ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

PUBLIC CLOUD-BASED REMOTE ACCESS INFRASTRUCTURE FOR NEUTRON SCATTERING EXPERIMENTS AT MLF, J-PARC

K. Moriyama, Comprehenshive Research Organization for Science and Society (CROSS), Tokai, Ibaraki, Japan
T. Nakatani, Japan Atomic Energy Agency (JAEA), Tokai, Ibaraki, Japan

Abstract

An infrastructure for remote access for supporting research workflow is essential for neutron scattering user facilities such as J-PARC MLF. Because the experimental period spans day and night, service monitoring the measurement status from outside the facility is required. Additionally, convenient way to bring a large amount of data back to user's home institution and to analyse it after experiments is required. To meet these requirements, we are developing a remote access infrastructure as a front-end for facility users based on public clouds. Recently, public clouds such as Amazon AWS and Google Cloud, have been rapidly developed, so that development and operation schemes of computer systems have changed dramatically. Various architectures provided by public clouds enable advanced systems to develop quickly and effectively. Our cloud-based infrastructure comprises services for experimental monitoring, data distribution and data analysis, using architectures such as object storage, event-driven serverless computing, and virtual desktop infrastructure (VDI) based on a microservice approach for application implementation. Facility users can access this infrastructure using just a web browser and VDI client. This paper reports the current status of this remote access infrastructure.

INTRODUCTION

The Materials and Life Science Experimental Facility (MLF) at the Japan Proton Accelerator Research Complex (J-PARC) is a neutron scattering experiment user facility with one of the world's highest intensity pulsed neutron beams in operation since 2008. Currently, the 21 installed neutron instruments are used by domestic and international users, in various research fields to conduct experiments.

Many of the instruments at MLF have introduced IROHA2 [1], which is a web-based integrated instrument control framework that controls data acquisition and sample environmental devices. IROHA2 is able to perform automatic measurement changing measurement conditions. Using this function, long-running measurement over periods of several days can be performed. Therefore, the progress of measurement can be monitored remotely via the web. However, due to security issues and system capabilities, currently, only instrument staff are allowed to monitor the measurement status in this way.

With intense neutron beams, high-precision and -resolution position-sensitive detectors are used with advanced event recording methods for data acquisition; experimental data is generated at very high rates [2]. The total amount of data generated in experiment ranges from hundreds of MB to several TB per experiment.

These data are analysed using Linux-based analysis software. After each experiment, most beamline users would normally take their data back to their home institutions in the portable storage such as hard disks. Since many facility users are unfamiliar with the Linux environment, virtual machines with analysis software installed are distributed to them.

Given this facility usage, to support research workflows of facility users, the remote access infrastructure requirements are:

- The ability to remotely monitor the progress of long-running measurements over the experiments
- To remove the necessity for users to physically take large amounts of raw data back to their home institutions, by allowing data analysis and results acquisition to be performed by one-stop, remote access.
- To allow multiple facility users quick and easy remote access from both inside and outside the country.

To satisfy these requirements, we built a remote-access infrastructure linked to the on-site systems at MLF using the Amazon Web Services (AWS) [2], one of the main public clouds. Figure 1 shows an overview of our remote access.

WHY USE A PUBLIC CLOUD?

The reality of system and infrastructure development has various limitations such as budget, human resources, and security. Given these limitations, using a public cloud as a development platform has the following advantages compared to conventional on-site systems and private clouds:

- No hardware management or installation costs are required.
- System development and operations can start small and can be scaled as need. This can optimize resource usage and thus, operating costs as well.
- Advanced services and functions provided via public clouds enable efficient system development; in particular, a proactive use of advanced AI-related services can be expected.
- The security of the service infrastructure is ensured at a higher level than that of on-premises systems.
- A managed service that simplifies system management and operation.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

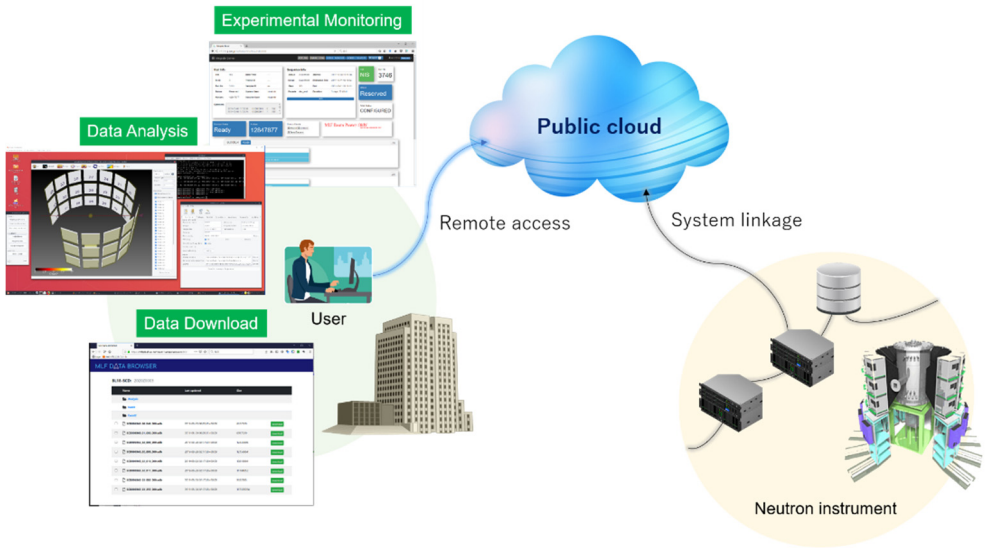

Figure 1: The overview of remote-access infrastructure.

- The global presence of a public cloud allows facility users access with low latency, whether domestic or foreign.

MLF has been in operation over a decade, seeing the budget for the development of new systems reduce as maintenance and upgrade costs for existing systems increase. Therefore, a public cloud, allowing development and operation of more advanced and challenging systems while reducing costs, is quite a useful infrastructure platform for facilities such as MLF

DEVELOPMENT APPROACH

In public clouds, a wide range of services is provided, from a physical level to software. Selecting the appropriate service to use, from these, is important from the view point of system development and operation.

Selection Policy of Cloud Service

Figure 2 shows the hierarchical structure of services described in "as a Service" (XaaS) format, based on a cloud computing service model. The letter "X" represents the consistent parts: Infrastructure, Platform, Container, Desktop, Function, and Software. The following corresponding services are provided:

IaaS is the service located at the bottom of the hierarchy and provides the most basic services such as virtual machines, virtual networks, and storage, in the use of public clouds.

PaaS is the service that provides development platforms including OS and middleware such as RDBMS.

CaaS provides a service for the orchestration of containers such as Docker [3], which is one of the virtualization architectures running applications independently on a host OS.

DaaS provides a desktop infrastructure based on VDI. In DaaS, the same desktop environment can be used regardless of client environment and location.

FaaS is a relatively new service that has attracted attention recently. It provides an execution environment for functions, called serverless computing. FaaS enables extremely rapid application implementation without the need to build extra infrastructure.

SaaS provides software-layer services that do not require server management, such as web mail, blog services and groupware.

As shown by the arrows in Fig. 2, the costs for development and operation as well as the flexibility in development are related to the position of the service within the hierarchy. In terms of operation cost, the infrastructures of the layers below a service in use are managed by the cloud vendor, so the cost decrease as higher-layer services are used. With regard to service usage fee, for example, virtual machines and containers, in the lower layers, are charged for continuous operation times, while serverless computing in the FaaS layer is charged only for function execution time, so the latter can be used at lower cost.

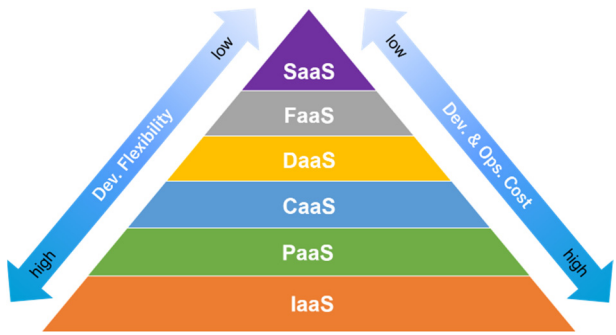

Figure 2: Hierarchical structure of cloud computing service model.

In terms of development cost and flexibility, development becomes easier and costs are reduced as higher-layer services are used; however, since functions and computing resources are locked into a service, the flexibility is reduced at higher layers.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

Therefore, the adopted optimum service, while meeting the requirements, should take into account the trade-off between development flexibility and cost.

Microservice Approach in Cloud Development

As described in the previous section, services of various layers are available in public clouds. To develop the remote-access infrastructure, we used these services to adopt a microservice approach, which has attracted attention as an architecture for application development.

The simplest approach for application implementation is to run conventional monolithic applications on the virtual machine provided in IaaS. In contrast, microservices divide functions into multiple independent small services that communicate with each other, using well-defined API, and build applications by collaboration. Using this approach to public clouds, applications are implemented based on container architecture in CaaS and serverless computing in FaaS.

Figure 3 shows the conceptual configuration of both approaches. In the monolithic application, various functions are implemented together so that they are in a tightly coupled relationship each other. In this case, performance degradation due to failure or resource shortage as well as partial function changes, may affect the entire system. Therefore, it is not easy to modify such a system.

Microservice are designed to overcome the issues of the monolithic application. Services are independent and loosely coupled to each other, leading to higher flexibility, compared with the monolithic application. Each service can be adopted the optimal language and architectures for development and can be modified, deployed, and scaled individually.

Based on this approach, microservices provides highly independent services, at higher layers such as FaaS and DaaS, in public clouds. This allows the effective implementation of applications with high maintainability and adaptability to future changes.

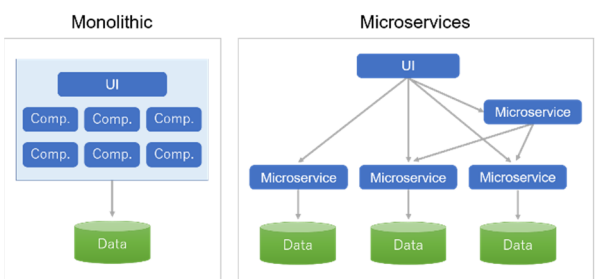

Figure 3: Monolithic and Microservices applications.

CLOUD SERVICES ON AWS

We adopted AWS as a platform for our remote-access infrastructure. It was developed mainly using the services described below.

Object Storage: Amazon S3

Amazon Simple Storage Service [4], called Amazon S3 is a highly available (99.99%) and scalable object storage service, suitable for storing large amounts of data. The-object storage does not use a hierarchical structure like traditional file storage but stores data as objects together with metadata in a flat space called a "bucket."." This service has the following main features:

- A well-defined API for handling data.
- A lifecycle function that automatically deletes or archives data after a certain period.
- Event notifications for collaborating services.
- Static web hosting.

Serverless Computing: AWS Lambda

AWS Lambda [5] is a key service for building cloud-native serverless applications and is an event-driven program execution environment. Applications can be implemented simply by coding, without building a server. This service has the following main features:

- Multi-language support: Java, Go, PowerShell, Node.js, C#, Python, and Ruby.
- Event-driven code execution triggered by collaborating services.
- A billing method with function execution time.
- Automatic scaling according to the size of the trigger event.

This service is particularly useful when building microservices on a serverless computing basis.

API Proxy: Amazon API Gateway

Amazon API Gateway [6] is another key service for serverless computing and is useful for implementing microservices. This service plays an API Proxy by creating an API layer between front-end and back-end services such as AWS Lambda, Amazon EC2, and third-party services. The main features are as follows:

- Fully managed services: API creation, publishing, maintenance, monitoring and protection.
- REST and WebSocket API.
- HTTP methods: GET, PUT, POST, DELETE, etc.

Content Delivery Network: Amazon CloudFront

Amazon CloudFront [7] is a service for the Content Delivery Network (CDN) that is a network optimized for delivering web content over the internet. Web content is cached across multiple *edge* locations, which is the data center of AWS global network, and user requests are routed with the lowest latency. Using this service allows the quick distribution of large amounts of data to facility users around the world.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

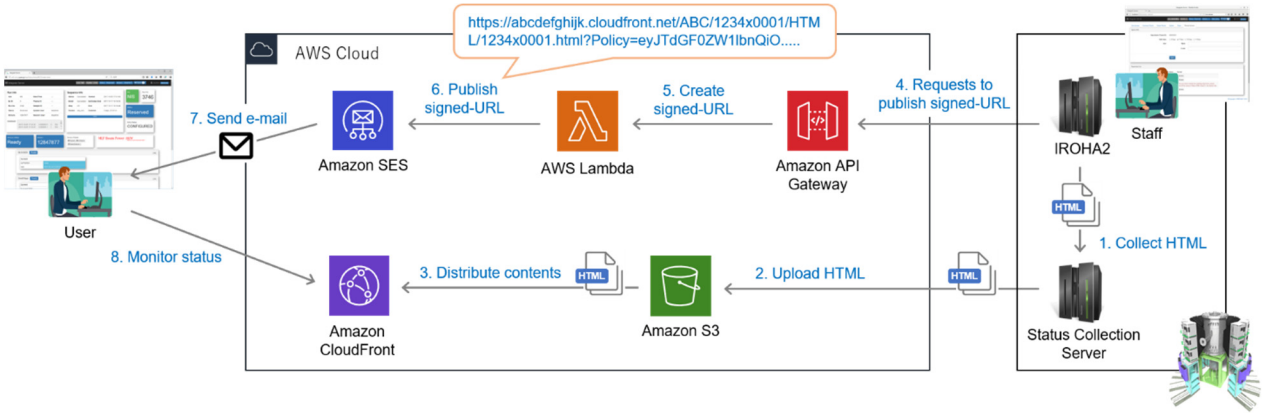

Figure 4: System configuration and processing flow of the experimental monitoring.

Mail Sending: Amazon SES

Amazon Simple Mail Service [8] called Amazon SES is an email sending service. Equipped with an SMTP interface and an API, it can be integrated with other applications and services, such as AWS Lambda, to send email.

VDI: Amazon WorkSpaces

Amazon WorkSpaces [9] is a DaaS-type service that provides a VDI environment in the cloud. This is a fully managed service allowing quick and easy deployment of desktop environments without hardware inventory, OS version patching or complex VDI management. The service has the following main features:

- Multiple OS deployment: Windows and Amazon Linux2.
- Multi-device client: Windows, Mac, Chromebooks, iPad, Fire tablet, and Android tablet.
- Global desktop deployment based on multiple AWS regions.

This service provides a convenient desktop for data analysis for facility users, both domestic and overseas.

EXPERIMENTAL MONITORING

Experimental monitoring is particularly useful when performing long-running measurements. It is not practical for the user to be sat, continuously, in front of the instrument display during an experiment, so the progress of the measurements would ideally be able to be monitored from outside the facility, via the internet.

So far, we have operated on-site systems to monitor experiments from outside the facility, but this provision was only available to instrument staff. Therefore, we have developed a new service, allowing facility users to monitor their experiments remotely by linking the existing on-site systems with public cloud services. Figure 4 shows the configuration and processing flow of this system.

Web Distribution of Measurement Status

Procedures 1–3 in Figure 2 are performed to distribute the measurement status on the web. IROHA2, which is the experimental management system, has a function to generate static HTML files, at regular intervals, of the measurement status displayed on the web interface, for monitoring. The static HTML format is a security measure to prevent unauthorized control from the outside.

The status collection server periodically collects HTML files from the neutron instruments and uploads them to the Amazon S3 bucket. Although Amazon S3 has a function for static web hosting, it does not support HTTPS communication and flexible access control. Therefore, web access is provided through Amazon CloudFront, which is the CDN service.

When HTML files are uploaded to the S3 bucket, these files are automatically distributed to the CloudFront edge location. The communication between Amazon S3 (Origin) and Amazon CloudFront (CDN) is encrypted by HTTPS.

Access Control with Signed URL

On AWS, the service combination of Amazon S3 and Amazon CloudFront is commonly used for delivering private content. Access control is performed using a CloudFront function called “signed URL.”.” Only those who have been issued the signed URL can access the CloudFront.

The signed URL is composed of the following elements:

1. Base URL (CloudFront domain)
2. Query strings
3. Policy statement (Base64 encoded)
4. Policy statement (RSA SHA-1 signed)
5. RSA key pair ID

The policy statement includes information such as accessible resources and expiration dates.

Issue of Signed URL

Procedures 4–7 in Figure 4 are performed to issue the signed URL. The signed URL is generated by a Python code deployed as a function in Amazon Lambda. This Lambda function also sends the generated signed URL to the facility user by email, in cooperation with Amazon SES.

The Lambda function can be executed on demand through Amazon API Gateway. Instrument staff submit a signed URL request using an IROHA2 web interface. Figure 5 shows a screenshot of this interface. The instrument

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

staff issues the signed URL by entering user information (name, email address) and validity period of URL on this web interface.

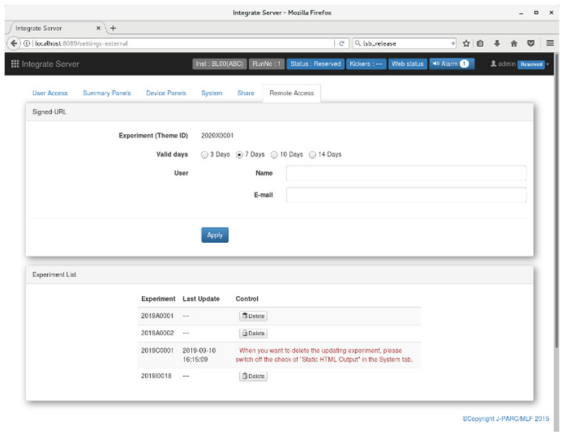

Figure 5: The screenshot of IROHA2 web interface to issue the signed URL.

Web Access

In procedure 8 in Fig. 4, facility users access the experimental monitoring with a web browser by simply clicking on the signed URL issued by email. Figure 6 shows a screenshot of the IROHA2 measurement status.

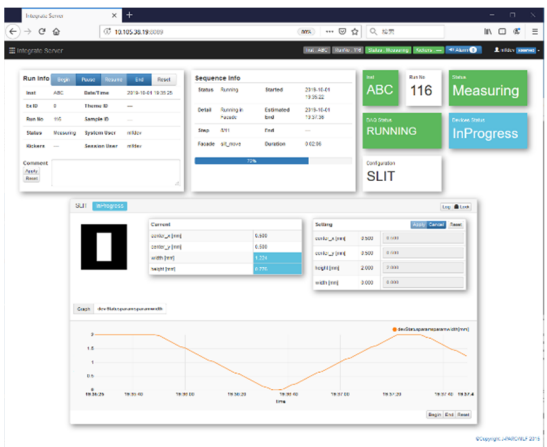

Figure 6: The screenshot of the IROHA2 measurement status.

DATA DISTRIBUTION

We have been distributing data generated in experiments at MLF BL18, which is a beamline for a TOF-Laue neutron diffractometer [10], using only Amazon S3 since FY2018. This system used an FTP client to access Amazon S3, but in order to enable easier data access, we have improved the system to a web-based service using the same web delivery and access controls as the experimental monitoring described in the previous section. In addition, the system introduced an event-driven serverless computing architecture linking Amazon S3 and Amazon Lambda. Figure 7 shows the system configuration and processing flow.

Uploading Data to the S3 Bucket

In the procedure 1 of Figure 7, data uploads from the instrument to the S3 bucket are performed on demand by the MLF Experimental Database (MLF EXP-DB), a web-based data management system [11]. Data uploaded on the S3 bucket are automatically encrypted.

HTML Generation with Serverless Computing

In procedures 2–3, using an event-driven serverless computation of Amazon Lambda, the data list in the S3 bucket is automatically converted into HTML format.

When data is uploaded, Amazon S3 publishes an event, described in JSON format that contains information about the uploaded data and directory structure, to Amazon Lambda. Then, a Lambda function is triggered, written in Python code, to generate the HTML files in the S3 bucket.

Web Distribution and Access Control

Procedures 4–8 are basically the same as the experimental monitoring, but the signed URL is issued from the MLF EXP-DB. Figure 8 shows a screenshot of MLF EXP-DB issuing a signed URL.

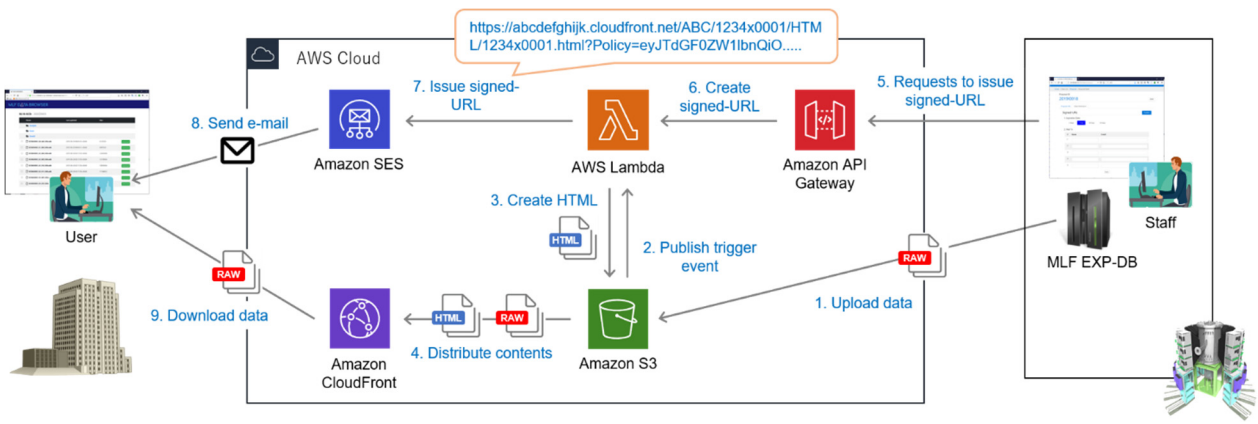

Figure 7: System configuration and processing flow of the data distribution.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

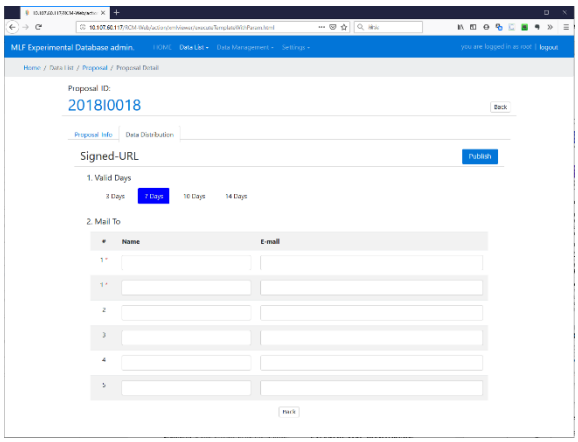

Figure 8: The screenshot of MLF EXP-DB web interface for issuing signed URL.

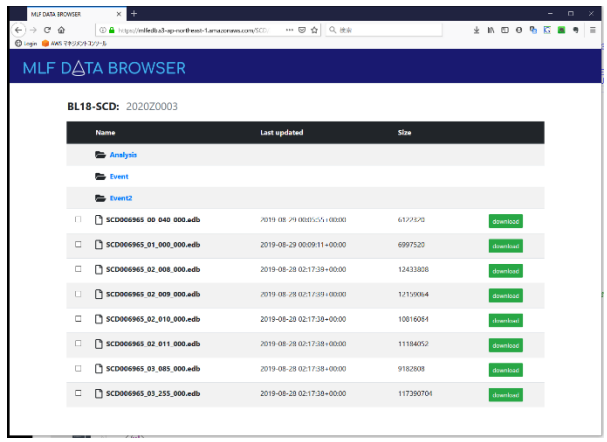

Figure 9: The screenshot of the MLF Data Browser.

Data Downloading

In procedure 8, the facility user accesses a web site for data downloading by clicking the signed URL issued by email and downloading the data. Figure 9 shows a screenshot of the web site called “MLF Data Browser”.

Access for facility users is directed to the lowest latency edge location by Amazon CloudFront, so they can download large amounts of data at high speeds regardless of the location they are accessing from.

DATA ANALYSIS

We built a remote desktop environment for data analysis using Amazon WorkSpaces, which used the DaaS service. Figure 10 shows the configuration of this system. Facility users can access the deployed desktop environment by using a WorkSpaces client. This desktop environment has analysis software already installed via Amazon Linux2, which is a RHEL7 compatible OS. User authentication is performed using a directory service called “Simple AD” of the Amazon Directory Service. In addition, the desktop environment is NFS mounted on the Amazon S3 bucket through Amazon Storage Gateway for using data on the S3 bucket.

Figure 11 shows a screenshot of the desktop environment installed.

STARGazer is a piece of data processing and visualization software for single-crystal neutron diffraction experiments in MLF BL18.

This desktop environment allows facility users to perform data analysis without installing software or downloading data to their PC. It also provides them with the same interface and online working environment regardless of their location. Furthermore, the workload of instrument staff regarding user software installation and support can be eliminated.

SUMMARY AND FUTURE PLANS

We have developed a remote-access infrastructure for facility users to perform data downloads and analysis alongside using the AWS cloud platform, linking the on-premises system such as IROHA2 and MLF EXP-DB. The microservice approach was adopted to develop these systems by utilizing architectures such as serverless computing, object storage, and a VDI, allowing effective development and management.

The developed services are planned to be introduced to the neutron instrument at MLF, gradually from FY2019. In addition, other public cloud services such as Google Cloud Platform and Microsoft Azure also will be introduced as needed in the future.

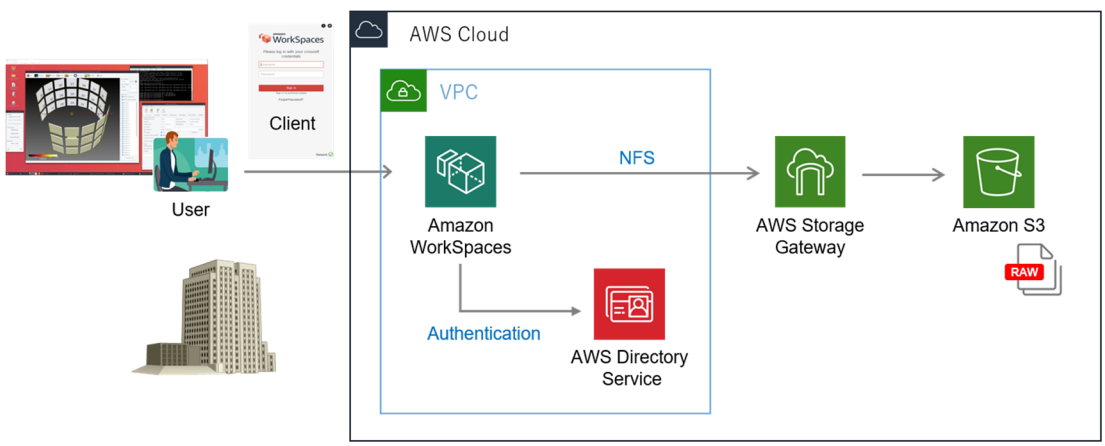

Figure 10: System configuration of the remote desktop environment for data analysis.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUBPL04

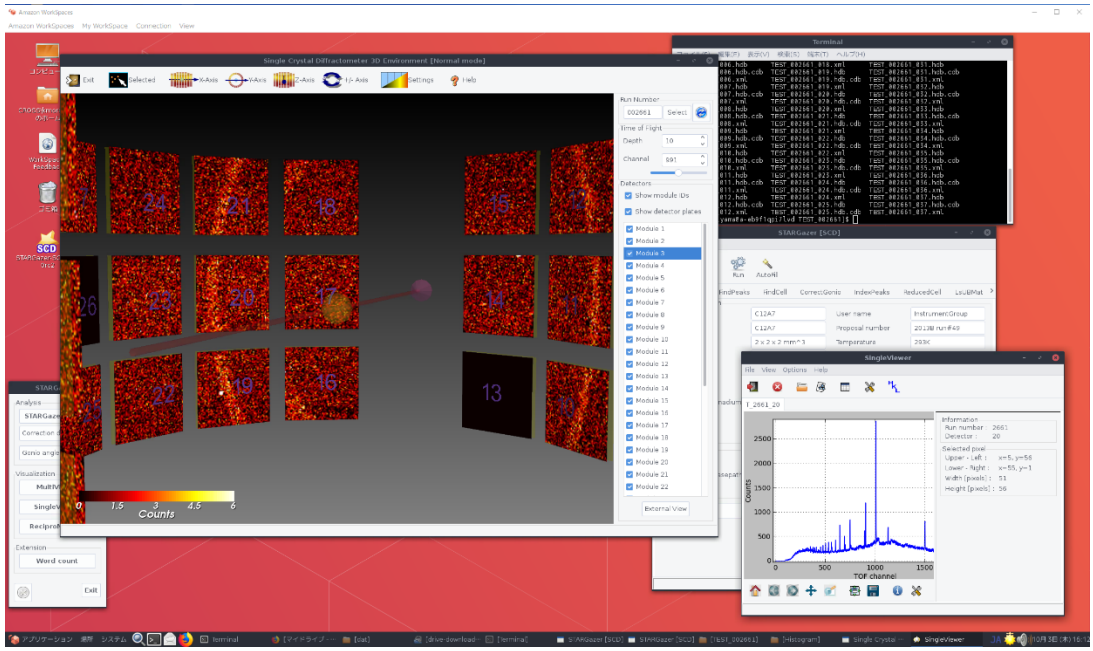

Figure 11: Linux-based remote desktop for data analysis.

REFERENCES

[1] T. Nakatani *et al.*, "IROHA2: Standard instrument control software framework in MLF. J-PARC," in *Proc. New Opportunities for Better User Group Software NOBUGS2016*, Copenhagen, Denmark, 2016, pp. 76-91.

[2] AWS, https://aws.amazon.com/?nc1=h_ls

[3] Docker Platform, https://www.docker.com/

[4] Amazon S3, https://aws.amazon.com/s3/?nc1=h_ls

[5] AWS Lambda, https://aws.amazon.com/lambda/?nc1=h_ls

[6] Amazon API Gateway, https://aws.amazon.com/api-gateway/?nc1=h_ls

[7] Amazon CloudFront, https://aws.amazon.com/cloudfront/?nc1=h_ls

[8] AWS SES, https://aws.amazon.com/ses/?nc1=h_ls

[9] AWS WorkSpaces, https://aws.amazon.com/workspaces/?nc1=h_ls

[10] T. Ohhara *et al.*, "SENJU: a new time-of-flight single-crystal diffractometer at J-PARC", *J. Appl. Cryst.*, vol. 49, part. 1, p.120, Feb. 2016. doi:10.1107/S1600576715022943

[11] K. Moriyama and T. Nakatani, "A Data Management Infrastructure for Neutron Scattering Experiments in J-PARC/MLF," in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 834-837. doi:10.18429/JACoW-ICALEPCS2015-WEPGF060