Server Systems Utilizing General-Purpose Cloud Service

MIYAZAKI, Tsuyoshi* KITAMURA, Takashi‡ TAKEUCHI, Osamu‡

ABSTRACT

Fuji Electric has developed a server system that offers an execution environment for IoT services using service applications on a general-purpose cloud service. It absorbs the difference between the cloud services provided by various cloud vendors, allowing service applications to have portability between cloud vendors. Other features include communicating with edge controllers, saving gathered data on the database and monitoring the states of running service applications and resource load. Utilizing server systems allows Fuji Electric to quickly deliver high-quality IoT services that increase customer value.

1. Introduction

Fuji Electric has developed the Fuji Electric IoT platform for the purpose of prevalence of the Internet of Things (IoT) service business. Figure 1 shows the configuration of the platform. On the Fuji Electric IoT platform, the part that offers an execution environment for service applications to provide services required by customers is called server system, which has been built by using general-purpose cloud services. Use of general-purpose cloud services allows infrastructure and hardware resources to be flexibly added or deleted according to the scale of the customer. This facilitates reduction of the lead-time for providing a new IoT service, allowing users to reduce the initial and running costs. It also reduces the troublesome task of operation and maintenance because troubleshooting during operation is left to cloud vendors. In addition, backing up in multiple locations provided by cloud vendors allows the platform to be highly resistant to disasters, such as earthquakes and floods.

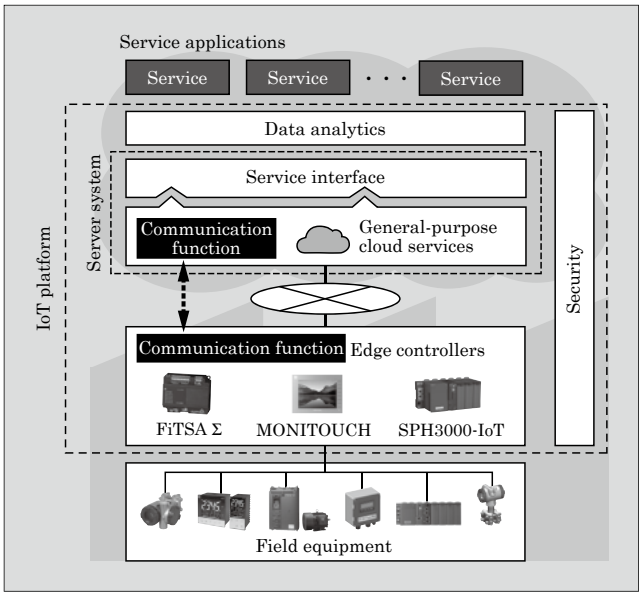

Fig.1 Configuration of Fuji Electric IoT platform

This paper presents an overall picture of the server system on the Fuji Electric IoT platform together with the individual functions that constitute the server system.

2. Server System

The server system on the Fuji Electric IoT platform is required to have the following features:

(a) Ease of system construction with a small-scale configuration
(b) Ease of resource expansion by addition of IoT services and edge controllers
(c) Capability of providing services linked with edge controllers and services of other companies
(d) Capability to ensure security
(e) Capability of stable operation and ease of operation and maintenance
(f) Capability of offering services to overseas customers
(g) Continued provision of existing services provided on premise*1

Figure 2 shows the structure of the server system. To meet these requirements, we have adopted general-

* Corporate R&D Headquarters, Fuji Electric Co., Ltd.
‡ Power Electronics Systems Business Group, Fuji Electric Co., Ltd.

*1: On-premises: Installing and operating information systems, including servers and software, on the premises managed by the user (generally a company)

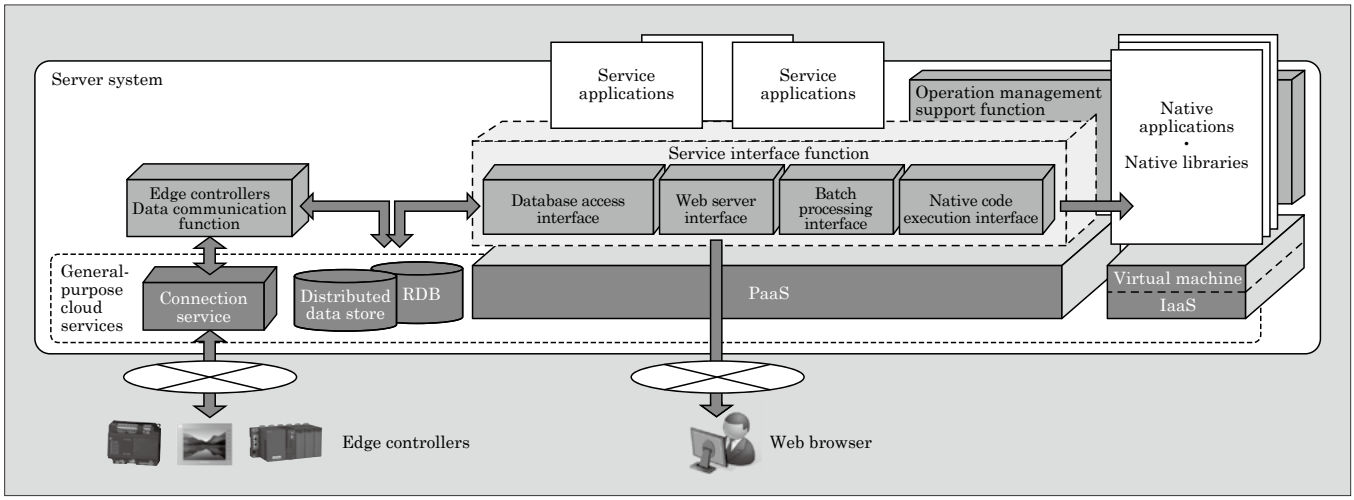

Fig.2 Structure of server system.

purpose cloud services provided by cloud vendors (see Table 1) for the server system and developed the functions described as follows:

(1) Service interface function

This function provides common interface that is used to achieve portability between different cloud vendors for service applications and efficiently develop service applications for users.

(2) Edge controller data communication function

This function provides communication with edge controllers responsible for gathering data in the field and stores this data in a distributed data store.

(3) Operation management support function

This function monitors the service application running state and resource load state and notify any error generated to improve service operation quality.

Table 1 General-purpose cloud services

Service name	Service description
Relational database (RDB)	○ Managed database with high availability ○ Allows disk capacity expansion and other scale-up
Distributed data store	○ Managed database for storing text data, binary data, etc. ○ Allows scale-up such as disk capacity expansion and cluster addition
Platform as a Service (PaaS)	○ Platform for running service applications including hardware and OS ○ Open-source Cloud Foundry used ○ Allows resource expansion of hardware used
Infrastructure as a Service (IaaS)	○ Infrastructure environment required for operating information systems such as hardware and network ○ Allows resource expansion of hardware used and secure communication divided between multiple network levels
Virtual machine	○ Computer environment that a user can occupy on an IaaS ○ Allows resource expansion of hardware used
Connection service	○ Environment for simple and secure gathering of data from edge controllers ○ Allows resource expansion following addition of edge controllers and access control for gathered data

3. Service Interface Function

This function is required to provide the following features:

(a) Multi-vendor support in view of the case that the customer specify a specific cloud vendor
(b) Capable of quickly adding service applications and increasing the number of users
(c) Ensured sense of security for users in the use of service applications
(d) Support for various Web browsers
(e) Capability of high-speed handling of data of various formats (numeric value, text, image, voice)
(f) Support for existing services

To meet these requirements, we have developed the following 4 types of interfaces.

3.1 Database access interface

The database access interface is used to access the relational database (RDB) and distributed data store provided by general-purpose cloud services.

Cloud vendors employ their own particular RDB and distributed data store for their general-purpose cloud services, and the difference between different vendor's databases must be absorbed within the database access interface. To deal with this issue, we have developed the common interface through which service applications access databases. This interface absorbs the database difference by individually implementing processes for accessing different types of databases. This has made various general-purpose cloud services available for use without depending on specific cloud vendors. As a database, using distributed data store capable of accumulating text and binary data allows users to handle various data formats.

3.2 Web server interface

The Web server interface help develop Web applications efficiently to release to customers.

For efficient development of Web applications, we employed the model-view-controller framework provided for open-source software (OSS) to develop the security function for user authentication and user-specific access control and session management function that accommodates increase of the number of users. This allows users to use their desired Web browser for using Web applications with a sense of security. Increase of the number of users is also quickly accommodated.

3.3 Batch processing interface

The batch processing interface is used to efficiently develop and execute batch processing applications for purposes such as aggregation and form creation.

We developed parallel processing execution and distributed processing execution functions in a multi-thread and multi-instance environment for high-speed execution of individual batch processes and simultaneous execution of multiple batch processes. These functions minimize the impact of adding new service applications on existing service applications.

3.4 Native code execution interface

The native code execution interface is to execute applications and libraries written in the C Language.

Generally, service applications are run on a Platform as a Service (PaaS), which makes unusable existing services and data analytics services written in the C language that are provided on-premise. To use these services, we have developed the function for this server system that service applications on a PaaS execute existing services and data analytics services implemented on a virtual machine on an Infrastructure as a Service (IaaS).

4. Edge Controller Data Communication Function

This function is required to provide the following features:

(a) Capability of communicating with edge controllers and IoT devices provided by other companies
(b) Ease of communication even with increased number of devices connected
(c) Safe communication between edge controllers and the server system
(d) Capability of communication of data of various formats (numeric value, text, image, voice) in view of future expansion of applications

4.1 Employment of de facto standard protocol

Communications between edge controllers and the server system employ the Message Queue Telemetry Transport (MQTT) protocol, which is widely in use as a de facto standard. This allows communications with edge controllers and IoT devices provided by other companies and facilitates communications even when the number of devices connected increases.

4.2 Connection authentication

Data sent by edge controllers are provided with an access token for device authentication. The access token can be verified by the server system to allow communication only with edge controllers permitted to communicate. This ensures safe communications between edge controllers and the server system.

4.3 Employment of distributed data store

Data sent from edge controllers are stored in a distributed data store. As a file data format for storing in a distributed data store, we have employed the Java Script Object Notation (JSON) format, which supports various data formats. This allows various types of data such as measured values and images to be communicated between edge controllers and the server system.

Service applications can use data stored in a distributed data store via the service interface function described earlier.

5. Operation Management Support Function

This function is required to provide the following features:

(a) Ensured stable operation of service applications
(b) Capability of efficient operation management
(c) Capability of prompt recovery from any error generated
(d) Capability of preventing impact of any error

Table 2 Operation management support function

Function name	Function description
Application monitoring	○ Monitors the viability and log of various service applications and detects any error generated in real time. ○ Allows stable operation of service applications and prompt recovery.
Resource monitoring	○ Monitors the resource state of virtual machines on a PaaS and IaaS and detects any error such as resource insufficiency in real time. ○ Allows stable operation of service applications, prompt recovery and prevention of any error from spreading.
E-mail notification of any error detected	○ Notifies the administrator of any error detected by application monitoring and resource monitoring by e-mail. ○ Allows improvement of efficiency of operation management and prompt recovery.
Operation monitoring portal site	○ Provides a portal site to show the operating state of service applications. ○ Allows stable operation of service applications and improvement of efficiency of operation management.

generated on other service applications

To meet these requirements, we have developed the functions listed in Table 2.

6. Postscript

This paper has described a server system that makes use of general-purpose cloud services. By using general-purpose cloud services, IoT services with high customer value can be provided promptly at a high quality level. The server system absorbs the difference between various general-purpose cloud services to allow use of the cloud vendors specified by customers.

In the future, Fuji Electric intends to develop functions of linking with service applications provided by other companies and providing our service applications to other companies. We will thereby work to enhance the functionality of the server system to offer IoT service with even higher customer value.

Fuji Electric

Innovating Energy Technology