NTT's Contributions to OSS Upstream First Development

Kota Tsuyuzaki and Takeshi Yamamuro

Abstract

Open source software (OSS) has been used in various information technology systems in recent years. OSS is developed by the collaborative OSS community, which consists of many companies and individuals. Additionally, several large cloud service providers such as Google and Facebook have released their own software used in their services as open source to continue to improve the software in the OSS community. We explain here the advantages of companies developing software as OSS and the significance of companies' contributions to OSS communities. In addition, we also describe two examples of the NTT laboratories' involvement in OSS communities.

Keywords: OSS, OpenStack, Apache Spark/Hivemall

1. Upstream first

Open source software (OSS) is computer software whose source code has been made available with a license in which the copyright holder gives anyone the right to freely change and distribute the software for any purpose. OSS is continually upgraded and enhanced as features are added, and its bugs are fixed through an ongoing collaborative effort by the OSS community. In recent years, we have seen a growing tendency for major companies such as Google, Facebook, and IBM to make their own software available as OSS.

Companies that adopt OSS can do so using two strategies: utilizing the software as-is or developing new features to better meet the companies' needs. In both cases, a significant new approach called *upstream first* [1] has emerged. The idea of upstream first is to enable the companies' own developers to work together with the upstream developers and build relationships with the upstream community to improve the OSS. When the first strategy of utilizing OSS as-is is applied, it is beneficial to follow the latest version of the software as upstream first, as it ensures the best quality of the software. With the second strategy of building additional functionalities into the OSS, the upstream first approach is effective in avoiding conflicts between source code lines added by company developers and those of the community version of the OSS. We explain these two strategies more closely using real examples.

1.1 First strategy: following the latest version

Most software packages developed recently must be updated periodically with the latest version to maintain the software quality. While these changes are mostly minor, they sometimes address serious issues such as security vulnerabilities. Most OSS today employs shared identifiers called Common Vulnerabilities and Exposures (CVE)[*1] [2] to manage the impact of vulnerabilities and how they can be corrected. Fixing vulnerabilities typically involves a three-step process: (1) the party having discovered the vulnerability issues a confidential report detailing the vulnerability to the security contact team; (2) core developers in the security contact team devise a fix for the vulnerability working in a closed environment; (3) and finally, the fix is made available, and the CVE is announced to the general public. Therefore, when a company lets its own developers join the

*1 CVE: The standard for information security vulnerability names for managing vulnerability fixes, identifying publicly known vulnerabilities with standard identifiers, and notifying users.

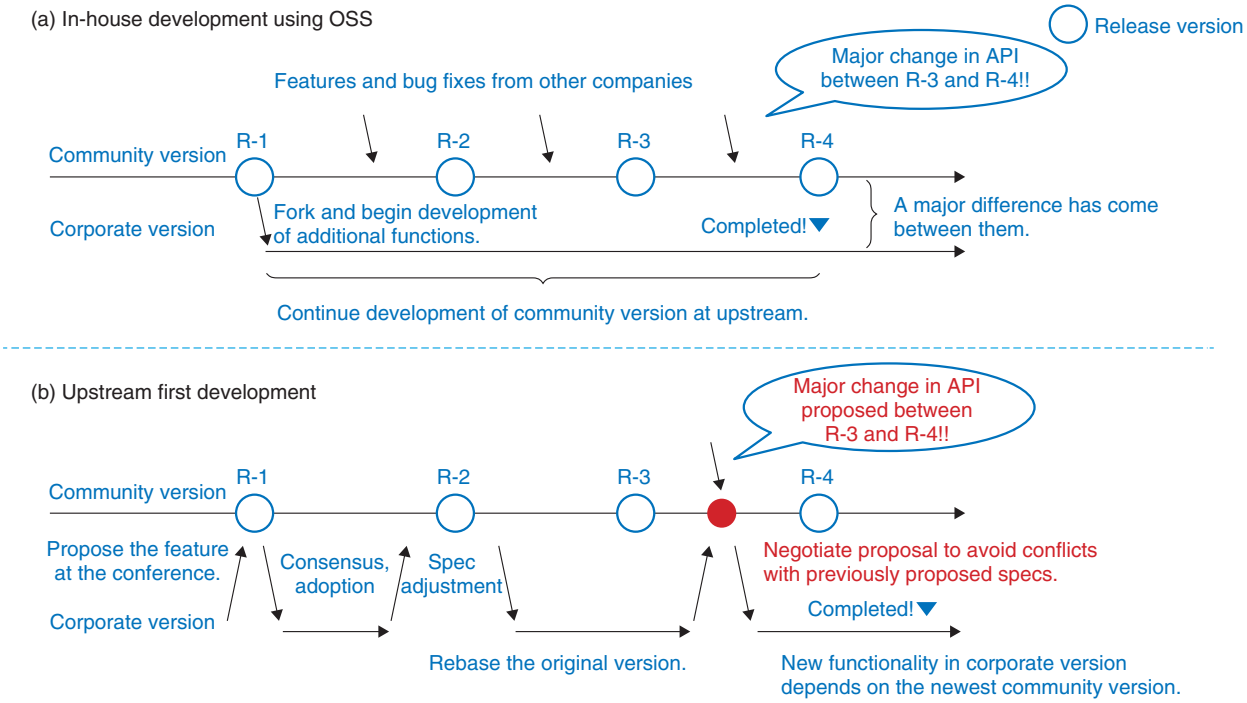


Fig. 1. Difference between in-house development using OSS and upstream first approach.

security contact team responsible for developing the OSS, it brings the company—as a stakeholder—on board so that members are clearly aware of the security vulnerability and can help come up with a solution that fixes the vulnerability when the issue is disclosed. Note that if there is an insufficient number of core developers working on a solution in step (2), the progress made during that process will be slow in addressing the vulnerability. Therefore, taking the lead in supporting the core developers of the OSS ultimately benefits the company, as it helps to safeguard the security of their own corporate systems.

1.2 Second strategy: avoiding conflict with the community version

There may be instances when a company would like to use OSS, but the software lacks some critical capability that the company requires. When this capability is truly indispensable, the quick and easy solution might be simply to develop the features in house. In this approach (**Fig. 1(a)**), the company forks the community software and then develops its own features for its product. However, because OSS is developed by an open community, the application programming interface (API) specifications can be changed in a specific time period. In order to manage the synch with the latest community version, the company has to constantly upgrade that portion of the software that it modified. This is not always easy, especially in the case of OSS communities that are very active, where the portion to upgrade could include tens of thousands or even hundreds of thousands of changed lines of code. Maintaining both the upstream changes and the in-house code would incur an enormous cost much greater than just keeping the firm's own in-house modified version of the software up and running.

The way to avoid this situation is to develop the features as OSS community code and give it back to the community. Software modification costs can be held down by approaching the requests early on at the initial idea stage. This approach may reduce the instances of the software conflicting with other features proposed by other companies in the community. These activities make it possible to minimize differences between a company's added features and the community version of the software that includes significant bug fixes, and it will make it far easier to integrate upstream work into a company's system. Moreover, developing engaging functional features as OSS will enable the feature to be enhanced by other companies. This kind of collaborative approach

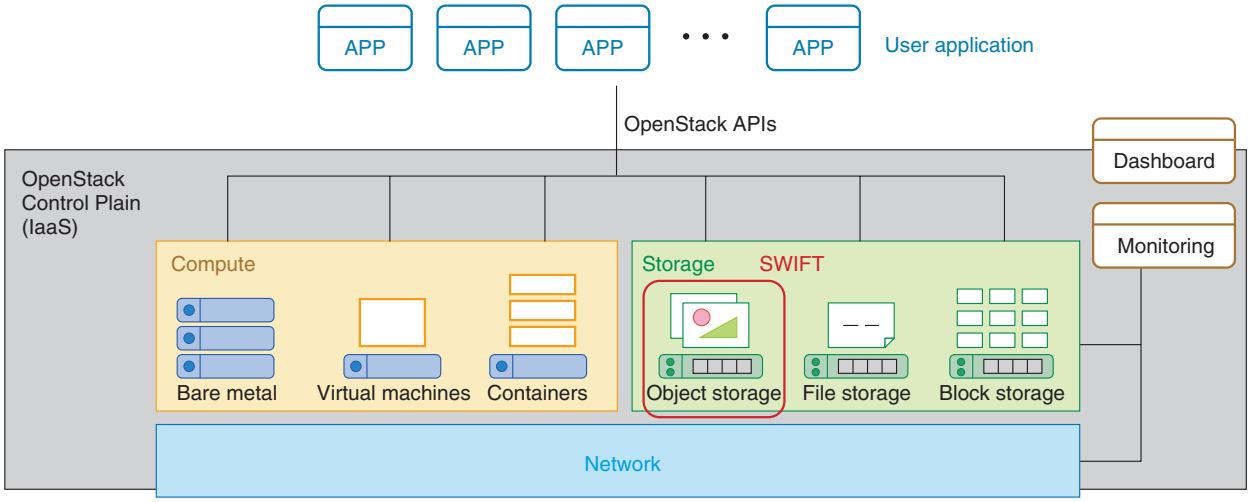


Fig. 2. OpenStack components and Swift.

to OSS development that minimizes conflicts among software features in advance is the way of upstream first, and it results in OSS products that benefit the entire community of OSS users (**Fig. 1(b)**).

This is why companies practice upstream first, develop new software features in collaboration with OSS communities, and increasingly enjoy the advantages of software quality assurance. The NTT Group is involved in the upstream first approach through active participation in several OSS communities including Linux, OpenStack, and Apache. In the next two sections, we describe how the NTT Software Innovation Center accomplishes an upstream first approach in dealing with two open source projects: OpenStack Swift and Apache Spark/Apache Hivemall.

2. OpenStack Swift

OpenStack [3] is an OSS platform maintained by the OpenStack Foundation for implementing infrastructure-as-a-service (IaaS)[*2] solutions. OpenStack consists of several components; an overview is shown in **Fig. 2**. Development policies for each component are discussed at two development conferences that are held biannually: OpenStack Summits, with current attendance in excess of 5000 people, and Project Team Gatherings (PTGs)[*3]. Day-to-day development work is conducted by developers in the globally distributed community via Internet Relay Chat (IRC) and mailing lists. OpenStack is released around six-month cycles, a significantly shorter cycle time than products released by other OSS communities.

OpenStack Swift [4] is an OpenStack component that implements storage technology called *object storage*. Swift object storage provides certain features to ensure data durability: data are stored redundantly; redundancy is monitored within the system; and Swift works to automatically replicate the data in case of a hard drive failure to avoid loss of data. OpenStack Swift has been robust and stable since its release in the first rollout of OpenStack in 2010. Consequently, Rackspace, one of the major players of OpenStack, initially employed the project as a commercial offering.

NTT Group companies have also conducted several case studies on OpenStack [5]. In case studies using OpenStack Swift, the NTT labs have proposed various functional improvements and bug fixes and have made other contributions to the community. Having gained the trust and respect of the community through these activities, the NTT labs have built a community action team that includes one of the core developers in the OpenStack Swift community, and are actively involved in OSS research and development (R&D).

Here, we highlight the NTT labs' involvement in the erasure coding scheme[*4] for OpenStack Swift, an area where NTT's input has been especially influential. OpenStack Swift released a powerful new data

*2 IaaS: Infrastructure as a service, a form of cloud computing that provides virtualized computing resources over the internet.

*3 PTG: The Project Teams Gathering is an event held every six months where OpenStack developers decide the content of the next development cycle.

storage scheme called *erasure coding* in 2015. This new scheme helped to significantly reduce the cost of deploying Swift at NTT Group companies.

The NTT labs have experimented with the upstream first strategy since the beginning of development, and they took the lead in proposing fixes and features to the OpenStack community. For example, we contributed a significant bug fix [6] to the erasure code library. Although the bug was not a security vulnerability, it was a library error that could have resulted in a catastrophic loss of stored data. The NTT labs discovered it during the verification process. Our community action team with a core developer analyzed the problem immediately, quickly verified the library bug responsible, and came up with a fix [7]. These efforts not only preserved the quality of our own products, but also helped save considerable data of many other users in the OpenStack Swift community.

In another case involving the development of features, NTT made a major contribution through its development of *globally distributed erasure coding*. When disaster recovery (DR) clusters[*5] are built with OpenStack Swift across multiple datacenters, global cluster functionality is used to ensure high performance. However, this presents a serious problem when this functionality is used together with an erasure coding scheme; it does not guarantee there will be sufficient data stored in a datacenter, and there is a risk that data might be unavailable if the network connection between datacenters is blocked or interrupted.

The NTT labs contributed the globally distributed erasure coding scheme to resolve this issue. A stable version of the scheme was released in August 2017 just two years after it was first proposed in August 2015, and it was highlighted as a major new functional capability in the new features in the OpenStack community [8]. During the two-year development period for this new scheme, roughly 180,000 lines of code were altered for OpenStack Swift overall, but OpenStack Swift was able to smoothly merge the 15,000 lines of code in the globally distributed erasure coding due to the application of the upstream first strategy in the initial idea proposal and the code development. Furthermore, as the result of upstream first development, the OpenStack Swift community is now considering further work on the feature to make it more convenient and efficient, and this will also benefit NTT Group companies.

Illustrating the use case, making it beneficial as the core technology, and achieving upstream first will enable other companies to join the development, and it will result in the growth of the software, the code, and even the community much like a living thing. Of course, this greatly benefits NTT as well.

It is apparent from these examples that it is important to maintain a healthy OSS community in order to develop new functional capabilities and achieve high quality. Needless to say, this is all vitally important to ensure further quality improvement of an individual company's own products.

3. Apache Spark/Apache Hivemall

Apache Spark [9] is an OSS distributed parallel processing framework developed by AMPLab, a research group at the University of California, Berkeley. Apache Spark includes not only a core component for distributed parallel execution, but also many useful libraries for SQL, machine learning, and streaming, as shown in **Fig. 3**. Apache Hivemall [10] is an OSS distributed machine learning library accepted as an incubator project by the Apache Software Foundation (ASF) in October 2016. Makoto Yui at Treasure Data, Inc. is an original author of Hivemall; NTT developers also became involved in the community activity at the beginning of 2016. Finally, the developers both at Treasure Data and NTT proposed Hivemall to the ASF. Apache Hivemall can be used on widely used distributed processing frameworks such as Spark. Hivemall incorporates state-of-the-art distributed machine learning algorithms, which is different from existing machine learning libraries.

The Spark community is growing rapidly, and over 3000 participants from all over the world attended the recent Spark Summit held in June 2017. NTT is also involved in the development and is committed to the community. For example, it has enhanced APIs for porting Hivemall functions into Spark and has made a number of bug fixes and improvements in Catalyst. Catalyst is an optimizer component that affects many applications running on Spark, and it will likely play a significant role in NTT Group's use cases in the future. Therefore, we have stepped up efforts to fix bugs and enhance the performance of Catalyst. The

*4 Erasure coding scheme: A reliable data protection scheme implemented in Swift that reduces the amount of data stored on physical disks. Original data are segmented into fragments and encoded with redundant data pieces and stored across a set of different locations or storage media.

*5 DR cluster: A disaster recovery cluster is a system configuration in which hardware is deployed at multiple geographically dispersed sites, so if one (or several) datacenters fail due to a natural disaster or other unforeseen event, the overall system remains up and running.

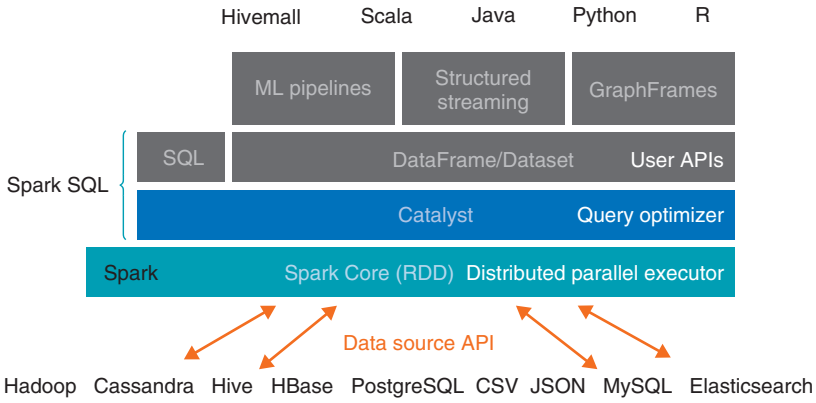


Fig. 3. Spark and Hivemall software stack.

Spark community recently took a step forward to support the use of graphics processing units (GPUs) in a resource scheduler. Spark currently only uses central processing unit (CPU) cores as resource scheduling units; a job scheduler splits a user query into tasks and assigns them to CPU cores. With the increasing popularity of deep learning and its need for computing resources, it is important to officially discuss GPU support in Spark native components.

The development of Apache Hivemall is progressing as Makoto Yui and other developers continue to implement a range of sophisticated machine learning algorithms and utility functions for feature engineering. In these activities, NTT is responsible for porting Hivemall functions into Spark and implementing useful and efficient functionalities that users actively request and Spark does not provide. Some of these functionalities are not implemented by Spark because they do not comply with the Spark development policy or are too complicated to implement as first-class Spark functionalities. With the valuable experience we have obtained through these OSS activities, we are currently trying to provide a scalable and highly efficient distributed parallel framework with Spark and Hivemall.

We believe that being involved in OSS communities such as Spark and competing with distinguished developers in the world will not only help us keep up with existing OSS products but will also open the way to discover innovative next-generation technologies that will emerge in future. We are now moving forward to apply Spark and Hivemall into realistic use cases with NTT Group companies. We plan to develop robust technologies that are widely used by NTT Group companies while building complementary relationships with OSS communities.

4. Future work

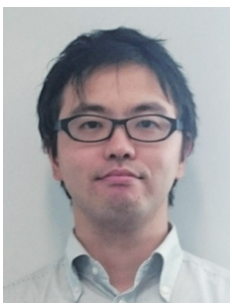
This article highlighted two major OSS projects that illustrate NTT's deep involvement in a range of worldwide efforts through its active participation in various OSS communities. OSS development is growing not simply because companies are eager to adopt free software for their own private purposes, but rather because involvement in such communities provides a way for companies to achieve success and grow with the communities. The NTT labs are leveraging open source through an upstream first approach in OpenStack, Apache Spark, Apache Hivemall, and other OSS projects. We are committed to ongoing R&D that contributes to NTT Group companies while benefiting the OSS communities to which we belong.

References

[1] R. Bryant, "OpenStack: Upstream First," Aug. 2016. https://www.slideshare.net/tesoracorp/openstack-upstream-first
[2] CVE, https://cve.mitre.org/
[3] OpenStack, https://www.openstack.org/
[4] Swift, https://www.openstack.org/software/releases/ocata/components/swift
[5] Press release issued by NTT DATA on January 15, 2015 (in Japanese). http://www.nttdata.com/jp/ja/news/release/2015/011500.html
[6] K. Tsuyuzaki, Bug report, OpenStack Object Storage (swift), Bug #1639691, Nov. 2016. https://bugs.launchpad.net/swift/+bug/1639691
[7] Corrupted fragment report, https://github.com/01org/isa-l/issues/10
[8] Press release issued by OpenStack on August 30, 2017, "OpenStack Pike Delivers Composable Infrastructure Services and Improved Lifecycle Management." https://www.openstack.org/news/view/340/openstack-pike-delivers-composable-infrastructure-services-and-improved-lifecycle-management

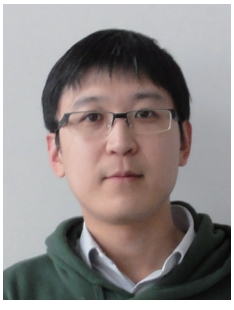
[9] Apache Spark, https://spark.apache.org
[10] Apache Hivemall, https://hivemall.incubator.apache.org

Trademark notes



Kota Tsuyuzaki
Software Engineer, Cloud Solution Project, NTT Software Innovation Center.

He received an M.E. in information engineering from Waseda University, Tokyo, in 2010. Since joining NTT in 2010, he has been developing distributed storage systems. He has been working on OpenStack Swift for approximately five years and has been a member of the Swift Core Team since June 2015.



Takeshi Yamamuro
Research Engineer, Distributed Computing Technology Project, NTT Software Innovation Center.

He received an M.E. from the Faculty of Science and Technology, Sophia University, Tokyo, in 2008 and joined NTT the same year. He has since been working on database management systems. His research interests include compression and hardware-aware algorithms (e.g., SIMD, NUMA and GPU). He received the IPSJ Yamashita SIG Research Award from the Information Processing Society of Japan (IPSJ) in 2015. He is a member of IPSJ and the Database Society of Japan.