Jubatus in Action: Report on Realtime Big Data Analysis by Jubatus

Keitaro Horikawa, Yuzuru Kitayama, Satoshi Oda, Hiroki Kumazaki, Jungyu Han, Hiroyuki Makino, Masakuni Ishii, Koji Aoya, Min Luo, and Shohei Uchikawa

Abstract

This article revisits Jubatus, a scalable distributed framework for profound realtime analysis of big data that improves availability and reduces communication overheads among servers by using parallel data processing and by loosely sharing intermediate results. After briefly reviewing Jubatus, it describes the technical challenges and goals that should be resolved and introduces our design concept, open source community activities, achievements to date, and future plans for expanding Jubatus.

1. Introduction

There is little doubt that data is of great value and importance in databases, data mining, and other data-centered applications. With the recent spread of networks, attention is being focused on the large quantities of a wide variety of data being generated and transmitted as *big data*. This trend is accelerating [1] as a result of advances in information and communications technology (ICT), which simplifies the collection and analysis of big data.

Even now, there is a strong need to make new discoveries and find patterns that were not noticed before, by taking huge amounts of data in a certain area that has been stockpiled, such as a dozen or so years of clinical data, and analyzing it from all angles, as one scenario in which big data is put to good use in business. This trend is not limited to operations confined to a specific area; it includes the possibility of swiftly finding new business possibilities or synergistic effects by focusing on relationships in big data spanning different fields or specialist areas. We think that this trend will expand into cross-domain big data analysis in the future[*1].

There are two major types of big data and techniques for analyzing it.

(1) Stockpile type: Lumped high-speed analysis of accumulated big data (batch processing)
(2) Stream type: Sequential high-speed analysis of data stream being continuously generated without accumulation (realtime processing)

With case (2) in particular, the ambiguity inherent in the environment is creating a growing need to make judgments and decisions on the basis of insufficient information.

In this article, we revisit Jubatus, a framework intended for realtime stream-type big data analysis that provides profound analysis ability by online machine learning as added value. Jubatus was introduced in the June 2012 issue of NTT Technical Review [1]. That article focused on a key mechanism called *mix*. In this article, after briefly reviewing Jubatus, we describe the technical challenges and goals that should be resolved and introduce our design concept, open source community activities,

*1 Cross domain: Access to data spanning different domains. http://en.wikipedia.org/wiki/Cross-domain_solution

achievements to date, and future plans for expanding Jubatus.

2. Jubatus

2.1 Overview

Jubatus is a scalable distributed computing framework for online machine learning. The origin of the name is the Latin term for that agile animal the cheetah. It is pronounced "yu-ba-tus". Developed jointly by Preferred Infrastructure Corporation and NTT Software Innovation Center, it is currently published on websites as a Japan-originated big data open source project [2]–[4].

The goal of Jubatus is to enable swift and profound analysis of stream-type big data. One example of its use is as a social media application for automatically classifying the huge number of tweets[*2] (over 8000 per second) generated all over the world. This processing includes the three requirements: large volume, fast, and profound. In other words, it supports natural language processing and automatic multicategory classification, at high speed without lag, of a data stream of 16 Mbit/s.

However, these three requirements basically have a trade-off relationship and it is intrinsically difficult to satisfy all of them simultaneously. Jubatus satisfies both profound analysis and scalability. Here, profound analysis is the automatic categorization of unstructured information intended for human beings, such as natural language. It can also replace human labor for unclearly formulated processing work, such as recommendation, prediction, and relationship discovery. From the technical perspective, it involves challenges in the areas of machine learning, artificial intelligence, and pattern recognition.

On the other hand, scalability encompasses the issues of (1) increases in processing requests and (2) increases in data size. Issue (1) can be further divided into throughput (volume of requests to be handled per unit time) and response (response for each instance, without lag). In general, batch processing focuses on throughput while realtime processing focuses on response. Approaches to issue (2) are either processing the data without waiting or dividing and storing it.

Jubatus answers the need for making prompt judgments in situations exhibiting uncertainty and ambiguity. After our comprehensive investigation of the design concept, under which information for judgment is being collected continuously and decisions are being made without lag, we separate the profound analysis (functionality) from the scalability (non-functionality). More specifically, a profound analysis design likens the logic of online machine learning to an *engine or central processing unit* (CPU), which can be continuously upgraded as removable analysis modules. A scalable design is seen as a common infrastructure *chassis or motherboard*, which can be scaled by installing analysis modules into the common framework. The ultimate goal of Jubatus is to provide everyone with scalable machine learning. Our policy is to provide a broadly easy-to-understand online machine learning framework for big data that is easy to use, with hardware that scales out over inexpensive commodity servers to enable massively parallel distributed processing and software that is not restricted to a few data scientists, programmers, and specialists.

2.2 Applicable areas

Google's PageRank is well known as a technique for calculating the importance of web pages from all over the world [5]. Since website link structures are updated comparatively slowly, batch analysis is suitable in this case. Social media such as Twitter tweets, on the other hand, are characterized by having finer granularity than the web, by having light content, by having small chunks of data that are transmitted in large numbers from a wider demographic than that of web users, and by the immediacy of information being vital. Through social media, we can analyze how a broad user population reacts to changes and events in the external environment and make effective practical use of realtime analysis of stream-type big data, with cases applied to business enterprises.

We have provided primary classifications of Twitter information as an application of Jubatus. Since the information representation is in natural language and is presented with a limited number of characters, expressions having distinctive omissions, abbreviations, coined words, jargon, and mannerisms were included in the analysis subject matter. From the over 2000 tweets per second in the Japanese language alone, it is difficult for humans themselves to extract and organize useful information instantly by sight. Therefore, general-purpose primary filters that provide major classifications and refining functions are useful. Using the approximately 1600 companies listed on the first section of the Tokyo Stock Exchange as the classification categories, Jubatus speedily

*2 Tweets: short messages sent using the popular social networking service and microblogging service Twitter.

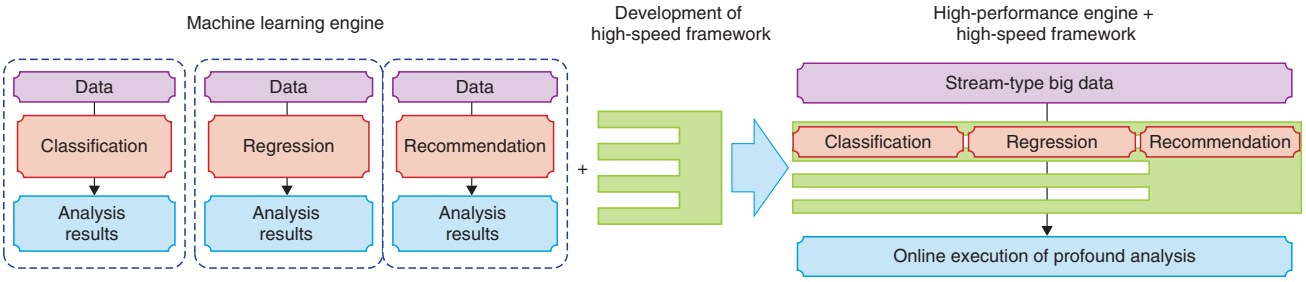


Fig. 1. Outline of the design concept.

classifies the 2000 tweets per second into their corresponding business categories and supplies information to an analysis application. This implementation uses an online machine learning technique called Jubatus classifiers (multi-valued classification). We used highly trustworthy published information, such as Wikipedia, as training data and automatically constructed a learning model for classification. Unlike item classification by keyword matching or related words, we use online learning to update the vector equations of separate planes that divide an n-dimensional feature vector space[*3] into a number of categories.

We can also expect the system to become smarter on its own by continuing to incorporate big data from outside, with the effect that even unknown words will be classified appropriately. The categories for the primary classification are not limited to businesses: it is possible to broaden the application to countries, local governments, municipalities, celebrities, products, and etc.

2.3 Architecture and functions

The design concept of Jubatus is outlined in **Fig. 1**. As stated above, Jubatus is composed of a cluster of machine learning engines and a high-speed framework that supports them.

In contrast to previous machine learning engine units, which usually handled small- to medium-scale data and required batch processing and individual development, Jubatus has a huge variety of engines installed in a high-speed framework and a development mechanism with common specifications for high-speed big data processing with errors within a permitted range tolerated [6]. June 2012 saw the release of version 0.3 of Jubatus and we are planning to augment the lineup of machine learning functions and strengthen the support framework. The types of machine learning that we currently support so far are outlined in **Table 1**.

Jubatus will be useful for applications that require speedy judgment. It is expected to be able to discover and analyze original relationships among the data volume from different domains.

2.4 Distributed processing architecture

The distributed processing architecture is shown in detail in **Fig. 2**. The stream of big data flows from left to right. Clients are configured of a number of user processes and proxy processes.

The proxy processes relay clients' requests to the server processes, enabling the servers be transparent to the user processes. User processes are implemented by using the Jubatus client application programming interface (API); they are written in a scripting language or in a general programming language.

We appreciate the outside contributions from the open source community, so we enabled the use of a large variety of language bindings [7], such as Python, Ruby, and Java. Communications between proxy processes and server processes are based on MessagePack [8] remote procedure calls (RPCs). Non-block input/output enables more efficient communications and synchronization control. The server processes perform the training, prediction processing, and learning model synchronization, which has linearly increasing performance with the number of servers. In addition, Zookeeper [9] processes manage the cooperation between proxy and server processes, the balancing between distributed servers, the monitoring of server state (alive/dead), and the selection of new leaders.

*3 n-dimensional feature vector space: n is the number of words that characterize the Japanese language. In this case, it is approximately 2000.

Table 1. Examples of functionalities for machine learning.

Supported machine learning engines	Functionalities for machine learning
Classification	Learning request Classification request
Regression	Learning request Analogy request
Recommendation	Update or deletion of row data (user ID/items) Search for similar data Figure out recommendation items
Statistics	Update Calculation of: sum, deviation, maximum & minimum, entropy, moment
Graph mining	Node creation, deletion, update, reference Edge creation, deletion, update, reference Centrality measures Shortest path calculations between two nodes

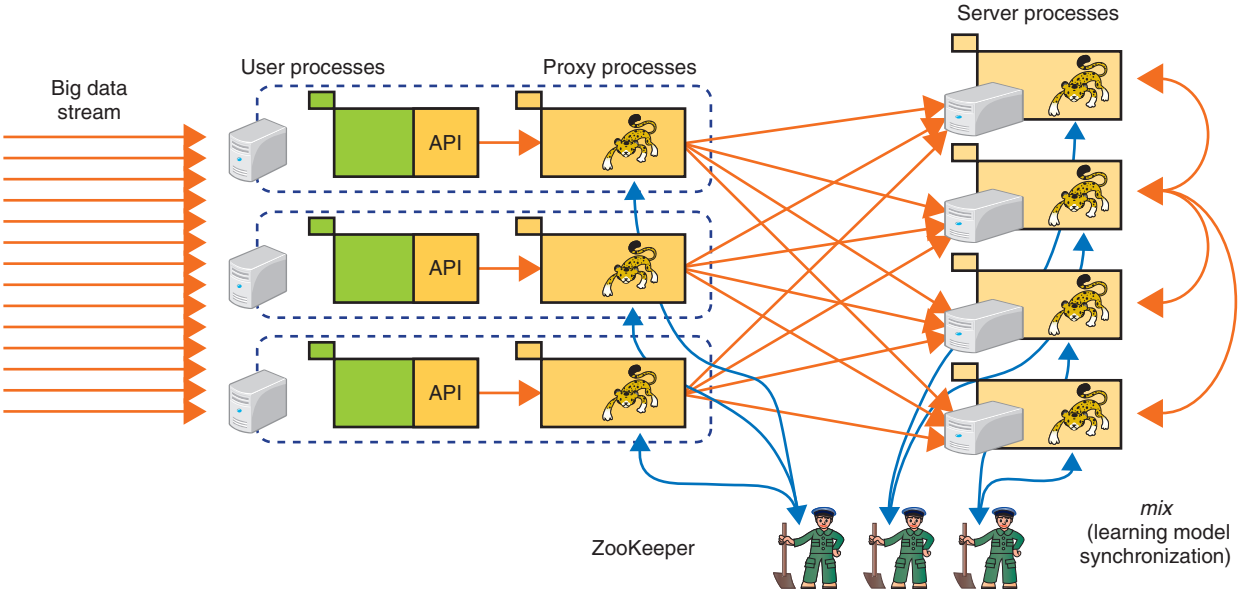


Fig. 2. Distributed processing architecture.

Table 2. Characteristics of Jubatus distributed processing.

	Batch processing	Jubatus
Processing capability	Stockpile type: Lumped process with data accumulation	Stream type: on-demand processing without data accumulation
Running time	Scheduled type: Start and end timing are defined.	24 hours per day, 7 days per week, i.e., non-stop

The characteristics of Jubatus distributed processing are compared with those of batch processing in **Table 2**. For example, when there are 1000 distributed parallel processing servers, batch processing (e.g., using Map-Reduce) means that the 1000 servers all execute Map simultaneously, and then all of them

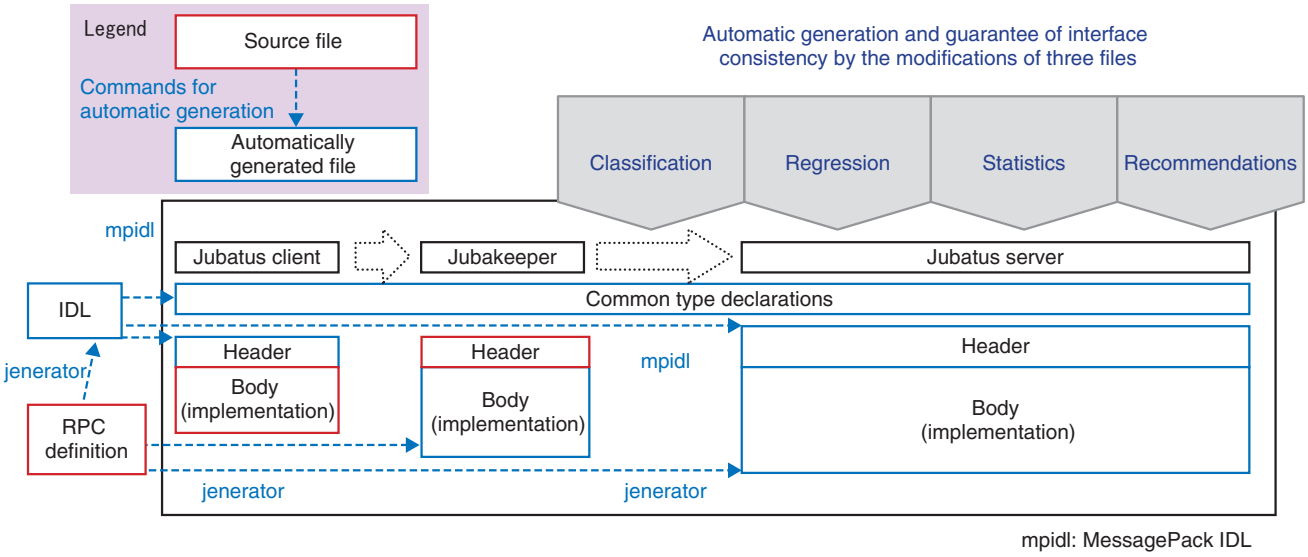


Fig. 3. Automatic creation of template interface by IDL.

together execute Reduce to summarize the results. By contrast, with Jubatus, each server repeats the learning and analysis autonomously.

In parallel processing among distributed servers, one technique for satisfying both profound analysis and scalability is *mix* processing [1], [6], [10]. This processing can be regarded as resembling a group study meeting for self-teaching and checking answers (or called synchronization) with others. If all the 1000 servers synchronize their answers after every single learning or analysis task, then overall there will be a huge drop in processing speed due to the waiting time, but a moderate frequency of synchronization may enable the data learning and analysis without any decrease in overall performance.

Let us introduce *mix* processing and the unified update-mix-analyze (UMA) interface.

- Update: Asynchronous execution of update-related queries and additions or changes to big data
- Mix: Loose sharing of intermediate analysis results within the component servers
- Analyze: Abstraction interface that suggests the execution of reference queries and analysis requests from clients.

The aim is to implement a common interface that is independent of the analysis logic. In the future, we will continue upgrading the machine learning engine, but designing and maintaining an easy-to-understand framework with consistent interfaces are also important tasks.

3. Improvements in development efficiency

To improve the agility of development and ease the burden on developers, a full range of designs and tools is being provided in Jubatus.

(1) Feature vector conversion

Whatever the data, it is necessary to capture its feature vectors as the input format for machine learning. Conversely, if feature vectorization is successful, the machine learning algorithms only need to be processed appropriately. We have provided a tool that defines this important feature vectorization by means of an outgoing settings file, without hard-coding it as program logic.

(2) IDL and jenerator

In Jubatus version 0.01, any addition to the analysis logic led to the need for manual editing and verification of seven files every time. Therefore, we devised a mechanism for automatically maintaining interface consistency and localizing the interface alteration work. We define the interface by using an interface description language (IDL) file, and template skeletons of six files are generated automatically by a tool called a jenerator (**Fig. 3**). As a result, approximately 3000 lines of code can be generated automatically by using a 100-line definition file, and unification of the source code maintenance work is also improved by IDL. The meaning and usage of the API can be simply

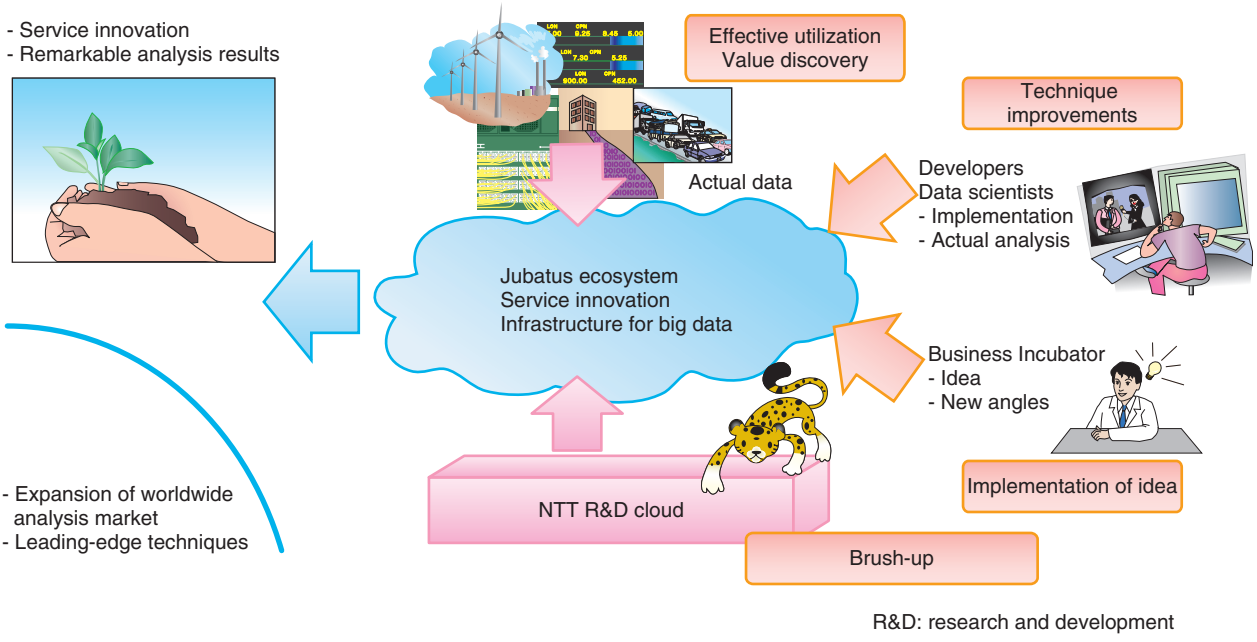


Fig. 4. Jubatus ecosystem.

understood just by reading the IDL file, which greatly improves the development efficiency.

4. Performance evaluation

We verified the performance beforehand while expanding the analysis logic. This evaluated the scalability of Jubatus.

(1) Classification: We classified Twitter tweets on a global scale (8000 per second) with two or three commodity servers. An accuracy rate of 90% in batch processing was achieved with data obtained within 10 s. The quantity of training data was not necessarily that large.

(2) Recommendations: The update frequency of online shopping transactions can be over 100,000 times per second per server and it is possible to produce recommendations for 30 million users within a response time of 0.1 s (approximately ten times as fast as Mahout). We also verified that the throughput (number of registered data items per second) scales up linearly with increasing number of servers.

(3) Graph mining
- Addition of 100 million edges: by ten commodity servers in parallel for 5 minutes
- Scalability in edge addition through an increase in the number of servers: (1, 2, 4, 8) servers ==> (3, 6, 13, 25) at 10,000 edges per second (throughput).
- Analysis latency: {update, mix, centrality}: 0.1 to 0.3 μs per node; {shortest-path}: from 0.1 μs to several tens of microseconds per node (paths of approximately 1000 hops). This is up to a thousand times as fast as one well-known graph database product.
- Data size: Retention of over 16 million edges per server (8 GB of memory) ==> Retention of 100 million edges with 50 GB.

5. Open source status and future schedule

As of July 2012, Jubatus is in its third version and an OSS release is available. It provides an integrated, easy-to-use infrastructure aimed at specialists, programmers, and developers. It also provides an infrastructure designed to let data scientists slot in analysis algorithms that they want to try out and rapidly scale up. The open source community will continue to work actively on this.

A Jubatus analysis idea contest (Jubatus Challenge Japan 2012) was held from July to September, 2012 [11]. The aim of this contest was to seek ideas and business scenarios that reveal the potential of Jubatus to the maximum by active application of open innovation. An important point is the design of a Jubatus

ecosystem. Our main challenge is to act like a catalyst in a chemical reaction amidst organic links formed by main components without any deficiencies in the Jubatus infrastructure technology, machine learning algorithms, big data, or data scientists, in other words, to design a realtime machine learning and analysis ecosystem such as that shown in **Fig. 4**.

6. Future expansion

To date, the big data analysis business has centered on making industrial use of the discovery of previously unnoticed relationships by data mining, and batch processing analysis has centered on taking a large volume of data that had been stockpiled and analyzing it from many angles by trial and error to obtain original discoveries. Henceforth, it will be important to perform research and development from the following viewpoints.

(1) Cross domain relationship analysis: Instead of analyzing big data confined to a specific area, discover original business possibilities and synergistic effects from relationships within big data spanning different fields or specialist areas.

(2) Realtime analysis: Support speedy operations and create competitive advantages by making rapid decisions based on stream-type data analysis.

(3) Profound, realtime, big data analysis: Develop highly value-added techniques that meet the need for automatic judgment (based on machine learning) where decision logic has not been previously defined, but can reflect any change in status rapidly on the fly for automatic judgment on the microsecond time-scale to provide rigorous analysis results that cannot wait until tomorrow or the day after and where actions taken through human judgment (decision-making route) would be too slow.

In the future, we will continue to promote open source development and open innovation and the expansion of real-life cases.

References

[1] S. Oda, K. Uenishi, and S. Kinoshita, "Jubatus: Scalable Distributed Processing Framework for Realtime Analysis of Big Data," NTT Technical Review, Vol. 10, No. 6, 2012. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201206ra2.html

[2] Jubatus website. http://jubat.us/

[3] Jubatus on Twitter. https://twitter.com/jubatusofficial

[4] Jubatus on github. https://github.com/jubatus/jubatus

[5] "The PageRank Citation Ranking: Bringing Order to the Web," http://infolab.stanford.edu/~backrub/pageranksub.ps

[6] NTT Press release, "Leading Development of Scalable Distributed Computing Framework for Real-Time Analysis of Big Data," http://www.ntt.co.jp/news2011/1110e/111026a.html

[7] Language binding. http://en.wikipedia.org/wiki/Language_binding

[8] MessagePack. https://github.com/msgpack/msgpack-rpc

[9] Zookeeper. http://zookeeper.apache.org/

[10] S. Oda, S. Nakayama, K. Uenishi, and S. Kinoshita, "Jubatus: Distributed Processing Technique Enabling Realtime Processing of Big Data," IEICE Tech. Rep., Vol. 111, No. 409, IN2011-126, pp. 35–40, Jan. 2012.

[11] Jubatus Challenge. http://www.facebook.com/JubatusChallenge2012

Keitaro Horikawa

Senior Research Engineer, Supervisor, Distributed Computing Technology Project, NTT Software Innovation Center.

He received the B.S. and M.S. degrees in information engineering from Niigata University in 1988 and 1990, respectively. Since joining NTT in 1990, he has been engaged in R&D of software design and architecture, concurrent distributed object computing, metaprogramming and computational reflection, single-sign-on for web services, lightweight programming languages, mobile cloud computing and big data computing, CSCW, CSCL, and game theoretical modeling for decision making and human factors in R&D. He received the Best Paper Award for Young Researchers from the Information Processing Society of Japan (IPSJ) National Convention in 1995. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center. He is currently managing distributed computing projects including Jubatus. He has PMP certification by PMI and TOGAF 9 certification by the Open Group.

Yuzuru Kitayama

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

He received the B.E. and M.S. degrees in engineering from Kobe University, Hyogo, in 1996 and 1998, respectively. Since joining NTT in 1998, he has been engaged in the development and management of large-scale web search systems and R&D of distributed processing systems. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center.

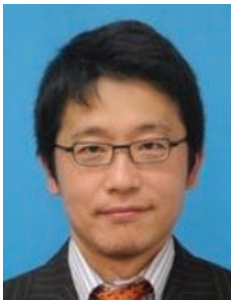
Satoshi Oda
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.E. and M.E. degrees in engineering from Keio University, Kanagawa, in 2003 and 2005, respectively. Since joining NTT Information Sharing Platform Laboratories in 2005, he has been engaged in R&D of information security, fast implementation of cryptography, and security protocols. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center. He received the 2007 Outstanding Presentation Award from the Japan Society for Industrial and Applied Mathematics (JSIAM) and the 2009 Life Intelligence and Office Information System (LOIS) Research Award. He is a member of JSIAM.

Hiroki Kumazaki
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.E. and M.E. degrees in engineering from Nagoya Institute of Technology, Aichi, in 2010 and 2012, respectively. Since joining NTT Software Innovation Center in 2012, he has been researching distributed computing systems.

Jungkyu Han
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.S. and M.S degrees in computer science and engineering from Seoul National University, Korea, in 2005 and 2007, respectively. Since joining NTT in 2007, he has been engaged in R&D of stream data accumulation and distributed processing systems. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center. His current interests include big data analysis.

Hiroyuki Makino
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.E. and M.S. degrees in engineering from Doshisha University, Kyoto, in 2007 and 2009, respectively. Since joining NTT Information Sharing Platform Laboratories in 2009, he has been engaged in R&D of identity federation systems and distributed processing systems. 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).

Masakuni Ishii
Researcher, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.S. degree in mathematics and computer science from the University of Arizona, USA, in 2007 and the M.S. degree in computer engineering from Keio University, Kanagawa, in 2009. Since joining NTT in 2010, he has been engaged in R&D of stream data accumulation and distributed processing systems. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center. He is a member of IPSJ.



Koji Aoya
Senior Research Engineer, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the M.S. degree in mathematics from Osaka University in 2001. After joining NTT Information Sharing Platform Laboratories in 2001, he was engaged in R&D of information security, fast implementation of cryptography (especially ECDSA), and some applied protocols such as electronic money. In 2007, he was transferred to NTT DATA Corporation, where he was engaged in marketing and developed some solutions for retail and airline industries in the related departments. Since moving to NTT Software Innovation Center in 2012, he has been incubating and developing the market for Jubatus.

Min Luo
Scientist, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.S. and M.S. degrees from Wuhan University, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer science from Tokyo Institute of Technology in 2011. After that, he was a full-time Associate Researcher at Tokyo Institute of Technology. Since joining NTT Software Innovation Center in 2012, he has been engaged in research on realtime big data analysis. He received the Best Paper Award from the International Conference on Web-Age Information Management (WAIM) in 2010.

Shohei Uchikawa
Senior Research Engineer, Supervisor, Group Leader, Distributed Computing Technology Project, NTT Software Innovation Center.
He received the B.E. degree in information engineering from Tokyo Institute of Technology in 1988. After joining NTT Information Processing Systems Laboratories in 1988, he was engaged in developing operating system software for the Denden Information Processing System (DIPS) and designing a high-availability information system that uses DIPS. Following the gradual termination of the DIPS project in the 1990s, he took on the job of integrating several high-availability client/server type information systems as a systems engineer. In 2008, he moved to NTT Communications as a general manager in the Platform Service Department and he contributed to the company's application service provider business. He returned to NTT in 2011. As a result of organizational changes in April 2012, he is now in NTT Software Innovation Center.