PaaS Software Based on Cloud Foundry

Yudai Iwasaki, Shunsuke Kurumatani, Tsutomu Nomoto, Takahiko Nagata, and Shinichi Nakagawa

Abstract

NTT is developing platform-as-a-service (PaaS) software based on Cloud Foundry, which is open-source PaaS software. We introduce the advantages of developing applications on a PaaS, the features of Cloud Foundry, and our efforts to put these developments into commercial use.

1. Introduction

The cost and difficulty of developing applications is increasing. Modern ones are becoming connected to the network and required to handle a huge number of users. They depend on various types of middleware and may run on more than 100 servers. In addition, to ensure that they are kept up to date with the changing market, they are also required to be flexible.

NTT is developing platform-as-a-service (PaaS) software[*1] based on Cloud Foundry [1], which is a cloud-computing-based open source software PaaS (open PaaS), to enable us to develop and manage applications more easily, quickly, and flexibly at a lower cost.

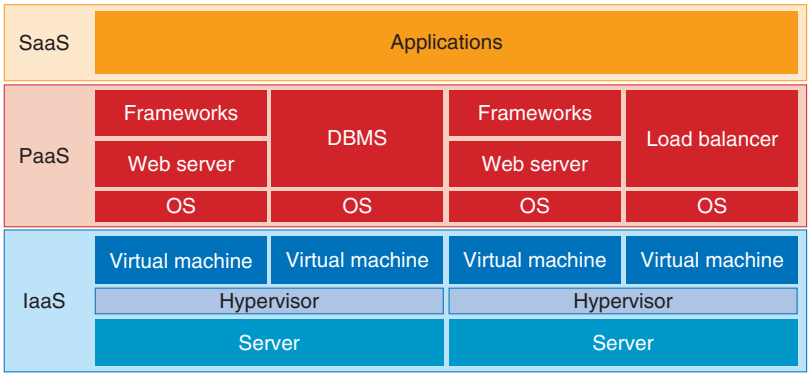
2. Advantages of PaaS for service development

PaaS is a layer on top of the infrastructure-as-a-service (IaaS) layer that provides ready-to-use middleware environments to service developers (**Fig. 1**). This middleware includes operating systems, web servers, database management systems (DBMSs), and application frameworks, and it is essential to run cloud applications. However, with the functionality of modern applications becoming much more sophisticated, middleware environments are also becoming much more complex. This means that efficient management of the middleware is critical to control service costs. However, middleware is commonly shared by various applications. For example, a middleware set composed of a Linux operating system, Apache web server, MySQL DBMS, and Perl (or PHP or Python) programming language is referred to as LAMP and is very popular, especially for web applications. Therefore, a PaaS includes the provision of common middleware environments for service developers and their management for users.

The greatest advantage of using a PaaS as a service environment is the reduced cost and faster speed of service development and management. In addition, a PaaS makes large-scale development easy and does not require such a large initial investment, so it enables a small flexible startup with on-demand scale-up.

During a development, cost reductions are achieved by reducing the number of man-hours required to set up both the hardware and middleware for the services. Without a PaaS, developers must build their application environments by themselves. For example, they must install servers at their datacenter and set up operating systems, DMBSs, web servers, and any other software they need in each machine. This is manageable if there are only a few machines, but if there are over a hundred servers, it becomes a very labor-intensive task. A PaaS enables such simple tasks to be offloaded to the PaaS providers. The only necessary step is to input the required amount of

*1 In this article, we use "PaaS software" to refer to a stack of technologies (also known as a platform) for constructing a PaaS and "application" to refer to a service application that runs on the PaaS. A PaaS user is a user that develops and deploys applications on the PaaS; an application user is a person that uses the application.



OS: operating system
SaaS: software as a service

Fig. 1. Technology stack of cloud computing.

resources into the management console. The environment can be set up immediately with fewer man-hours and less lead-time.

Another benefit during the development process is that a PaaS can offer developers mature tools for building high-performance applications. Creating high-performance scalable applications in a distributed environment generally requires deep and extensive know-how. With a PaaS, however, such know-how is included in the tools (backend systems and software development kit (SDK) or application programming interface (API)). Users can easily develop high-level applications using the provided tools.

A PaaS also allows users to reduce the costs of operating user services. Such services typically require system updates such as patches for bugs and security fixes for operating systems and middleware that have been in operation for an extended time. In addition, server hardware and software must be constantly monitored 24 hours a day, 7 days a week to ensure correct functioning. The PaaS takes over these simple tasks and allows the service provider to deliver complex services with minimal monitoring.

Another reason to use a PaaS is flexibility. It allows computing resources to be consumed on demand. This means that users do not have to continuously reserve sufficient resources to handle peak times such as monthly batch processing or irregular user events. In addition, providers can start services with a minimum number of servers and add additional resources step by step. This is a big advantage when an experimental service is being started.

Several commercial PaaS offerings are in operation, for example, Google App Engine, Sales Force's Heroku and Force.com, and Microsoft Azure. Each has its own characteristics in terms of support for languages, middleware, and frameworks (**Table 1**). For example, Google App Engine provides easy-to-scale middleware, and Heroku supports many programming languages such as Ruby and JavaScript. Force.com differs from the other services in providing useful business logic.

3. Cloud Foundry: open-source PaaS software

Cloud Foundry is open source software developed mainly by VMware in a project that started in 2011. By contrast, most PaaS software such as Google App Engine and Heroku uses closed proprietary software. Since the source code of Cloud Foundry is open, anyone can use it to build their own PaaS. Despite its newness, Cloud Foundry has become a popular type of PaaS software. For example, Active State and AppFog have developed and are selling PaaS solutions based on Cloud Foundry, and Rakuten is using Cloud Foundry for its in-house PaaS. In the NTT Group, NTT Communications is planning to launch a public PaaS based on Cloud Foundry [2].

Cloud Foundry has three main features, which are described below.

(1) No vendor lock-in

PaaS users can avoid vendor lock-in with specific PaaS vendors by using the open-source Cloud Foundry. In general, PaaS offerings differ in usage such as in the API that is provided or the process used to deploy applications. This means that it is difficult to migrate an application developed for a specific PaaS to another PaaS. The migration might require the

Table 1. List of existing PaaS offerings.

Name	Vendor	Source code	Supported programming languages*	Features
Google App Engine	Google	Closed	Java, Python, Go	Easy to develop highly scalable applications using BigTable and related APIs
AWS Elastic Beanstalk	Amazon	Closed	.NET, PHP, Java	Supports various other services provided in AWS
Windows Azure (Cloud Services)	Microsoft	Closed	.net, JavaScript, Java, PHP, Python	Integrates well with Microsoft's products including its integrated development environment
Force.com	salesforce.com	Closed	Apex	Provides various types of business logic
Heroku	salesforce.com	Closed	Ruby, Java, Python, Clojure, Scala, JavaScript	Supports many programming languages
Cloud Foundry	VMWare	Cloud Foundry	Java, JavaScript, Ruby, Scala	Commercial service using Cloud Foundry by VMware
AppFog	AppFog	Cloud Foundry	Java, Python, JavaScript, .Net, Ruby, PHP	Commercial service based on Cloud Foundry
Stackato	ActiveState	Cloud Foundry	Java, Python, Perl, PHP, Ruby, JavaScript, Clojure, Scala, Erlang	Private PaaS solution based on Cloud Foundry
OpenShift	Red Hat	OpenShift	Java, Ruby, JavaScript, Python, PHP, Perl	Red Hat's commercial Paas offering using OpenShift

* Languages listed on official sites. PaaS offerings may support other compatible languages or have restriction for support.

application to be redesigned or rebuilt. This can impose several risks; for example, PaaS users might be forced to accept a price increase, or the service might suddenly be suspended by the provider. In contrast, when PaaS users (i.e., application developers) use a Cloud-Foundry-based PaaS, all of their developed applications are compatible with each other, so they can run them with less risk. They can choose the best PaaS provider at the moment and even build their own private PaaS environment using Cloud Foundry.

(2) Flexible configuration

Cloud Foundry is designed with sufficient flexibility to satisfy a huge variety of needs for PaaS environments. The Cloud Foundry system consists of several components, and users can set up their environment by choosing the necessary combination and number of components for their needs (**Fig. 2**). Cloud Foundry supports a wide range of environments—from a private PaaS on a single server to a huge public PaaS on a cluster of over a thousand servers—and can be set up to suit the scale as well as its reliability and support features. Moreover, users can add components on demand, so it is possible to start with the minimum configuration and increase the scale as the load grows.

(3) Support for languages and frameworks

Cloud Foundry supports most of the major programming languages such as Ruby, Java, and JavaScript. It also supports a variety of popular frameworks used for building web applications; for example, Ruby on Rails, Sinatra, Spring, and Node.js are supported as default. DBMS, MySQL, and PostgreSQL are supported, and other modern forms of DBMS such as MongoDB and Redis are also supported. PaaS users can develop their applications in their own way even in Cloud Foundry's PaaS environment. This makes it easy to introduce a PaaS to the development process with no additional costs.

Another example of an open PaaS is OpenShift Origin. Developed by Red Hat as an open-source version of their PaaS called OpenShift, it was announced on April 30, 2012. At the time of writing, Cloud Foundry has advantages over the younger OpenShift Origin in its more mature quality and market perception. However, Red Hat has a lot of experience and has made significant contributions to open source software development, e.g., to Linux and KVM, which are popularly used for cloud systems. Therefore, we need to monitor further development of OpenShift Origin.

4. Contributions by NTT

NTT is developing its own PaaS software based on Cloud Foundry. Although Cloud Foundry is a promising PaaS software solution that has many good features, as described above, it is still not perfect, especially when used for NTT's PaaS software. For example, its reliability and convenience in commercial usage were not sufficient in the early phase, and

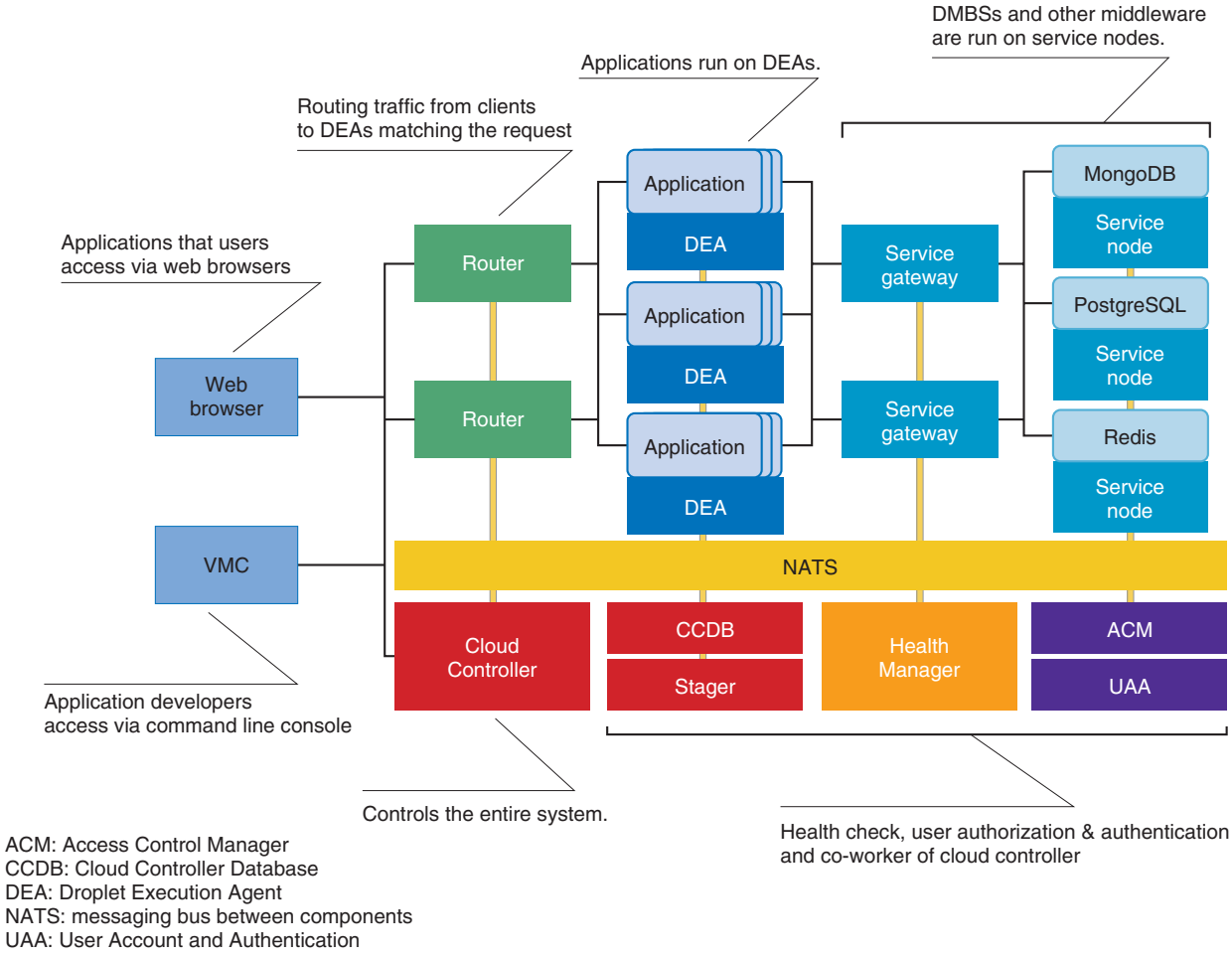


Fig. 2. Components of Cloud Foundry.

integration with other NTT services was naturally not provided by other vendors. In addition, Cloud Foundry is still not in wide use, and this is an obstacle that prevents it from growing as an open source software solution. Therefore, we have been making efforts to improve Cloud Foundry.

4.1 Contributions to Cloud Foundry

(1) Reliability of Cloud Foundry components

Because Cloud Foundry was a very young project when we started our development, it did not have sufficient reliability to be used for a commercial service based on it. Therefore, NTT has been examining the performance and scalability of Cloud Foundry by conducting various tests and fixing any problems revealed by them. For example, we solved a problem involving an important component that was a single point of failure and a problem where some components could not be restored after failures by fixing the source code and adding additional external systems.

(2) Convenience of Cloud Foundry

Since convenience is important for commercial services, we created an installer for the virtual machine container (VMC), which is the console for PaaS users, and we are now developing a function for linking the VMC and version control systems such as Git.

(3) Integration with IaaS software

Although Cloud Foundry has a flexible component system, as described above, to leverage the features, it is essential to integrate it with an IaaS layer under Cloud Foundry. Therefore, NTT is developing a management system that controls both Cloud Foundry and the underlying IaaS systems. This orchestration

system increases competitiveness against other providers because it reduces operating costs and maximizes datacenter utilization, despite using vendor-lock-in-free open source software.

4.2 Contributions to the community

The presence of active communities is indispensable in the long-term evolution of open source software. We launched the Japan Cloud Foundry Group in collaboration with NTT Communications and are fostering a developers' and users' community of Cloud Foundry in Japan [3]. In addition, we have opened the source code created in the first two of our abovementioned contributions (component reliability and convenience) by committing them to the official repository, and we also provide the know-how in workshops.

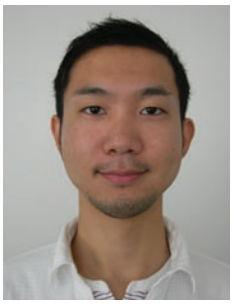
5. Conclusion and future work

Cloud Foundry is promising open-source PaaS software that has many good features. It enables us to develop applications more easily, quickly, and flexibly at a lower cost. However, if we use Cloud Foundry as it is, it will be difficult to compete in the cloud computing market, which is becoming highly competitive. Therefore, NTT is continuously developing its own PaaS software by improving upon the features of Cloud Foundry. In addition, we are investigating possible future integration with other NTT products such as Jubatus [4] and other open source software such as Hadoop*2.

References

[1] Cloud Foundry. http://www.cloudfoundry.com
[2] NTT Communications Press release (in Japanese). http://www.ntt.com/release/monthNEWS/detail/20120627.html
[3] NTT Press release (in Japanese). http://www.ntt.co.jp/news2012/1202/120224a.html
[4] K. Horikawa, Y. Kitayama, S. Oda, H. Kumazaki, J. Han, H. Makino, M. Ishii, K. Aoya, M. Luo, and S. Uchikawa, "Jubatus in Action: Report on Realtime Big Data Analysis by Jubatus," NTT Technical Review, Vol. 10, No. 12, 2012. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201212fa5.html

*2 Hadoop: Java software framework that supports distributed processing of big data.

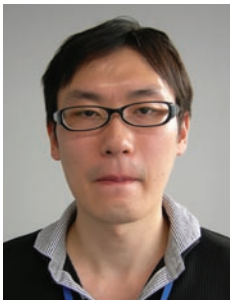


Yudai Iwasaki
Engineer, Platform Technology SE Project, NTT Software Innovation Center.

He received the B.A. and M.M.G. degrees from Keio University, Kanagawa, in 2009 and 2011, respectively. He joined NTT Information Sharing Platform Laboratories in 2009. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center. His research topics are the Semantic Web and cloud computing technology.

Shunsuke Kurumatani
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.

He received the B.A. degree in environmental information from Keio University, Kanagawa, in 2009. Since joining NTT Information Sharing Platform Laboratories in 2009, he has been engaged in R&D of mobile cloud computing technology and an application PaaS. His research interests include web technology, accessibility, and mobile cloud computing. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE).



Tsutomu Nomoto
Engineer, Cloud System SE Project, NTT Software Innovation Center.

He received the B.E. and M.E. degrees in engineering from the University of Electro-Communications, Tokyo, in 2008 and 2010, respectively. Since joining NTT Information Sharing Platform Laboratories in 2010, he has been engaged in R&D of operations management software. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center.

Takahiko Nagata
Senior Research Engineer, Cloud System SE Project, NTT Software Innovation Center.

He received the B.E. and M.E. degrees from the Faculty of Instrumentation Engineering, Hiroshima University in 1993 and 1995, respectively. Since joining NTT Information and Communication Systems Laboratories in 1995, he has been engaged in R&D of high-speed communication processing boards, reliable multicast delivery systems, secure electronic voting systems, and secure file delivery systems. During 2005–2008, he worked in a department supporting the development of systems in NTT WEST. He is currently engaged in R&D of cloud computing technology. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center.

Shinichi Nakagawa
Senior Research Engineer, Cloud System SE Project, NTT Software Innovation Center.

He received the B.S. degree in physics from Saitama University in 1987 and the M.S. degree in physics from Chiba University in 1989. He joined NTT in 1989. He is a member of IEICE.