Large-scale Distributed Data Processing | Big Data | Service Platform

Technology Reports

Mobile Spatial Statistics Supporting Development of Society and Industry
—Population Estimation Using Mobile Network Statistical Data and its Applications —

Large-Scale Data Processing Infrastructure for Mobile Spatial Statistics

*NTT DOCOMO is conducting research on a data processing infrastructure that analyzes large-scale data to develop and create new services such as MSS. In this article, we describe the design, configuration and operation of our Hadoop™ *1 system which provides infrastructure for large scale data processing, as well as program development for this system. The system is composed of over 1,000 commodity IA servers and uses open source software for monitoring and operation. We have not only developed large-scale data processing programs using MapReduce but also optimized execution of the programs.*

Research Laboratories
So Ishida
Manhee Jo
Kenji Ishii
Gorou Kunitou
Makoto Nakayama
Ryohei Suzuki
Daisuke Ochi
Norihiro Kawasaki
Masafumi Oomachi
DOCOMO Technology, Inc., Multimedia Division
Ken Koumoto

1. Introduction

We are conducting research on a data processing infrastructure that analyzes large-scale data (Big Data) and developing new services [1]. One good example of these services is Mobile Spatial Statistics (MSS). This system analyzes the huge amounts of operational data from mobile networks to understand activities in society.

Hadoop[2] is drawing much attention as a platform for processing Big Data. A Hadoop cluster is constructed from commodity IA servers*2. It consists of two major functionalities: the Hadoop Distributed File System (HDFS)™*3 for storing and managing Big Data, and MapReduce to process it. Hadoop is middleware*4 for minimizing the effects of server failure over the entire system and it enables seamless substitution of a failed server with a new one. Thus, even though the number of available servers decreases due to server failure, issues such as data loss or system crash can be avoided. MapReduce is a framework that decreases processing latency by distributing Big Data over multiple servers and processing in parallel, allowing processing capacity to be increased flexibly according to the number of available servers. Thus, a Hadoop cluster composed of a large group of servers permits some server failures and reduces the operations and maintenance load.

NTT DOCOMO has built a Hadoop

*1 **Hadoop™**: A registered trademark of Apache Software Foundation.
*2 **IA server**: A server equipped with an Intel microprocessor. Its internal structure is very similar to that of an ordinary PC, and it is less expensive than servers based on other types of microprocessor.

cluster called Stevia, which is a Big Data processing infrastructure composed of over 1,000 machines. Stevia is composed entirely of commodity IA servers and network equipment and was designed using open source software for monitoring and operations. This made possible a short construction period and low cost. Due to the strengths of Hadoop, we can maintain and operate Stevia with a small number of people. We have developed Big Data processing programs using MapReduce, and increased utilization of the large-scale server cluster by optimization.

In this article, we describe the Stevia system design and construction, maintenance and operations using its open-source-software based monitoring and operations system, and execution optimization of distributed programs processing Big Data.

2. Design and Construction of Big Data Processing Infrastructure

2.1 Hadoop

Hadoop is open source software for Big Data processing comprising the HDFS and the MapReduce programming framework. HDFS manages Big Data in the form of fixed-size blocks. MapReduce programs run in parallel on the servers that hold these blocks. In a conventional database system, the storage that holds the Big Data is separate from the database engine. Thus, during processing, small amounts of data are repeatedly transferred between the storage and the database engine. The basic concept with Hadoop is that each server maintains its own blocks of data, so that unnecessary data transfer is saved.

The basic structure of the Hadoop system is shown in **Figure 1**. The system is composed of master nodes and slave nodes. The slave nodes store and process data. Two processes run on each slave node: the DataNode process maintains data blocks, and the TaskTracker process runs the MapReduce program. The master nodes manage the slave nodes, and consist mainly of a NameNode and a JobTracker. The NameNode manages the distribution of blocks to the slave nodes. The JobTracker controls execution of the MapReduce program, sending controls to start and stop the MapReduce program to the TaskTrackers. This sequence of MapReduce program executions is called a job.

Basically, copies of each block are maintained on multiple DataNodes. If a particular DataNode dies, the NameNode directs other DataNodes to create copies of the blocks that were on the dead DataNode, automatically maintaining a predefined number of copies. In this way data loss is prevented, even if multiple slave nodes fail intermittently. The JobTracker sends instructions to the TaskTrackers to run the MapReduce program and monitors its execution. If the execution on a specific

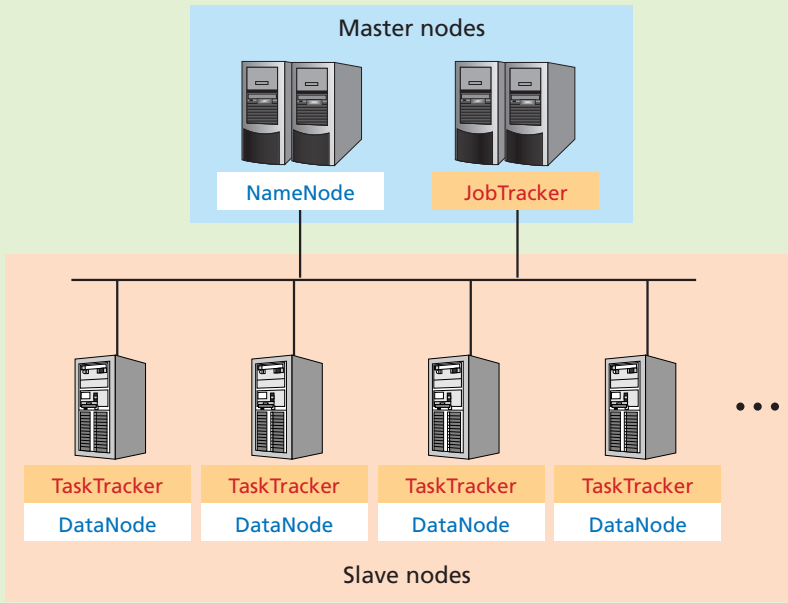

Figure 1 Hadoop configuration

*3 **HDFS™**: The distributed file system used by Hadoop. HDFS is a registered trademark of the Apache Software Foundation.
*4 **Middleware**: Software positioned between the OS and actual applications, providing common functions for diverse applications and thereby making application development more efficient.

TaskTracker fails, the command to run the MapReduce program is re-issued to another TaskTracker. The JobTracker also controls other parameters such as execution priorities when multiple jobs are submitted. These functions enable the system to continue operating even if some slave nodes fail and the number of available slave nodes decreases.

However, when a master node including the NameNode or the JobTracker fails, it cannot be replaced by another node so it is a single point of failure[*5]. Therefore, to increase reliability, these servers must be designed with redundancy.

2.2 Design and Construction of Stevia

The server configuration for Stevia is shown in **Figure 2**. The master node group, which is a single point of failure, is made redundant using a Distributed Replicated Block Device (DRBD)®[*6][3], and Heartbeat[*7][4]. The disk is completely synchronized between the two servers by the DRBD, and Heartbeat transitions the service to the standby machine automatically if the active machine fails, providing high reliability. The monitoring server and install server systems support monitoring and operations.

Stevia is a large-scale server cluster, and a network is necessary to connect the many servers together. The Stevia network configuration is shown in **Figure 3**. It consists of a working network used for data processing, and a monitoring network used for system management. To prevent failure of a single component from stopping the entire system, Layer 2 SWitching hubs (L2SW)[*8], L3SW[*9] and server ports are made redundant in the working network. Server monitoring is done using two methods: the OS and a remote control board[*10]. OS level monitoring is done through the working network and remote control board monitoring is done through the monitoring network.

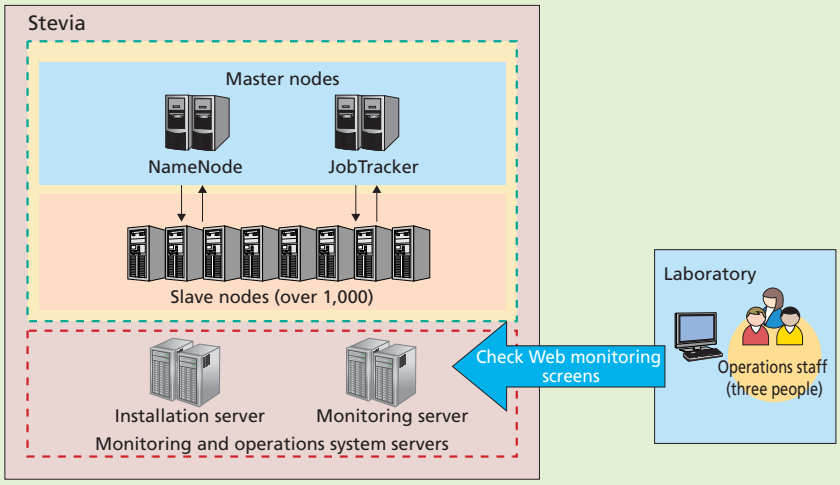

Figure 2 Stevia server configuration

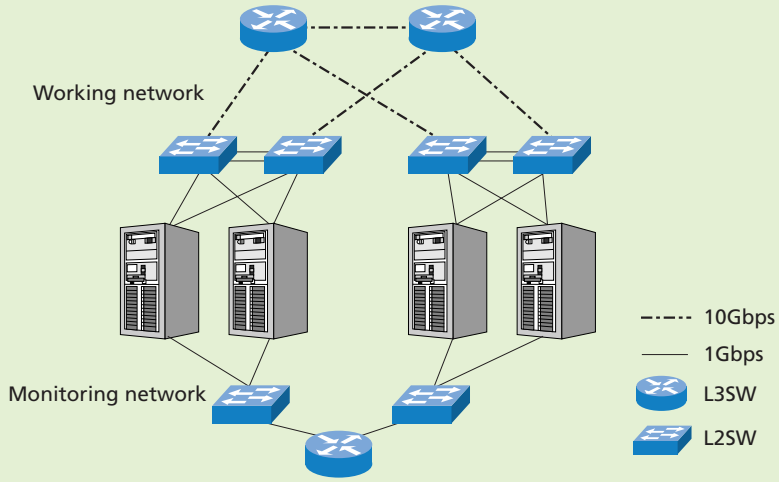

Figure 3 Stevia network configuration

*5 **Single point of failure**: A location in a system that would cause a whole system to fail if it were to fail.

*6 **DRBD**®: Middleware that performs disk partition mirroring among multiple Linux servers. DRBD is a trademark or registered trademark of LINBIT Information Technologies GmbH in Australia, USA, and other countries.

*7 **Heartbeat**: Software used for configuring high-availability clusters. It provides functionality to substitute servers a service when the original server providing the service fails.

*8 **L2SW**: A switch at the second layer of the OSI reference model, the data link layer. Usually this refers to a switching hub that transfers Ethernet packets.

*9 **L3SW**: A switch at the third layer of the OSI reference model, the network layer. Usually this refers to a router that transfers TCP/IP packets.

The operational state of the OS can be monitored through the remote control board, and the state of the remote control board can be obtained by logging in to the OS, so the monitoring network itself was not made redundant. Each server is connected to a per-rack L2SW by a 1 Gbps connection. There are 10 Gbps connections between the per-rack L2SW and the higher-level L3SW.

Stevia was built in six months, including the monitoring and operations systems, from hardware delivery to the start of operations. The server and network equipment was all off-the-shelf products, procured at low cost. When operation of Stevia began, the HDFS had a total capacity of approximately one petabyte, but was found to be not enough, so new slave-node servers were added later. The servers could be added easily, using the same procedures as are used to recover failed servers, and without stopping operation of all of the servers.

3. Monitoring and Operations of Stevia

We installed Stevia at a remote location, separate from the authors' ordinary workplace and laboratory. On the other hand, we also built another small scale server cluster for program development and testing in the laboratory. This created the need for a monitoring and operations system not dependent on the location or scale of the system being monitored. Thus, we built such a system using open source software.

The Stevia monitoring and operations system implements the following three functions.

- Server and network equipment monitoring.
- Gathering server and network equipment resource statistics (memory usage, CPU loads, etc.).
- Automatic installation of software.

These functions were implemented using the open source software shown in **Table 1**.

Nagios®*11[5] monitors the live-status of servers and switches, server logs, processes, HTTP*12, Simple Network Management Protocol (SNMP)*13 for network devices and other elements, and notifies operations staff of these states. Comprehensive monitoring of all Stevia equipment can be done through Nagios, but due to the large number of notification messages, message filtering was also necessary. It is also necessary to check the status indicators on the actual devices, because there are faults that do not result in SNMP warnings.

Ganglia*14[6] and Cacti*15[7] gather CPU load, memory usage and disk usage from server and network equipment and display them. Ganglia can also display the Java®*16 memory usage and number of connections of the NameNode, which is a vital component of the HDFS, by means of additional modules.

KickStart*17[8] provides the automatic installation function. KickStart maintains a repository*18 that is always updated to the latest state, reducing the work involved in recovering and adding slave nodes. Set up of a single Stevia slave node completed in approximately 18 minutes. Installation of DataNodes can be done on 50 servers simultaneously, which reduces the time required when all servers need to be changed at once.

Since most of the hardware faults in Stevia occur on slave nodes and do not affect operation of Hadoop, no immediate reaction is required. When a hardware fault occurs, the conditions are checked and a request for repair is sent to a hardware vendor offline. Measurements have shown that approximately 0.5 person-months of work per month is required to check faults and contact vendors. Repair work does not need to be done for every hardware fault, and doing it once or twice per month is enough.

Table 1 Use of open-source software for monitoring and operations

Type	Software
Monitoring	Nagios
Statistics gathering	Ganglia
	Cacti
Automatic installation	KickStart

*10 **Remote control board**: A control board that enables remote computer operation regardless of the state of the OS, including turning the power on and off, performing a hardware reset, displaying a console screen and performing computer operations using a keyboard and mouse.

*11 **Nagios®**: An application for monitoring UNIX computers and network services. Nagios is a servicemark, trademark and registered trademark of Nagios Enterprises.

*12 **HTTP**: A communications protocol used between Web browsers and Web servers to send and receive HyperText Markup Language (HTML) and other content.

*13 **SNMP**: A protocol for the monitoring and control of communication devices (routers, computers, terminals, etc.) on a TCP/IP network.

*14 **Ganglia**: A real-time monitoring tool to gather CPU and memory state information from multiple servers, enabling them to be monitored over the Web.

Since it was launched, Stevia has had an operations rate of 99.5%, based on the time providing functionality as a working Hadoop cluster. During this period, we have had more than 1,000 operating servers continuously.

4. Optimization of Large-scale MapReduce Programs

Big Data processing with Hadoop is implemented with programs based on the MapReduce framework. A MapReduce implementation consists of two subprograms, a Mapper and a Reducer. On each slave node, Mappers read blocks stored there, select data conforming to specific conditions as key-value pairs, and forward them to a specific Reducer based on the key (shuffle process). Mappers perform this processing in parallel. Reducers read the data forwarded from Mappers and count values for each key or perform other computations. Reducers also perform these processes in parallel.

As an example, consider a program that counts the number of mobile-network control signals issued in Tokyo during a certain date and time. Mappers would check all signals issued during the specified date and time and output only those conforming to the Tokyo condition. Then, the Reducers would count the output from the Mappers to get the result. If only one slave node was used as the Mapper, processing would require a very long time, but using many slave nodes operating in parallel the work can be completed in a shorter time. This is similar to how a large job can be completed more quickly by dividing the work among many people than by one person working alone. If Mappers classify signals using the prefecture as the key and send the data to Reducers that count the signals, 47 Reducers can be used to count the signals from the 47 prefectures in parallel. In this way a MapReduce program can process larger-scale data. Large-scale programs implementing more complex processing can be implemented by chaining multiple MapReduce programs in series.

In a program chaining multiple MapReduce programs, the progress of each MapReduce program must be synchronized over all slave nodes. As shown in **Figure 4**, no slave nodes can begin the next MapReduce program until all of the current MapReduce programs have completed, even they have completed their own previous MapReduce program. To reduce the overall execution time of a large-scale program, parallelism must be maximized and run time must be minimized for each MapReduce program, and execution time in each slave node must be balanced as much as possible. However, program execution time in each slave node can easily differ greatly depending on the amount of data to be processed and other factors. For exam-

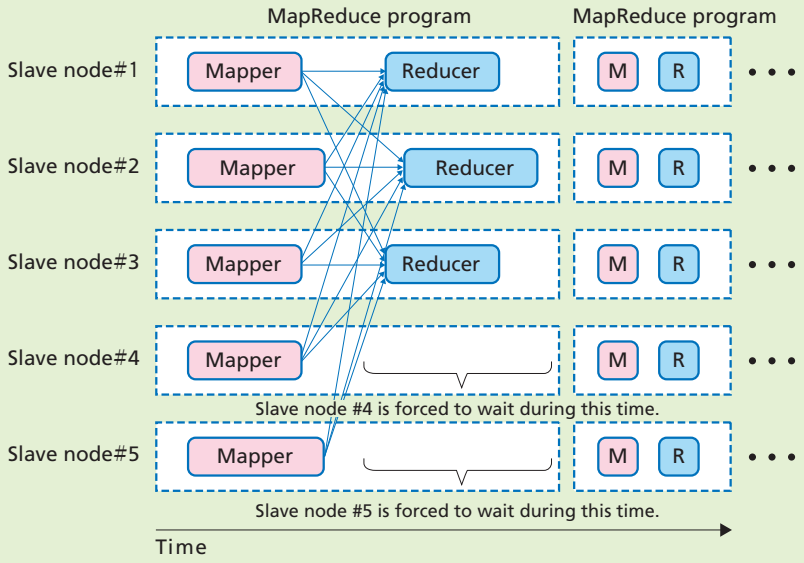

Figure 4 Example of running a large-scale MapReduce program

*15 **Cacti**: Cacti is a registered trademark of Cacti Group, Inc. It is a tool for creating graphs from SNMP data.

*16 **Java**®: An object-oriented programming language. Applications implemented in Java execute on a virtual machine, so they can operate on different platforms. Oracle and Java are registered trademarks of Oracle Corporation, its subsidiaries, and affiliates in the United States and other countries. Company and product names appearing in the text are trademarks or registered trademarks of each company.

*17 **KickStart**: A system for installing RedHat-type Linux OS's automatically.

*18 **Repository**: A system that records application and system configuration data in one place.

ple, when calculating totals using prefecture as the key in the earlier case, the number of signals issued in Tokyo would be very large, so the execution time of the Reducer calculating the total for Tokyo would also be very long. In such cases, the overall processing time could be reduced by dividing the process in two stages, for example, first with a smaller scale unit such as municipalities and then re-aggregating in prefecture units.

As described above, it is necessary to partition processing over as many slave nodes as possible and to optimize execution so that the load is not concentrated on any particular node when developing a large scale program with MapReduce. However, such optimization of large-scale MapReduce programs must be done manually, which is quite difficult. If the amount of processing on each slave node is too small, communications overhead increases or key complexity increases. Making use of this experience in developing large-scale MapReduce programs, NTT DOCOMO is now developing tools to monitor the execution state of MapReduce programs across all servers, to later visualize the results[9] and to automate the process of execution optimization[10]. Optimizing execution further is an important research topic for the future.

5. Future Issues

When Stevia was first built, we expected the results of computation with Big Data to be relatively small, and assumed the initial HDFS capacity would be enough. However, as the types of programs diversified, it became clear that the size of some results were larger than expected, and the HDFS capacity became insufficient. We increased capacity by adding more slave nodes, but we encountered scalability issues with the NameNode. Policies for flexibly handling this sort of explosive data increase and rules for regulating newly generated data are needed.

Hadoop is fundamentally a batch processing system that shows its strength in analyzing Big Data at fixed intervals. However, there is also much demand for analyzing the most recent data immediately and displaying the results in real time. One way to satisfy such demand is to use data stream processing. In the future, we will study data stream processing methods as a part of our data processing infrastructure.

One issue with utilizing big data is gathering it together. Regardless of how much data there is, if it cannot be collected in one place so that it can be analyzed, it is of no use. Furthermore, gathering data that is generated daily from its sources without loss requires much designing, building and operating work. We also spent much time in designing and building systems that automatically and continually collect data. In the future, we will also work on implementing cost effective mechanisms for gathering such data, with due attention to handling data appropriately.

6. Conclusion

We have described Stevia, a Big Data processing infrastructure. Stevia is a petabyte class large-scale system supporting the creation of MSS, but it has been possible to maintain stable operation continuously with only a small operations staff, by utilizing the strengths of Hadoop. In the future we will study data processing infrastructures to handle even larger, exabyte-scale data, and aim to implement new services using the mobile network.

REFERENCES

[1] I. Horikoshi: "NTT DOCOMO Builds Giant Mining Facility," Nikkei Communications, pp. 30-31, Oct. 2009 (In Japanese).
[2] T. White (auth), R. Tamagawa, S. Kaneda (trans.): "Hadoop," O'Reilly Japan, Jan. 2010 (In Japanese).
[3] DRBD: "DRBD.jp by Thirdware Inc." http://www.drbd.jp.
[4] Heartbeat: "Heartbeat - Linux-HA." http://www.linux-ha.org/wiki/Heartbeat
[5] Nagios: "Nagios - The Industry Standard in IT Infrastructure Monitoring." http://www.Nagios.org
[6] Ganglia monitoring system: "Ganglia Monitoring System." http://Ganglia.sourceforge.net

[7] Cacti: "Cacti - The Complete RRD Tool-based Graphing Solution." http://www.Cacti.net/index.php
[8] KickStart: "Part VI. Advanced Installation and Deployment." http://www.centos.org/docs/5/html/5.2/Installation_Guide/pt-installadvanced-deployment.html
[9] N. Kawasaki, S. Tanaka, I. Okajima: "A method to visualize the communication and execution processes of the MapReduce program on large-scale distributed systems," IEICE technical report, Vol. 110, No. 448, NS2010-257, pp. 527-532, Mar. 2011 (In Japanese).
[10] M. Nakayama, K. Yamazaki, S. Tanaka: "Alleviation Technique for Data Skew of Reduce Phase with MapReduce Programming Model," IPSJ Journal, Vol. 53, No. 3, pp.1189-1203, Mar. 2012 (In Japanese).