image becomes the container when it is run on the docker engine.Containerized software always runs the same regardless of infrastructure.[11]

III. System Architecturefor Wearable Robot Platform

It wants to design and implement server structure of wearable robot platform to process sensor data from wearable robots.

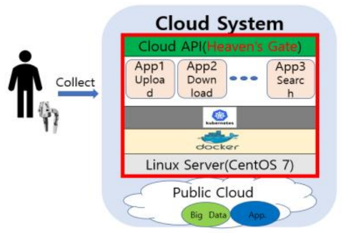


Figure 1. System Architecture for Wearable Robot Platform

Figure 1 is the wearable robot platform system architecture thatdesigned the system architecture to handle measured sensor data from wearable robots.Install a docker on top of a Linux server environment on Amazon Web Services (AWS).Applications that operate on top of that dock are encouraged to perform better by executing a service that helps in the fate and deployment of the application called Kubernetes.The functions of the application are divided into three functions: uploading, downloading and browsing.Each applicable function is developed and containerized as an application based on the micro-service architecture.The user then designed the service to be available on the server through the API.

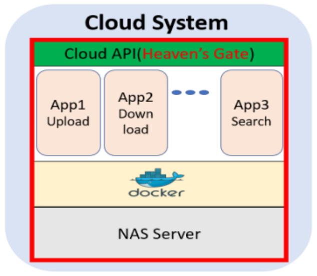
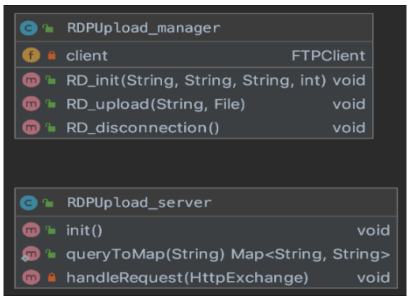


Figure 2. Design a Cloud System Architecture

Figure 2 was designed as a prototype-type architecture before developing a cloud system architecture based on a micro-service architecture.A docker was designed to be placed on the NAS server on behalf of the Public cloud and containerized for applications related to data processing services for sensor data.



```
RDPUpload_manager
client                                   FTPClient
RD_init(String, String, String, int)     void
RD_upload(String, File)                  void
RD_disconnection()                       void

RDPUpload_server
init()                                   void
queryToMap(String) Map<String, String>
handleRequest(HttpExchange)              void
```

Figure 3. RDPUpload Function Application

Class Diagram

Figure 3 defines the source code of the robot sensor data upload function application as a class diagram.The FTPUpload_manager class is a class that has an upload function defined, and the init() method defined within the class is the ability to initialize FTP login lists and ports.The upload() method is a method that contains functions for the FTP upload, and finally the disconnection() method is a method that terminates the connection after the upload function is completed.The Upload_server class is the http communication server class.Theinit() method is the http server creation, port setup, handler setup method, and the queryToMap() method is the method that obtains the URL address when a request for that data is received.

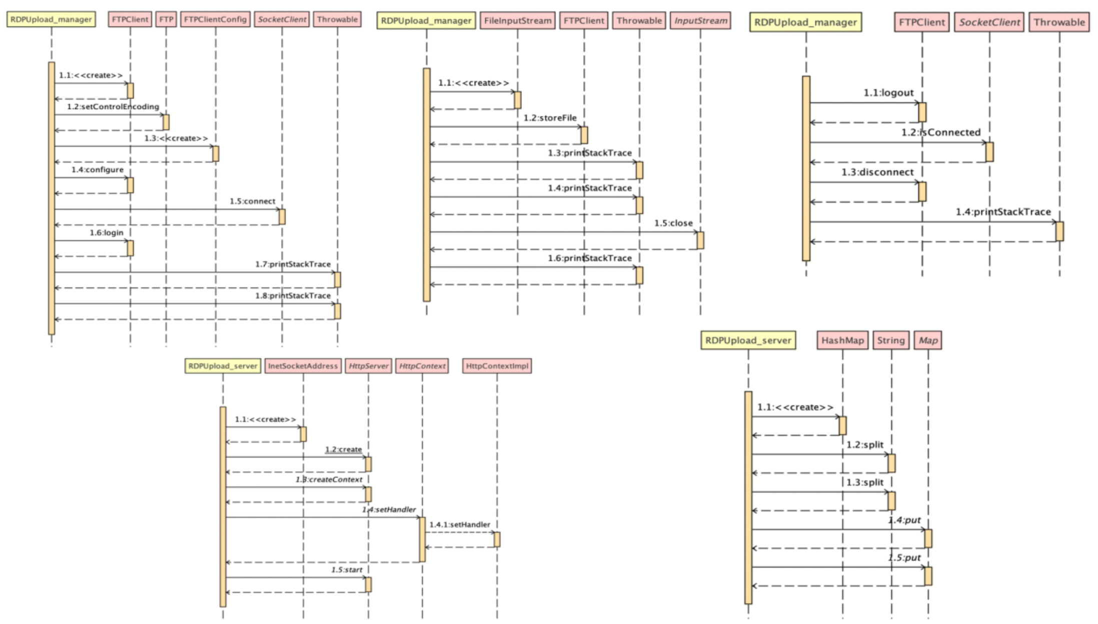
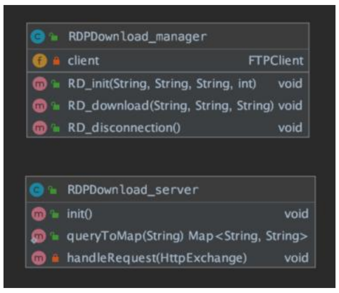


Figure 4. RDPUpload Function Application Sequence Diagram

Figure 4 is a sequence diagram based on the methods of the classes that operate the RDPUpload Function Application.



```
RDPDownload_manager
client                                   FTPClient
RD_init(String, String, String, int)     void
RD_download(String, String, String)      void
RD_disconnection()                       void

RDPDownload_server
init()                                   void
queryToMap(String) Map<String, String>
handleRequest(HttpExchange)              void
```

Figure 5.RDPDownload Function

Application Class Diagram

Figure 5 defines the source code of the robot sensor data download function application as a class diagram.The FTPDownload_manager class is a class that has a defined download capability, and init() method defined within the class is the ability to initialize FTP login lists and ports.The upload() method is the method that contains the function for FTP downloads, and the disconnection() method is the method that terminates the connection after the download function completes.The

Download_server class is the http communication server class.The init() method is the http server creation, port setup, handler setup method, and the queryToMap() method is the method that obtains the URL address when a request for that data is received.

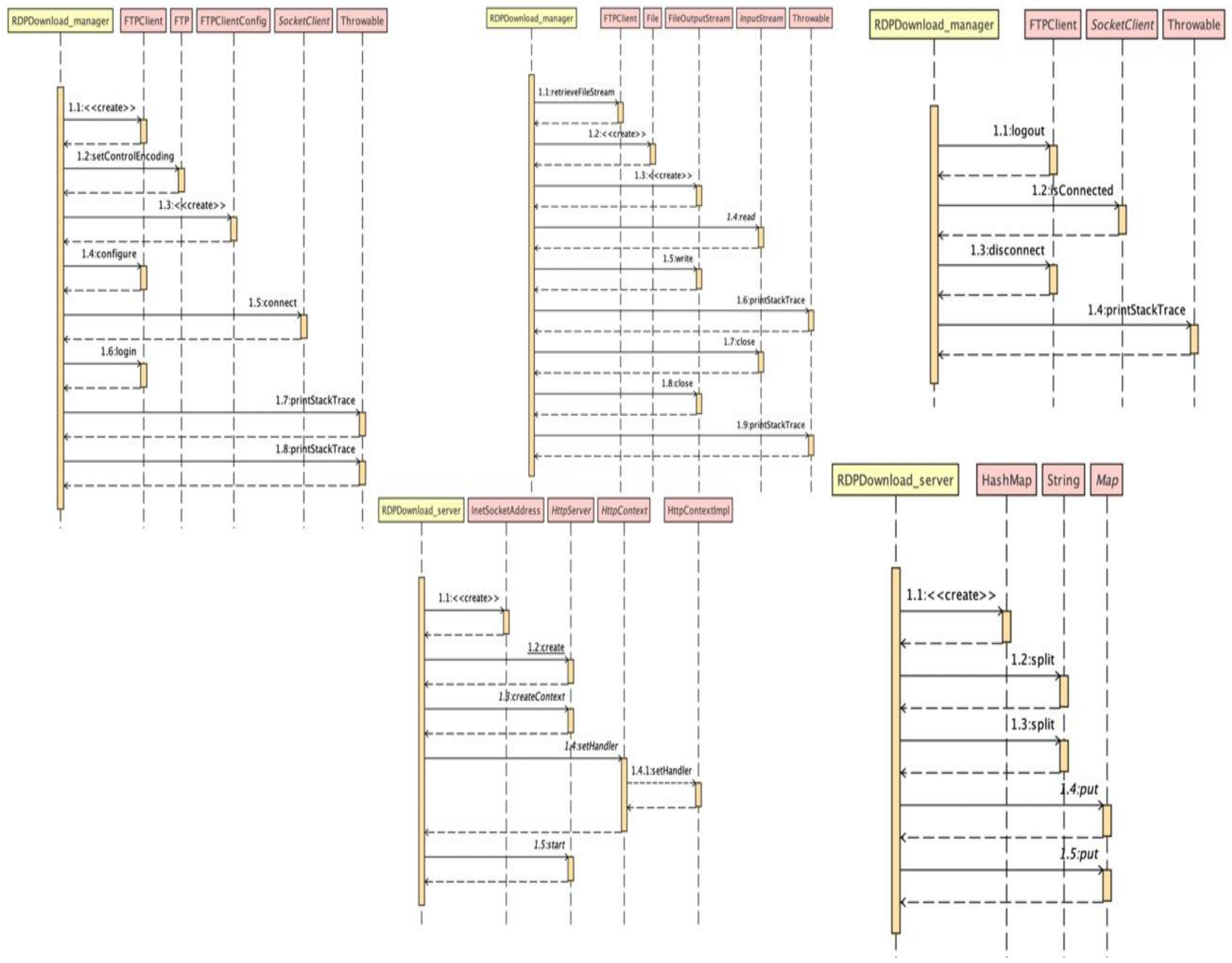
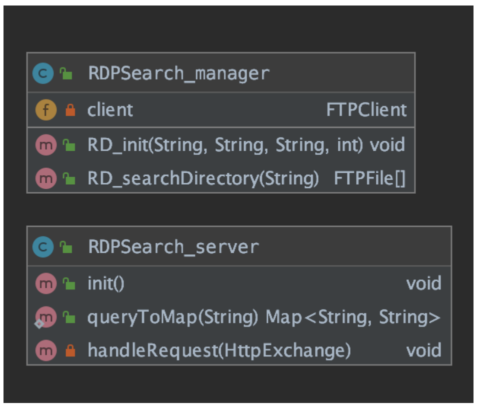


Figure 6. RDPDownload Function Application Sequence Diagram

Figure 6 is a sequence diagram based on the methods of the classes that operate the RDPDownload Function Application.



RDPSearch_manager
client FTPClient
RD_init(String, String, String, int) void
RD_searchDirectory(String) FTPFile[]

RDPSearch_server
init() void
queryToMap(String) Map<String, String>
handleRequest(HttpExchange) void

Figure 7. RDPSearch Function Application

Class Diagram

Figure 7 defines the source code of the robot sensor data search function application as a class diagram.The FTPSearch_manager class is a class in which navigation functions are defined, and the init() method defined within the class is the ability to initialize FTP login lists and ports.The searchDirectory() method is the method that imports the directory information.The Search_server class is the http communication server class.The init() method is the http server creation, port setup, handler setup method, and the queryToMap() method is the method that obtains the URL address when a request for that data is received.

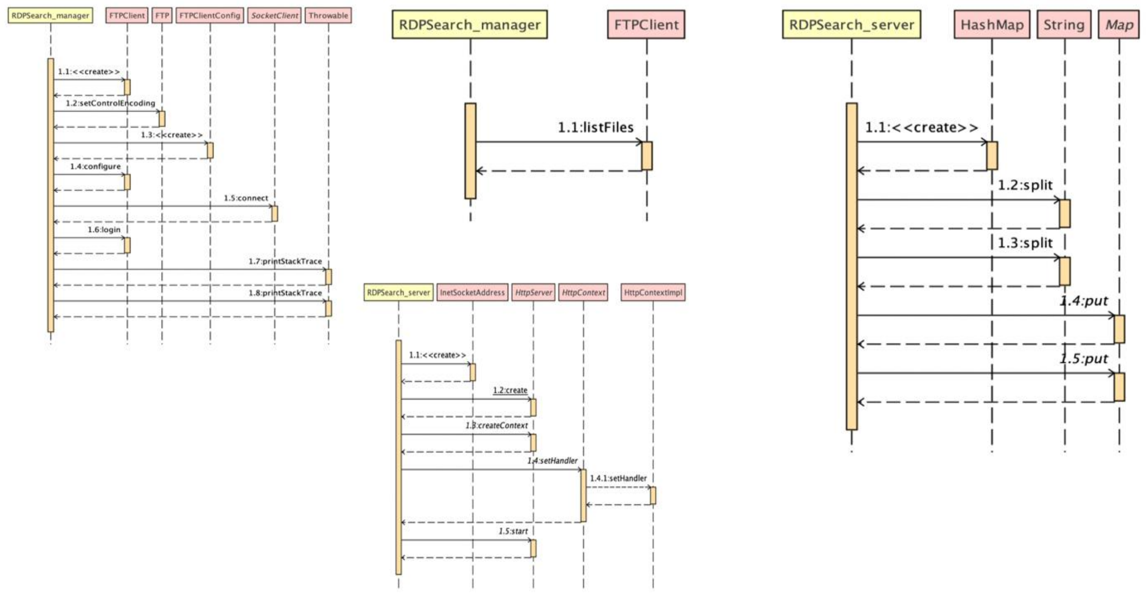
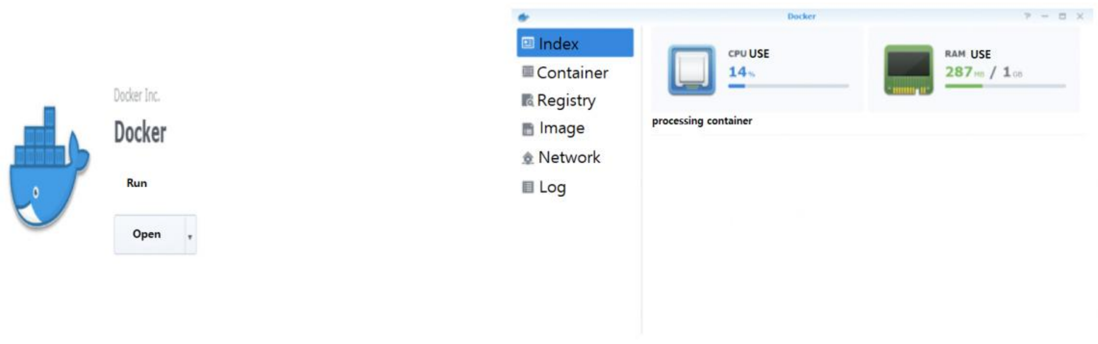


Figure 8. RDPSearch Function Application Sequence Diagram

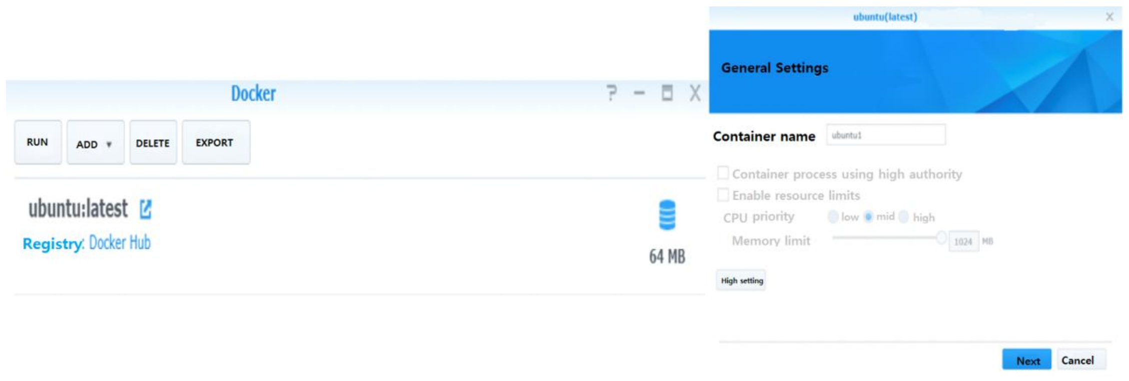
Figure 8 is a sequence diagram based on the methods of the classes that operate the RDPSearch Function Application.



CPU USE 14%
RAM USE 287 MB / 1 GB
processing container

Figure 9. Install Docker and Processing Docker in NAS

Figure 9 runs the Docker after installing the Docker on the NAS server that replaces the Public cloud and shows subsequent CPU and RAM usage. Since the container has not been created at the moment, there is nothing in the status window.

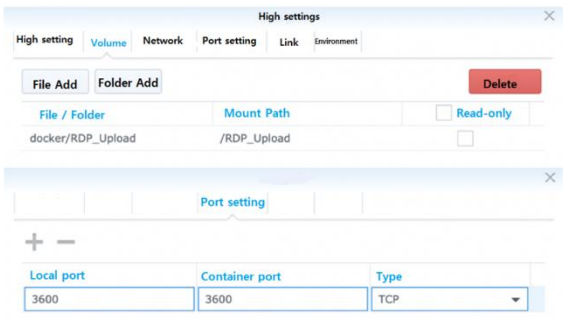


ubuntu:latest
Registry: Docker Hub
64 MB

Container name ubuntu1
Container process using high authority
Enable resource limits
CPU priority low mid high
Memory limit 1024 MB

Figure 10. Install Docker Image & Create a Container

Figure 10 downloads and selects a contextual OS on which operating system to operate in the Docker container and then creates a container that runs on the Docker.


High settings

High setting | Volume | Network | Port setting | Link | Environment

File Add | Folder Add | Delete

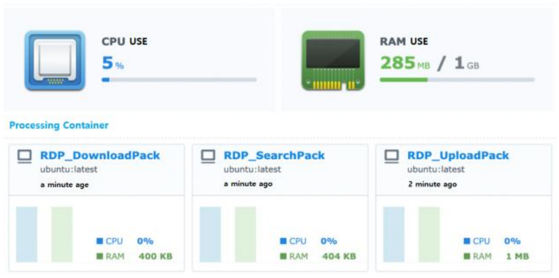
File / Folder	Mount Path	Read-only
docker/RDP_Upload	/RDP_Upload	☐

Port setting

Local port	Container port	Type
3600	3600	TCP

Figure 11. Setting a Volume & Port

Figure 11 is a path setting such as an application executable to run in that Docker container, and when a container is created, it is copied to the mount path specified by the user.In addition, the HTTP communication port number set in the executable file must match the port number set in that container.So the HTTP communication port numbers in each container are different.The code set in the application execution code was set to 3600 in the Upload function class, the Download function method to 3601 and the Search function method to 3602.


CPU USE 5%

RAM USE 285MB / 1GB

Processing Container

RDP_DownloadPack
ubuntu:latest
a minute age
CPU 0%
RAM 400 KB

RDP_SearchPack
ubuntu:latest
a minute ago
CPU 0%
RAM 404 KB

RDP_UploadPack
ubuntu:latest
2 minute ago
CPU 0%
RAM 1 MB

Figure 12. Run a Docker Container

Figure 12 containerizes the application corresponding to each function and activates the container for that service, and the server is complete in accordance with Figure 2's architecture.And because each container is using ubuntu OS, and the execution code is developed in Java, you have to set up a Java environment for each container.Therefore, JDK and Vim editor were installed to activate the corresponding execution code.

It packages each of the applications that are containerized to make it easy for users to serve

IV. Conclusion

To utilize sensor data from wearable robots, we designed a microservice-based wearable robot platform server architecture, containerized applications of corresponding upload, download, and navigation functions to process data on servers through dockers, and code reuse and maintenance due to development based on micro-service will perform much better than a monolithic-based software structure. It is also expected to be very easy to use by making it easy to provide to users through packaging. In the future, it will add data processing features and server features that work on wearable robot platforms, implement them to work on public clouds, not on actual NAS servers, and implement packages to be provided on the cloud.

V. Acknowledgment

This work was supported by the Technology Innovation Program (Robot Industry Core Technology Development Project) (20003914, Development and commercialization of motorized exoskeleton robots for the assistant of daily life exercise) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea)

References

[1] Wikipedia.org : Wikipedia. USA: Society; [cited 2019 Sep 18]. Available from: https://en.wikipedia.org/wiki/Amazon_Web_Services.

[2] Amazonaws.com : Amazon. USA: Amazon; [cited 2019 Sep 18]. Available from: https://www.amazonaws.cn/en/ec2/.

[3] Amazonaws.com : Amazon. USA: Amazon; [cited 2019 Sep 18]. Available

from: https://www.amazonaws.cn/en/s3/?nc2=h_l3_sc.

[4] Codeburst.io : Codeburst. USA: Mohamed Aladdin; [updated 2018 Jul 28; cited 2019 Sep 18]. Available from: https://codeburst.io/software-architecture-the-difference-between-architecture-and-design-7936abdd5830.

[5] WhatIs.com : WhatIs. USA: Margaret Rouse; [cited 2019 Sep 18]. Available from: https://whatis.techtarget.com/definition/monolithic-architecture.

[6] Microservices.io : Microsoft. USA: Chris Richardson; [cited 2019 Sep 18]. Available from: https://microservices.io.

[7] Chris Richardson Microservice.io. USA: Microservice.io; [cited 2019 Sep 24]. Available from: https://microservices.io/patterns/microservices.html.

[8] Wikipedia.org : Wikipedia. USA: Society; [cited 2019 Sep 18]. Available from: https://en.wikipedia.org/wiki/Virtualization.

[9] Wikipedia.org : Wikipedia. USA: Society; [cited 2019 Sep 18]. Available from: https://en.wikipedia.org/wiki/Virtual_machine.

[10] Amazonaws.com : Amazon. USA: Amazon; [cited 2019 Sep 24]. Available from: https://www.amazonaws.cn/en/docker/.

[11] Docker.com : Docker. USA: Docker; [cited 2019 Sep 18]. Available from: https://www.docker.com/resources/what-container.