# Open Source Software Efforts to Transform IoT/AI Services

*Jun-ya Kato, Hitoshi Mitake, Akihiro Suda, Hideki Yamada, Kengo Okitsu, and Sekitoshi Kanai*

**Abstract**

The NTT laboratories have been focusing on expanding the use and development of open source software (OSS), primarily through the NTT Software Innovation Center (NTT SIC). NTT's active contributions to OSS communities such as OpenStack and Apache Hadoop have yielded productive results with refinement of the company's own technologies and training of personnel. Meanwhile, the wave of OSS is steadily spreading to new fields, including virtualization and artificial intelligence. NTT SIC is also engaged in these efforts, and its contributions are beginning to bear fruit. In this article, NTT SIC's work in OSS is introduced from a technical perspective and its community activities are also described.

*Keywords: container, deep learning, parallel distributed processing*

## 1. Introduction

Container-based virtualization technologies (hereinafter, container technologies) have gained attention in recent years, and the opportunities for their use have been increasing. Container technologies stand in contrast to virtual machine technologies from the standpoint of virtualization of the software execution environment. This set of technologies has lower overhead than hypervisor-based virtualization technologies, which places guest operating systems (OSs) in an intermediate layer, so launching and stopping containers is extremely fast.

Improvements to these technologies such as the addition of functions are also happening at a vigorous pace. Because container technologies generate a large quantity of containers within the same environment for an actual use case, and these containers are created and deleted frequently, container management is complex. As a result, open source software (OSS) for container management and execution has been proposed. Docker, Kubernetes, and etcd are introduced in this article as examples of such OSS platforms.

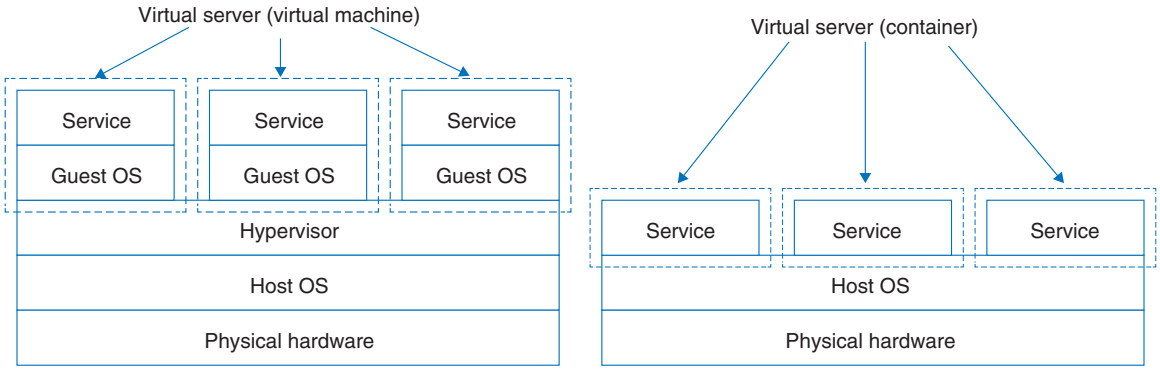
Next, we describe OSS in the field of artificial intelligence (AI). Here, we introduce deep learning frameworks, which are execution environments for deep learning techniques that have rapidly gained widespread use in recent years. Efforts to advance frameworks that support parallel distributed processing have also increased recently due to the growth of large-scale data models and the need to improve processing speeds. In this article, we present an overview of representative deep learning frameworks and examine technologies supporting their use, for example, graphics processing units (GPUs) for parallel distributed deep learning and high-speed interconnects.

## 2. Container technologies

In this section, we present an overview and describe examples of container technologies.

### 2.1 Overview of container-based virtualization

A set of technologies called container-based virtualization has been gaining attention in recent years. With container-based virtualization, only applications needed for services are retained between multiple virtual servers, and the virtual servers share an OS kernel (**Fig. 1**). In contrast to hypervisor-centered hardware emulation-based virtualization, container-based virtualization has little overhead that comes




(a) Hypervisor-based virtualization (b) Container-based virtualization

Fig. 1. Difference between virtual machines and containers.

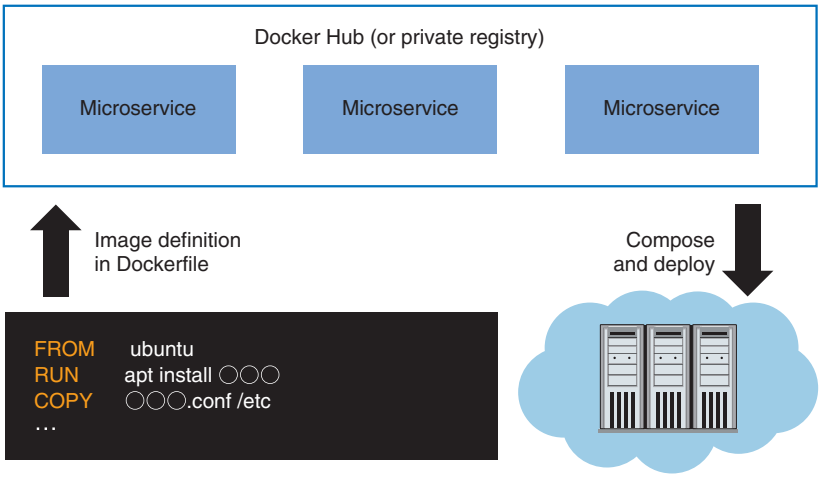



Fig. 2. Structure of Docker.

from the interposition of guest OSs. As a result, starting and stopping containers is extremely fast.

Many kinds of container platform software can build container images by combining the software needed to execute the images. We introduce well-known examples of such software here.

### 2.2 Docker-related efforts

Docker is a well-known container-management and execution platform. It uses a configuration file called Dockerfile that defines the content of the container to build a container image. The image can be distributed on a sharing platform called Docker Hub (**Fig. 2**). With this feature, users can compose images created by themselves or other users to easily build a service with the needed functions. Docker also comes standard with an orchestrator function called Swarm-mode, which enables distributed execution of containers on multiple nodes based on the workload. In addition, container images built with Dockerfile are highly portable. They can also be executed on other orchestrators such as Kubernetes and Mesos.

The only Docker/Moby* maintainer from Japan (as of October 2017) was selected from the ranks of the NTT Software Innovation Center (NTT SIC) in November 2016. Maintainers are developers who

* Docker/Moby: The transition of the Docker project to the Moby project began in April 2017. Product development by Docker, Inc. is based on the results of Moby's open development.

have the privilege of reviewing proposed changes to the source code such as bug patches and new functional additions, and deciding on whether to adopt the proposals. In other OSS communities, a committer is a role equivalent to that of a maintainer.

Maintainers are selected from developers who have regularly contributed to the community by the agreement of current maintainers. The maintainer from NTT SIC has been highly praised for his sustained activities of slightly less than a year, including analyzing and fixing issues related to Docker filesystem drivers and proposing container metadata management functions. Since assuming the role of a maintainer, he has taken on the responsibility of reviewing proposals from other developers and deciding on their adoption, as well as contributing a new functional proposal that shortens the container image generation time. This idea was included in the BuildKit project, which comprehensively rewrites the container image builder functions, in July 2017.

### 2.3 Kubernetes and etcd-related efforts

Kubernetes is a container scheduler for managing clusters on a large scale of several thousand units. Its main role is to deploy containers on resource-appropriate machines in response to task creation requests. Its strengths are its isolation of resources for each task by using container technologies such as Docker, smooth distribution of applications, and flexible scheduling based on the priority of each task.

The frequency of generating events such as notifications of user requests and task completions depends on the purpose and size of the cluster. In an enormous cluster, events are generated at high frequency. Storing the states of tasks in high-reliability and high-performance databases is therefore essential for stably operating a large-scale cluster.

A well known and leading OSS for storing task states is etcd. It is a distributed key value store that uses the Raft distributed consensus protocol. Its features are high availability and strong consistency. Kubernetes uses etcd for saving internal management information.

NTT SIC has been recognized for its sustained efforts to improve etcd, including improving product quality by fixing bugs related to etcd's Raft component and improving its authentication function at the design level. As a result, a maintainer of the etcd project was selected from NTT SIC in June 2016. There are only seven maintainers of the etcd project worldwide (as of October 2017).

In addition, Namazu, a testing tool for distributed systems developed by NTT SIC, is being used for etcd product improvement. At present, it is being incorporated into the official CI (continuous integration: automatic test environment) of the etcd project.

NTT SIC is also making efforts to expand the applicability of Kubernetes by working on the development of a scheduler that takes into account network topology and to provide support for remote direct memory access (RDMA), a low-latency and central processing unit (CPU)-efficient network protocol. These technologies are critical for deep learning frameworks, described below.

## 3. Deep learning frameworks

Here, we describe the current situation regarding deep learning frameworks and also touch on expected future trends.

### 3.1 Existing frameworks

The growing popularity of deep learning in recent years has led to the release of deep learning frameworks as OSS (**Table 1**). Deep learning methodology learns from data the parameters of a defined model (network structure used for deep learning) in response to a task. With this trained model, data such as images and speech can be recognized with high accuracy. Through the learning process, a great number of derivatives within the model are calculated and used. However, implementing the computational process manually each time a task or model is modified becomes extremely cumbersome. A framework has functions for automatically carrying out the computational process necessary for learning. Users can use deep learning simply by defining the model to be used and the learning method.

Companies have led the development of deep learning frameworks. For example, outside Japan, TensorFlow has been developed by Google, Microsoft Cognitive Toolkit (CNTK) by Microsoft, and MXNet by Amazon. Facebook has been developing Caffe2 as a framework for commercial development and released PyTorch as a research-oriented framework. TensorFlow's main benefit is that it has the most users and the largest community of all the frameworks. CNTK's strength is considered to be its fast performance and high scalability, and for MXNet, its flexibility in implementation. In Japan, Preferred Networks (PFN) has released Chainer, which provides support in Japanese. Chainer also uses a flexible scheme in which the model structure is automatically determined as data are processed. This scheme has spread to other

Table 1. List of deep learning frameworks.

| Framework | Platform | Interface | Features | Main developer |
|---|---|---|---|---|
| TensorFlow | Linux, macOS, Windows, Android, iOS | Python, Java, C++, Go, Julia, C#, R, Haskell | Largest community; Can visualize with TensorBoard | Google |
| MXNet | Linux, macOS, Windows, Android, iOS | Python, R, Julia, Scala, Go, (Matlab, Javascript) | Can be flexibly implemented to suit the task; Numerous platforms | Amazon, Baidu, Carnegie Mellon Univ. |
| CNTK | Linux, Windows | Python, C++ | Fast (especially RNN); High scalability | Microsoft |
| Chainer | Linux | Python | Effective implementation schemes for natural language processing | PFN |
| PyTorch | Linux, macOS | Python | Effective for natural language processing derived from Torch and Chainer | Facebook, Twitter, Salesforce |
| Caffe2 | Linux, macOS, Windows, Android, iOS | C++, Python | Numerous platforms; High-speed, memory-efficient, high scalability | Facebook |

frameworks.

Support for parallel distributed processing by frameworks is advancing. In Japan, Chainer is making headway in providing similar support. PFN has released an additional package for Chainer called ChainerMN, which supports distributed learning. The company claims that the product is six times as fast as TensorFlow (as of February 2017).

None of these frameworks has emerged as the decisive winner in terms of the work environment or processing capabilities. NTT SIC is therefore continuing to test each framework in accordance with a variety of use purposes.

### 3.2 Technological trends and challenges going forward

(1) Parallel distributed processing using high-performance computing resources

To improve the accuracy of deep learning, it is said that in general, the model must have deep layers. Consequently, however, the number of parameters to learn increases and the computational complexity explodes. The problem arises in which computations require several days to several weeks. The use of high-performance computing resources, which boast superior computational capabilities, and parallel distributed processing, which executes multiple computations in parallel, is being investigated in order to solve this problem. Frameworks are being extended to support this trend.

The use of GPUs as high-performance computing resources is becoming widespread. GPUs are especially strong when it comes to matrix operations, which make up almost all deep learning computations. At present, products from NVIDIA, which have become the de-facto standard, feature computational performance that is several times to several tens of times greater than that of CPUs. Several frameworks use cuDNN, a library provided by NVIDIA, to describe computation for their learning component. Thus, we see an example of the use of high-performance GPUs.

Parallel distributed processing techniques mainly use a method called data parallelism, which makes use of multiple GPUs. Data parallelism is a method in which the same model is copied to multiple GPUs, and different parameter learning computations are executed on the learning data in parallel. The parameters are then updated based on communications from each GPU. Vast improvements in speed can be expected by parallelizing parameter learning, for which until now learning data had been input into one GPU sequentially. As a result, active efforts are underway to support each framework. Work is also being carried out to provide multi-GPU support in one machine and multi-node support using GPUs in multiple machines.

We can thus expect to further improve speed by increasing the number of machines and GPUs. However, it is known that as parallelism increases, communication to update parameters becomes the performance bottleneck. Research on how to skillfully arrange parameter data and apply high-performance communication techniques has begun for each framework. The discussions and implementations are expected to be energetic from here on.

Going forward, NTT SIC plans to study which architectures are the most optimal for actual problems.

(2) Communication processing technologies to support parallel distributed processing

The use of high-speed interconnects such as InfiniBand, widely used in the field of high performance computing (HPC), is considered effective as a high-speed communication approach. RDMA is used instead of general Transmission Control Protocol/Internet Protocol (TCP/IP) to achieve low-latency communications when using high-speed interconnects to their maximal limits. RDMA is capable of reading/writing the memory of remote machines without CPUs as the intermediary. General communication programs are described using socket application programming interfaces (APIs). However, RDMA communication programs use low-level descriptive APIs called verbs and Message Passing Interface (MPI), a parallel computing interface widely used for HPC.

RDMA support for TensorFlow is steadily moving forward. RDMA verb communication was implemented in April 2017 [1], and GPU Direct RDMA was implemented in August 2017 [2]. Additionally, distributed processing using MPI for inter-node communication is being developed for ChainerMN [3]. Partial implementation and discussions in communities have also begun for MXNet and CNTK [4]. In the near future, RDMA-related activities will become even more critical.

Because RDMA communication procedures are more complex in actual use than general TCP/IP, NTT SIC is presently studying ways to simplify implementation by abstracting APIs in order to popularize RDMA communication.

## 4. Future development

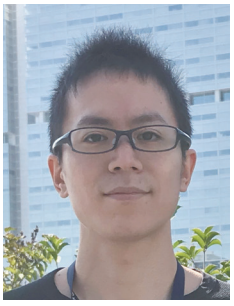
In this article, we introduced new OSS efforts being carried out by NTT SIC in the areas of container-based virtualization technologies and deep learning frameworks. Because virtualization is a required technology for system design and architecture, the application of lightweight virtualization container technologies will continue to expand going forward. Also, deep learning frameworks will advance in their ability to handle parallel distributed processing.

The NTT laboratories are continuing to advance research and development related to these technologies. At the same time, the labs are focusing on using their own approaches. For example, the NTT laboratories are building parallel distributed deep learning frameworks on container clusters and evaluating the frameworks during actual use. These efforts not only contribute to AI research, but are also expected to be applied to a wide range of other problems.

## References

[1] ibverbs-based RDMA path, https://github.com/tensorflow/tensorflow/pull/8943
[2] GPU Direct RDMA Out-of-Band Tensor Transport, https://github.com/tensorflow/tensorflow/pull/11392
[3] ChainerMN distributed deep learning frameworks (in Japanese), https://research.preferred.jp/2017/05/chainermn-beta-release/
[4] CNTK v.2.0 RC 2 Release Notes, https://github.com/Microsoft/CNTK/wiki/CNTK_2_0_RC_2_Release_Notes

**Jun-ya Kato**
Senior Research Engineer, Distributed Computing Technology Project, NTT Software Innovation Center.
He joined NTT Information Sharing Platform Laboratories in 2002, where he studied next-generation Internet protocol IPv6. He joined the architecture design project for IPv6 Internet service provider (ISP) support on NGN (Next Generation Network) in 2008. He also technically supported NTT operating companies participating in World IPv6 Day and World IPv6 Launch, which were global IPv6 deployment events in 2011 and 2012, respectively. He then moved to the Technology Development Department, NTT Communications, where he developed network security services such as DDoS (distributed denial of service) protection in the ISP backbone. His current research is focused on software-defined networking and distributed computing platforms.



**Hitoshi Mitake**
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received a B.E. and M.E. from Waseda University, Tokyo, in 2010 and 2012. He joined NTT in 2012 and has been working on distributed storage systems and cluster management systems.

**Akihiro Suda**
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received a B.E. and a Master of Informatics in computer science from Kyoto University in 2012 and 2014. He joined NTT in 2014 and has been a maintainer of several kinds of container-related OSS such as Moby (formerly Docker), Moby BuildKit, and CNCF containerd.

**Hideki Yamada**
Research Engineer, Distributed Computing Technology Project, NTT Software Innovation Center.
He received a B.E. and M.E. from Tokyo Institute of Technology in 2005 and 2007. He joined NTT Information Sharing Platform Laboratories in 2007. His recent research area is distributed storage.

**Kengo Okitsu**
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received a B.S. and M.S. in computer science from the Tokyo Institute of Technology in 2008 and 2010. He joined NTT in 2010 and has been studying cloud resource management and scheduling.

**Sekitoshi Kanai**
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received a B.E. and M.E. in physics and physico-informatics engineering from Keio University in 2013 and 2015. He joined NTT in 2015 and has been studying deep learning algorithms.