Large-scale Distributed Data Processing Platform for Analysis of Big Data

Mitsukazu Washisaka, Eiji Nakamura, Takeshi Takakura, Satoru Yoshida, and Seiji Tomita†

Abstract

Cloud technology can scale horizontally (scale out) through the addition of servers to handle more data and enables the analysis of large volumes of data (known as big data). This has led to innovation in the form of new services that create added value such as providing recommendations on the basis of attributes extracted from the big data. In this article, we describe technology and services implemented with large-scale data processing, the development of a large-scale distributed data processing platform, and technology fostered by the program design and system level testing/verification.

1. Introduction

One application area for cloud computing is large-scale data processing. The analysis of voluminous data within a realistic time requires abundant computer resources, including central processing units, disk space for data storage, and network bandwidth. A lot of researchers have been interested in scale-out (horizontal scaling) technology and have started research and development (R&D) activities as a way to deal with large-volume data analysis by adding blade/rack servers to increase processing speed and expand data capacity. NTT Information Sharing Platform Laboratories has also been developing a large-scale distributed data processing platform called CBoC Type 2 (CBoC: Common IT Bases over Cloud Computing; IT: information technology) [1].

In this article, we first describe services that use large-scale data processing and then introduce technology for achieving the high degree of availability required of a large-scale distributed data processing platform as well as testing technology for evaluating applicability.

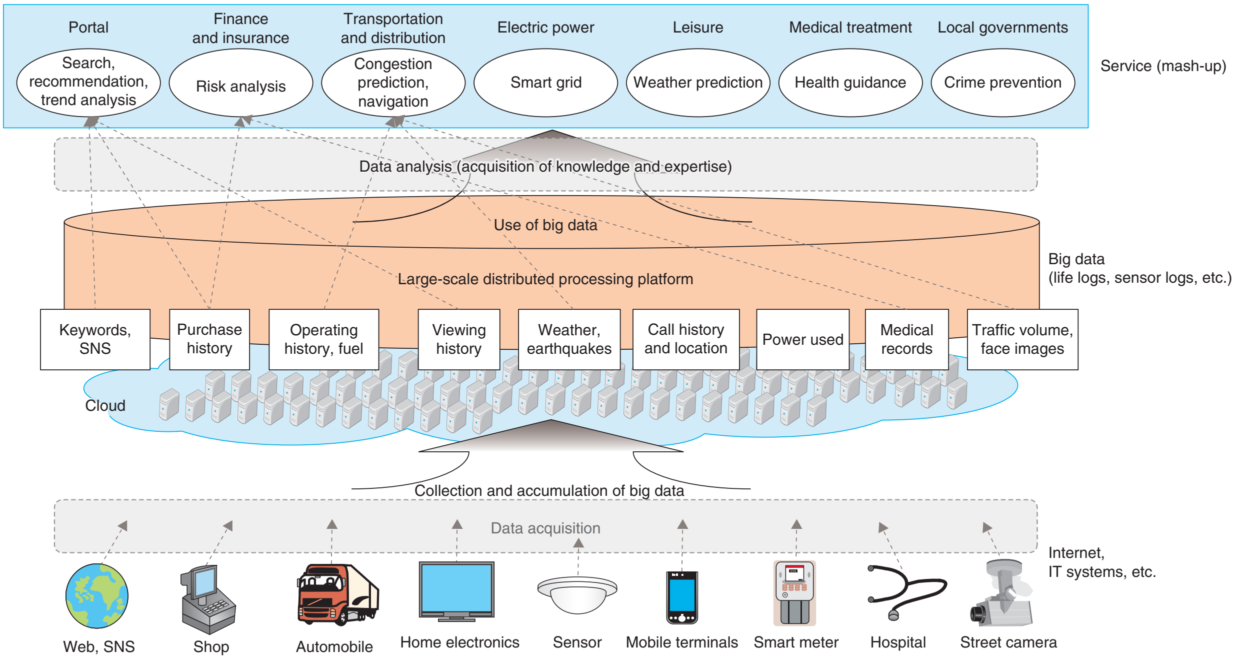
2. Advanced services through large-scale data processing

It is difficult to manage the data generated in the web, IT systems, etc. (e.g., transaction logs, sensor logs, and life logs) and other data that continues to increase explosively in volume. Analysis of such voluminous data (known as big data) in a conventional manner becomes exponentially costly even when the data is collected by the system, so the data has been either stored wastefully or discarded. The advent of scale-out technology, however, has reduced the cost of constructing systems for processing large-scale data, and new advanced services such as personalization based on analytical results are now possible.

Large-scale data processing enables the use of diverse types of big data in a cloud environment in order to create mash-up[*1] services, as shown in **Fig. 1.** A large-scale distributed data processing platform collects and stores the big data produced by IT systems or the Internet. By analyzing such large volumes of data, one can acquire new knowledge and expertise

† NTT Information Sharing Platform Laboratories
Musashino-shi, 180-8585 Japan

[*1] Mash-up: A new service constructed by combining multiple application programming interfaces.



SNS: social networking service

Fig. 1. Large-scale distributed data processing platform and application to services.

and create new mash-up services. A large-scale distributed data processing platform is expected to serve as a platform for creating knowledge on which to base advanced services for customers.

3. Development of CBoC Type 2

The conventional approach to the management of various types of data is to use a relational database management system (RDBMS)[*2], but for big data, a technology known as NoSQL (not only SQL (structured query language))[*3] is a cost-effective approach. NoSQL technology implements scale-out by relaxing the guarantee of data consistency that is essential for transaction processing in a RDBMS. Thus, while an RDBMS is suited to the analysis of structured data, the NoSQL approach, which is based on BASE (basically available, soft states, eventual consistency), is suited to the processing of big data that is less structured, such as natural language data.

Typical examples in this field include the service platforms provided by Google and Amazon [2], [3] and the open-source software Hadoop [4]. Hadoop users are increasing in number, and systems on the scale of thousands of servers have been reported to be in operation. However, some problems regarding introduction to mainstream systems that require continuous operation remain; one example is the inability to switch servers online when a management server fails. We therefore took up the challenge of developing CBoC Type 2 to improve the reliability, operability, and maintainability of a large-scale distributed data processing platform.

4. Improving fault tolerance in large-scale distributed data processing platforms

CBoC Type 2 comprises three distributed processing subsystems: a distributed file system for storing big data on many blade/rack servers in a distributed manner, a distributed table subsystem for managing big data as structured data, and a distributed lock subsystem, which provides a basic function that allows a high degree of settlement in these distributed systems (**Fig. 2**).

In the development of CBoC Type 2, particular effort was made to improve fault tolerance, which is

[*2] RDBMS: A database management system that features data representation in the form of two-dimensional tables.

[*3] NoSQL: Processing requests submitted to an RDBMS are often written in SQL, but NoSQL refers to a language designed as *not being an RDBMS*.

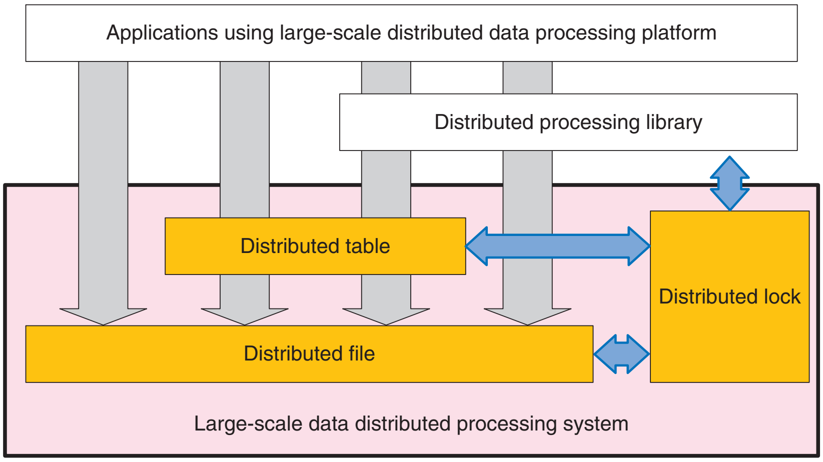

Applications using large-scale distributed data processing platform
Distributed processing library
Distributed table
Distributed lock
Distributed file
Large-scale data distributed processing system

Fig. 2. CBoC Type 2 software structure.

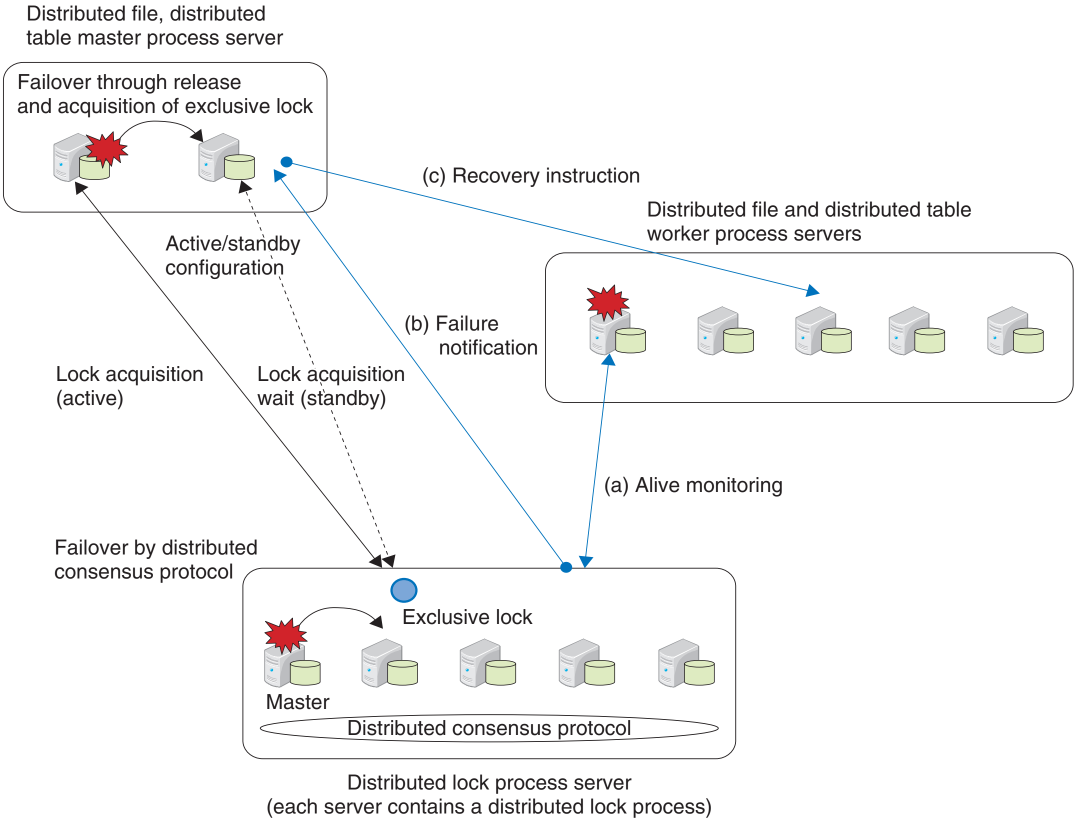

Distributed file, distributed table master process server
Failover through release and acquisition of exclusive lock
(c) Recovery instruction
Distributed file and distributed table worker process servers
Active/standby configuration
(b) Failure notification
Lock acquisition (active)
Lock acquisition wait (standby)
(a) Alive monitoring
Failover by distributed consensus protocol
Exclusive lock
Master
Distributed consensus protocol
Distributed lock process server (each server contains a distributed lock process)

Fig. 3. CBoC Type 2 failure recovery processing.

important for managing scale out. In systems that use many blade/rack servers, the probability of overall system failure is high, even if the failure rate of individual servers is low. Thus, ensuring hardware failure tolerance is a major problem. In CBoC Type 2, the distributed lock function provides the basis for fault tolerance in the distributed file and distributed table subsystems (**Fig. 3**).

4.1 Fault tolerance of distributed locks

A distributed lock works to set one of the five processess on different servers to be the master process. The five server processes communicate via a distributed consensus protocol to form a consensus among backup processes on servers [5]. If the master process server fails, a new master is selected from the remaining backup process servers by a distributed consensus protocol, and the new master takes over the previous master's processing (failover). By doing so, it maintains the distributed lock process with respect to the whole system [1].

4.2 Fault tolerance of distributed files and distributed tables

Distributed files and distributed tables consist of many worker processes that process requests from applications and two master processes that control the worker processes. The active master and standby master processes are distinguished by using the exclusive lock[*4] function of the distributed lock. The master that succeeds in acquiring an exclusive lock becomes the active master and the master that failed to acquire the lock becomes the standby master. A distributed lock monitors the life and death of each master process; if the active master fails, the exclusive lock is released. The standby master can then acquire the exclusive lock and become the active master (failover).

The distributed lock also monitors the life and death of worker processes ((a) in Fig. 3). If a worker process fails, the failure is reported to the active master (b). The active master issues an instruction to another running worker process to recover the data (c) that was being managed by the failed worker process and the running worker process takes up the processing that was being performed by the failed worker process. To prevent data loss due to distributed file failure, the data is made redundant and the same data is managed by multiple worker processes.

4.3 Fault tolerance considering the network

The network configuration of the servers also has a strong effect on fault tolerance. CBoC Type 2 uses a tree network configuration in which multiple edge switches are subordinate to a core switch and each edge switch connects to multiple servers. The master processes of the distributed file and distributed table system are positioned under different edge switches and the redundant data of distributed files is managed by worker processes that are under different edge switches.

In that way, CBoC Type 2 implements fault tolerance such that the whole system does not go down when a single unit of hardware (such as a server or switch) fails, even if an edge switch failure results in multiple servers being removed from the system at the same time.

5. System testing in large-scale distributed systems

Large-scale distributed systems designed for the analysis of big data may comprise from several tens to several hundreds of servers, or even thousands in some cases, depending on the scale of the data to be processed and the nature of the processing. The number of variations of state transitions in such distributed systems is huge, so system testing is correspondingly more important. Therefore, system testing in CBoC Type 2 is designed to allow the construction of a testing environment that involves thousands of servers and implementation of testing that assumes actual service use cases.

The difficulties faced by testing in the construction of a large-scale distributed data processing platform include 1) automation of testing environment construction, 2) faster registration of big data, and 3) more efficient confirmation of test results. CBoC Type 2 deals with those problems in the following ways.

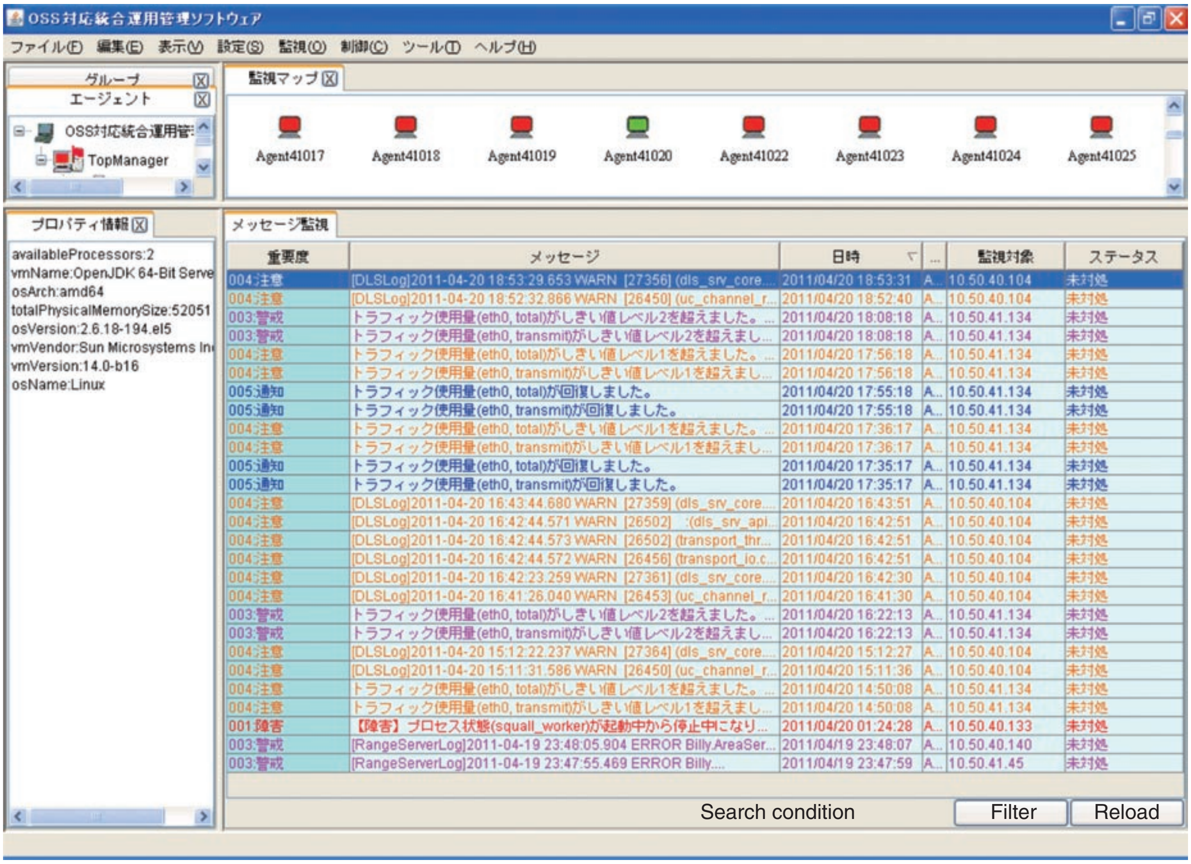
5.1 Automation of testing environment construction

The difficulty of constructing testing environments can probably be imagined by simply considering the work involved in installing software on several hundred servers. System management automation tools can be used effectively in the construction of such an environment. Environment construction and maintenance can be made more efficient by using system management automation tools such as Puppet [6] for unified management of installed operating systems, middleware, application programs and various settings, as well as for the management of the installation process.

5.2 Faster entry of big data

To check for stable system operation, we developed a data entry tool that can construct various data

[*4] Exclusive lock: A mechanism that limits the number of processes that can modify data to one to avoid problems caused by modification of the same data by multiple processes.



availableProcessors:2
vmName:OpenJDK 64-Bit Serve
osArch:amd64
totalPhysicalMemorySize:52051
osVersion:2.6.18-194.el5
vmVendor:Sun Microsystems In
vmVersion:14.0-b16
osName:Linux

重要度	メッセージ	日時	...	監視対象	ステータス
004:注意	[DLSLog]2011-04-20 18:53:29.653 WARN [27356] (dls_srv_core....	2011/04/20 18:53:31	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 18:52:32.866 WARN [26450] (uc_channel_r...	2011/04/20 18:52:40	A...	10.50.40.104	未対処
003:警戒	トラフィック使用量(eth0, total)がしきい値レベル2を超えました。...	2011/04/20 18:08:18	A...	10.50.41.134	未対処
003:警戒	トラフィック使用量(eth0, transmit)がしきい値レベル2を超えまし...	2011/04/20 18:08:18	A...	10.50.41.134	未対処
004:注意	トラフィック使用量(eth0, total)がしきい値レベル1を超えました。...	2011/04/20 17:56:18	A...	10.50.41.134	未対処
004:注意	トラフィック使用量(eth0, transmit)がしきい値レベル1を超えまし...	2011/04/20 17:56:18	A...	10.50.41.134	未対処
005:通知	トラフィック使用量(eth0, total)が回復しました。	2011/04/20 17:55:18	A...	10.50.41.134	未対処
005:通知	トラフィック使用量(eth0, transmit)が回復しました。	2011/04/20 17:55:18	A...	10.50.41.134	未対処
004:注意	トラフィック使用量(eth0, total)がしきい値レベル1を超えました。...	2011/04/20 17:36:17	A...	10.50.41.134	未対処
004:注意	トラフィック使用量(eth0, transmit)がしきい値レベル1を超えまし...	2011/04/20 17:36:17	A...	10.50.41.134	未対処
005:通知	トラフィック使用量(eth0, total)が回復しました。	2011/04/20 17:35:17	A...	10.50.41.134	未対処
005:通知	トラフィック使用量(eth0, transmit)が回復しました。	2011/04/20 17:35:17	A...	10.50.41.134	未対処
004:注意	[DLSLog]2011-04-20 16:43:44.680 WARN [27359] (dls_srv_core....	2011/04/20 16:43:51	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 16:42:44.571 WARN [26502] :(dls_srv_api...	2011/04/20 16:42:51	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 16:42:44.573 WARN [26502] (transport_thr...	2011/04/20 16:42:51	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 16:42:44.572 WARN [26456] (transport_io.c...	2011/04/20 16:42:51	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 16:42:23.259 WARN [27361] (dls_srv_core....	2011/04/20 16:42:30	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 16:41:26.040 WARN [26453] (uc_channel_r...	2011/04/20 16:41:30	A...	10.50.40.104	未対処
003:警戒	トラフィック使用量(eth0, total)がしきい値レベル2を超えました。...	2011/04/20 16:22:13	A...	10.50.41.134	未対処
003:警戒	トラフィック使用量(eth0, transmit)がしきい値レベル2を超えまし...	2011/04/20 16:22:13	A...	10.50.41.134	未対処
004:注意	[DLSLog]2011-04-20 15:12:22.237 WARN [27364] (dls_srv_core....	2011/04/20 15:12:27	A...	10.50.40.104	未対処
004:注意	[DLSLog]2011-04-20 15:11:31.586 WARN [26450] (uc_channel_r...	2011/04/20 15:11:36	A...	10.50.40.104	未対処
004:注意	トラフィック使用量(eth0, total)がしきい値レベル1を超えました。...	2011/04/20 14:50:08	A...	10.50.41.134	未対処
004:注意	トラフィック使用量(eth0, transmit)がしきい値レベル1を超えまし...	2011/04/20 14:50:08	A...	10.50.41.134	未対処
001:障害	【障害】プロセス状態(squall_worker)が起動中から停止中になり...	2011/04/20 01:24:28	A...	10.50.40.133	未対処
003:警戒	[RangeServerLog]2011-04-19 23:48:05.904 ERROR Billy.AreaSer...	2011/04/19 23:48:07	A...	10.50.40.140	未対処
003:警戒	[RangeServerLog]2011-04-19 23:47:55.469 ERROR Billy....	2011/04/19 23:47:59	A...	10.50.41.45	未対処

Fig. 4. Screenshot from the integrated operation and management software (Crane).

storage states in a short time; this function is essential for testing when a large amount of data has been entered. The tool incorporates functions for restarting from a state midway through data entry and for automatically adding previously entered data, etc.; these functions greatly reduce the preparation time required for testing.

5.3 More efficient validation of test results

The items involved in the validation of test results include the input and output results as seen by the application, the data entry state, and whether or not an error occurred. For large-scale testing environments in particular, realtime observation and visualization of error occurrence and server process states made possible by the monitoring of system operation and error log and other such records is important. For this purpose, NTT's Crane technology [7] or other similar integrated operation management software is used for efficient confirmation of test results. The operation management screen of Crane is shown in **Fig. 4**.

By proceeding with testing while using related technology and tools in addition to the three subsystems for distributed files, distributed tables, and distributed locks in this way, we are building up the expertise in system operation needed for large-scale environments in order to bring CBoC Type 2 to a highly practical level of development.

6. Future development

In developing CBoC Type 2, we encountered many difficulties in running programs in a large-scale environment and we learned much during the development and testing phases. By gaining use experience for various kinds of assumed services through the application of CBoC Type 2 to NTT R&D Cloud, the cloud environment for NTT's R&D centers, we will identify the common functions and performance requirements for the platform and increase applicability to specific needs to develop CBoC Type 2 into a large-scale distributed data processing platform that has high reliability, operability, and maintainability.

References

[1] T. Takakura, K. Sora, Y. Amagai, M. Washisaka, and S. Tomita, "Implementing Large-scale Distributed Processing Systems with CBoC," NTT Technical Journal, Vol. 21, No. 9, pp. 80–83, 2009 (in Japanese).

[2] Google App Engine. http://code.google.com/intl/en/appengine/docs/whatisgoogleappengine.html

[3] Amazon web services. http://aws.amazon.com/.

[4] Hadoop. http://hadoop.apache.org/

[5] L. Lamport, "Paxos Made Simple," ACM SIGACT News, Vol. 32, No. 4, pp. 18–25, 2001.

[6] Puppet. http://www.puppetlabs.com/

[7] NTT Open Source Software Center, Crane (in Japanese). https://www.oss.ecl.ntt.co.jp/ossc/oss/r_crane.html

Mitsukazu Washisaka

Senior Research Engineer, Supervisor, Distributed Data Processing Platform SE Project, NTT Information Sharing Platform Laboratories.

He received the B.S. and M.S. degrees in information and computer science engineering from Osaka University in 1985 and 1987, respectively. He joined NTT Basic Research Laboratories in 1987. He has been engaged in R&D of wide-area IP networks and their applications.

Eiji Nakamura

Senior Research Engineer, Supervisor, Distributed Data Processing Platform SE Project, NTT Information Sharing Platform Laboratories.

He received the B.E. and M.E. degrees in nuclear engineering from Hokkaido University in 1986 and 1988, respectively. He joined NTT in 1988. He has been engaged in R&D of facsimile communication systems and operation systems for home gateway devices. He is currently studying big data processing and management systems.

Takeshi Takakura

Senior Research Engineer, Supervisor, Distributed Data Processing Platform SE Project, NTT Information Sharing Platform Laboratories.

He received the B.E. and M.E. degrees in material physics engineering from Osaka University in 1990 and 1992, respectively. He joined NTT in 1992 and engaged in R&D in the Network Information Systems Laboratories, where he studied multimedia database systems and information processing systems. He received the 1997 Best Paper Award for a Young Researcher of IPSJ (Information Processing Society of Japan) National Convention. He is currently studying big data processing and management systems.

Satoru Yoshida

Senior Research Engineer, Supervisor, Distributed Data Processing Platform SE Project, NTT Information Sharing Platform Laboratories.

He received the B.S. and M.S. degrees in condensed matter physics engineering from Tokyo Institute of Technology in 1987 and 1989, respectively. He joined NTT Applied Electronics Laboratories in 1989. He is currently engaged in R&D of cloud computing systems. He is a member of the Institute of Electronics, Information and Communication Engineers.

Seiji Tomita

Senior Research Engineer, Supervisor, Distributed Data Processing Platform SE Project, NTT Information Sharing Platform Laboratories.

He received the B.S. and M.S. degrees in electronics from Kyushu University, Fukuoka, in 1983 and 1985, respectively. He joined the Yokosuka Electrical Communications Laboratories of Nippon Telegraph and Telephone Public Corporation (now NTT) in 1985. He has been engaged in R&D of system software in computer systems such as operating systems, communication software, transaction monitors, and database management systems. His current interest is big data processing and management systems.