Noname manuscript No.
(will be inserted by the editor)

Designing a Hadoop system based on computational resources and network delay for wide area networks

Tomohiro Matsuno · Bijoy Chand Chatterjee · Nattapong Kitsuwan · Eiji Oki · Malathi Veeraraghavan · Satoru Okamoto · and Naoaki Yamanaka

Received: date / Accepted: date

Abstract This paper proposes a Hadoop system that considers both slave server's processing capacity and network delay for wide area networks to reduce the job processing time. The task allocation scheme in the proposed Hadoop system divides each individual job into multiple tasks using suitable splitting ratios and then allocates the tasks to different slaves according to the computational capability of each server and the availability of network resources. We incorporate software-defined networking (SDN) to the proposed Hadoop system to manage path computation elements and network resources. The performance of proposed Hadoop system is experimentally evaluated with fourteen machines located in the different parts of the globe using a scale-out approach. A scale-out experiment using the proposed and conventional Hadoop systems is conducted by executing both single job and multiple jobs. The practical testbed and simulation results indicate that the proposed Hadoop system is effective compared to the conventional Hadoop system in terms of processing time.

Keywords Hadoop · Heterogeneous clusters · Job-tracker · Implementation.

This work was supported in part by the national institute of information and communications technology (NICT), Japan and by an NSF grant (CNS-1405171), USA. Parts of this paper were partially presented at The 21st Asia-pacific conference on communications (APCC 2015) and the 2016 IEEE 17th International conference on high performance switching and routing (HPSR 2016).

T. Matsuno · N. Kitsuwan
The department of computer and network engineering, The university of electro-communications, Chofugaoka 1-5-1 Tokyo 182-8585 Japan.
B.C. Chatterjee
Indraprastha institute of information technology, Delhi, India. He was previously with The university of electro-communications, Tokyo.
E. Oki
Graduate school of informatics, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan. He was previously with the university of electro-communications, Tokyo.
M. Veeraraghavan
The department of electrical and computer engineering, University of virginia, USA.
S. Okamoto · N. Yamanaka
The department of information and computer science, Keio university, Japan.
E-mail: kitsuwan@uec.ac.jp

1 Introduction

Internet usage is rapidly growing due to the explosive expansion of social media. People use social media to advertise their activities through live broadcasting and video sharing. The size of the data in a network is enormous and needs to be processed and stored in an efficient manner [1]. The term "big data" is used to describe huge data sets that may be analyzed computationally to expose patterns, tendencies, and relations, especially relating to human behavior and interactions. Big data is defined by three Vs: volume, velocity, and variety. Volume indicates that a huge amount of data is generated and stored. Velocity indicates the data that are often available in real time. Variety represents that all types of data formats, including structured and unstructured data, need to be processed. An efficient way to manage this type of data is necessary [2].

Hadoop [3] is an open-source software used for storing and executing data in a distributed architecture across clusters of commodity hardware [4–6]. Hadoop provides huge storage capacity for any type of data, unlimited processing power and the capability to manage virtually limitless parallel jobs or tasks. Hadoop supports scale-out properties by increasing the number of slave servers to improve system performance. Hadoop involves two main modules, namely, the Hadoop distributed file system (HDFS) and MapReduce. HDFS [7–9] is a typical file structure in the Hadoop framework that splits a huge job into a number of tasks to increase the speed of the reading and writing processes. MapReduce [10–14] is responsible for parallel job execution on grids or computer clusters; a MapReduce job is a unit of work that is identified for each user. Figure 1 shows the workflow of MapReduce. Hadoop requires effective task scheduling mechanisms to complete each transaction submitted by users. The terms *task* and *job* represent two dissimilar aspects in the Hadoop system [15]. When a transaction is submitted by a user, Hadoop generates a job and places it in the queue of jobs pending services. A job involves numerous activities, which are managed in parallel. A task is executed when Hadoop allocates it to a central processing unit (CPU). This separation of jobs into tasks enables effective system resource utilization.

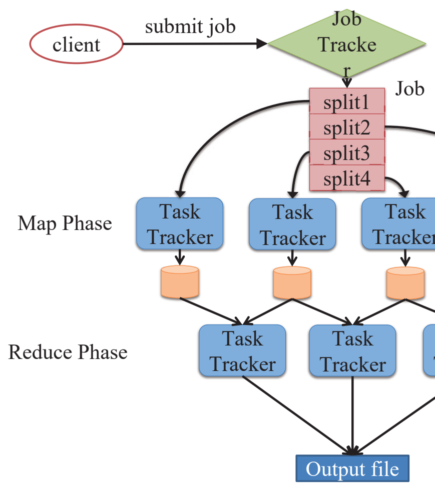


Fig. 1 Workflow of MapReduce [22].

Yet another resource negotiator (YARN) [3, 16], which is also known as an operating system of Hadoop, acts as a resource manager in the Hadoop system. YARN manages and monitors workloads, maintains a multi-tenant atmosphere by executing security controls, and handles the high-availability behaviors of the Hadoop system.

A homogeneous cluster is an environment that consists of all the computers with the same specifications, whereas in a heterogeneous cluster, the specifications of all the computers are different. On university campuses, a cluster typically consists of both outdated and modern resources for executing engineering and scientific applications. Moreover, the performance of network resources, which connect different computers through switches, may be different. In the existing Hadoop system, a job is split into equally sized tasks, which are then allocated by a scheduler, for example, by YARN in MapReduce 2.0 and jobtracker in MapReduce 1.0, to different slaves. YARN is not responsible for node processing. If the computational capabilities of the nodes in a cluster differ [17–19], the time for executing a job may not be shortened, even if the number of resources is increased. This indicates that we cannot utilize the advantages of Hadoop when low-performance computational resources are involved.

A resource-allocation scheme, which splits a job into multiple tasks using unequal ratios and then allocates these unequal-sized tasks to different slaves according to the computational capability of each server and the availability of network resources, was presented by T. Matsuno et al. [20, 21] to resolve the shortcomings of the current Hadoop. Thus, the processing time is reduced, even when outdated computational resources are involved in the task allocation process. In previous research, a theoretical analysis was performed to shorten the longest processing time of the slaves, and a simulation study was conducted to evaluate the performance of the scheme presented in [21]. The scheme was not assessed experimentally in a practical testbed scenario.

This paper proposes a Hadoop system that considers both the slave server's processing capacity and the network delay for wide area networks to shorten the job processing time. The task allocation scheme in the Hadoop system divides each individual job into different tasks using suitable splitting ratios and then allocates the different tasks to different slaves according to the computational capability of each server and the availability of network resources. The work presented in [22] is extended in this paper along with various additions; the extensions are mainly presented in the following sections. We first broadly explain the current literature on Hadoop systems. We then implement the proposed Hadoop system in a testbed environment, where the scale-out performance is experimentally evaluated using fourteen machines, which are placed in different locations around the world. Furthermore, the scale-out experiment is conducted by executing multiple jobs. To measure the network delay, a layer 2 virtual private network (L2-VPN) is created among The University of Electro-Communications (UEC), Japan; the National Institute of Information and Communications Technology (NICT), Japan; Keio University, Japan; and the University of Virginia (UVA), USA. We evaluate the performance of the proposed Hadoop system in terms of the processing time under different conditions. The experimental results demonstrate that the proposed Hadoop system outperforms the conventional Hadoop system, in which a job is divided into equally sized tasks that are assigned to slave servers without consideration of processing performance and network delay. A Hadoop system with software-defined network (SDN) was described in [22]. The experiment of the work in [22] was performed within a local network; it did not consider the wide area network environment.

The main contributions of this work are as follows.

(i) We design a Hadoop framework based on computational resources and network delay.
(ii) We implement the proposed system in a testbed Hadoop environment considering wide area networks.
(iii) We formalize the resource allocation scheme for heterogeneous Hadoop clusters.
(iv) We incorporate SDN to the Hadoop system in order to manage path computation elements and network resources.

The remainder of this paper is structured as follows. Section 2 describes the related work on Hadoop systems. The task allocation scheme for heterogeneous Hadoop clusters of the proposed Hadoop system is presented in section 4. Section 5 explains the design of the proposed Hadoop system. The performance of the proposed Hadoop system is evaluated in Section 6.2.1. Finally, section 7 summarizes this paper.

2 Related work

The default Hadoop scheduler processes data for tasks step-by-step and allocates them to idle nodes, thereby increasing the job completion time. M. Zahari et al. [23] presented a scheduling scheme that considers delay to explore the conflict between fairness and data locality. In their approach, when a job is scheduled, it is delayed for a small amount of time, allowing other jobs to proceed. As a result, the delay scheduling achieves nearly optimum data locality in a diversity of workloads; hence, the throughput in the network is improved. However, the introduced delay leads to both instability and under-utilization of network resources.

J. Tan et al. [24] observed that map and reduce tasks are not jointly optimized in the current Hadoop schedulers, which causes job starvation and unfavorable data locality. To resolve this issue, they introduced a coupling scheduler, where both map and reduce tasks are combined by utilizing wait scheduling and random peeking scheduling for MapTasks and ReduceTasks. As Hadoop assumes that all clusters are devoted to a single user, it is unable to ensure the highest performance in an environment where resources are shared.

To resolve the issue introduced in [24], S. Seo et al. [25] proposed prefetching and preshuffling schemes to enhance the overall system performance in a shared MapReduce scenario. The prefetching scheme performs data locality, and the network overhead is significantly reduced by the preshuffling scheme.

J. Jin et al. [26] introduced an effective task-scheduling algorithm that considers data locality for the cloud environment to reduce the job execution time. In their work, an initial task allocation scheme is triggered first, and then the job execution time is progressively reduced by tuning the initial task allocations. Finally, by considering the global view, the scheduling scheme dynamically adjusts the data locality according to the cluster workload and network state information.

M.J. Fischer et al. [27] presented a task allocation scheme to effectively allocate tasks to different clusters of Hadoop. They investigated task allocation in Hadoop and presented an idealized Hadoop system to estimate the cost of task allocation. They proved that the Hadoop task allocation problem is NP-complete, and presented a flow-based algorithm.

Software-defined networking (SDN) has recently been incorporated into Hadoop and big data to improve system performance. G. Wang et al. [28] investigated the impact of an SDN controller for optical switching to realize the tight integration of an application and the control system. They focused on the run-time network configuration for big data applications to optimize both application performance and resource utilization. Similarly, P. Qin et al. [29] presented a heuristic bandwidth-aware scheduling scheme, which reduces the job processing time in the network, to integrate Hadoop and SDN.

The related research proposed different efficient task allocation schemes. However, Hadoop task allocation for heterogeneous clusters in wide area networks is not considered. This paper designs a task allocation scheme for wide area networks that considers each server's computational capability and the availability of network resources.

3 Software-defined network

SDN makes the current application requirements possible [30, 31]. It allows network engineers and administrators to respond quickly to change requirements. The network administrator can flexibly manage the network with a centralized controller, which runs all the intelligent control and management software. SDN separates the network control and data forwarding planes to make it easier to optimize, regardless of vendors or model of switches, as shown in Fig. 2. The control plane is moved to a centralized controller, which performs a network brain to make a decision and instruction for a new request. The switch remains the data forwarding plane. The packet is processed by the data forwarding plane of the switch, as the instruction from the controller.

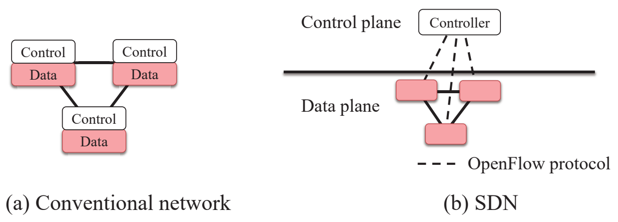


Fig. 2 Structure of conventional network and SDN network.

OpenFlow [32] is a common protocol used in SDN to enable the controller to interact with the data forwarding plane in the switches. A switch generates a packet-in to inform the controller when links go down or when a packet arrives without specified forwarding instructions. The forwarding instructions are based on a flow, which is defined by a set of specific parameters, such as source and destination Ethernet/Internet Protocol (IP) addresses, switch input port, virtual local area network (VLAN) tag, etc, as shown in Fig. 3. The controller specifies the set of parameters that define each flow and how packets that match the flow should be processed.

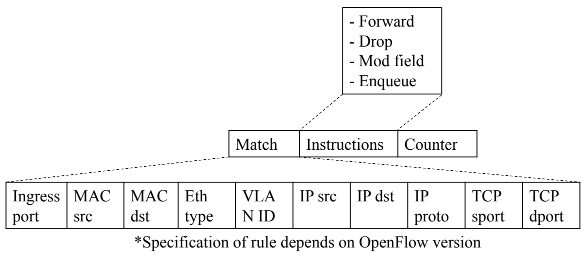


Fig. 3 Flow entry structure.

4 Task allocation scheme in heterogeneous Hadoop clusters

This section presents the task allocation scheme in heterogeneous Hadoop clusters for the proposed Hadoop system.

4.1 Concept of the proposed Hadoop system

The task allocation scheme of the proposed Hadoop system is intended to enhance the performance, in terms of processing time, by employing network and computational resources. If multiple servers with different performance are used, and the job is split with appropriate ratios according to the computer processing capability and the availability of network resources to reduce the overall processing time. The processing time is reduced by minimizing the maximum processing time of all the slaves. The task allocation scheme estimates the appropriate splitting ratios based on the availability of network resources and the processing capabilities of the slaves.

Table 1 Notation used in this paper

Symbol	Meaning
I	Set of tasks for a job
r	Maximum processing time among all tasks $i \in I$
x_i	Split ratio of task $i \in I$, where $0 \leq x_i \leq 1$
m_i	Indication whether to use resource i, where $m_i = 1$ if $x_i > 0$ and $m_i = 0$ otherwise
c_i	Processing capacity of resource i
$\boldsymbol{D}$	Delay, where $\boldsymbol{D} = \{d_1, d_2, d_3, ..., d_{\lvert I \rvert}\}$
$\boldsymbol{R}$	Routes from master to slave via the network, where $\boldsymbol{R} = \{\chi_1, \chi_2, \chi_3, ..., \chi_{\lvert I \rvert}\}$
M	Maximum number of used slave servers
$f(x_i, c_i, d_i)$	Processing time of task $i \in I$
U	Large value

Figure 4 illustrates the task allocation scheme, and Table 1 summarizes the notation used in this paper.

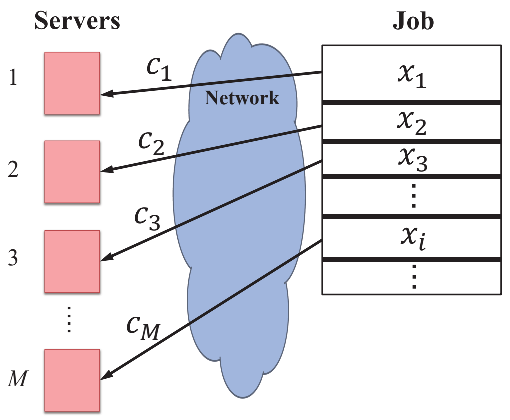


Fig. 4 Overview of the task allocation scheme [22].

4.2 Mathematical formulation

This subsection presents an integer linear programming (ILP) formulation [33] of the task allocation scheme for the proposed Hadoop system. The objective of this work is to minimize the longest processing time among all slave servers. In the following, we define the objective function.

$$\begin{aligned}
&\min \quad r && \text{(1a)}\\
\text{s.t.}\quad & \boldsymbol{D} = y(\boldsymbol{R}) && \text{(1b)}\\
& f(x_i, c_i, d_i) \leq r \qquad \forall i \in I && \text{(1c)}\\
& \sum_{i \in I} x_i = 1 && \text{(1d)}\\
& \sum_{i \in I} m_i \leq M && \text{(1e)}\\
& x_i \leq m_i \qquad \forall i \in I && \text{(1f)}\\
& m_i \leq U x_i \qquad \forall i \in I && \text{(1g)}\\
& 0 \leq x_i \leq 1 \qquad \forall i \in I && \text{(1h)}\\
& m_i = \{0, 1\} \qquad \forall i \in I && \text{(1i)}
\end{aligned}$$

Appropriate routes between the master and slaves must be identified. Eq. (1b) indicates that the delay depends on the selection of suitable routes between the master and slaves. The processing time of task i does not exceed r, which is expressed in Eq. (1c). To reduce the job processing time $f(x_i, c_i, d_i)$ of task i, $\forall i \in I$, we divide the job with appropriate splitting ratios, as mentioned previously. Eq. (1d) shows that the sum of the proportions of the divided tasks must equal one. Eq. (1e) states that the total number of used slaves must be less than or equal to M. Eqs. (1f) and (1g) express, in a linear form, that m_i is equal to 1 if $x_i > 0$ and is zero if $x_i = 0$. Eq. (1h) shows that the splitting ratio is between 0 and 1. Finally, the last constraint in Eq. (1i) is used to express the binary variable. The splitting ratio is obtained from this ILP formulation.

Note that the functions $f()$ and $y()$ of the introduced model, which are presented in section 3.2, will be mapped in our experiment. The delay will be assessed considering the location: a higher delay indicates a longer distance between the master and slave.

5 Design of the Hadoop system

This section presents the architecture and experimental setup of the proposed Hadoop system.

5.1 Hadoop system architecture

Hadoop manages and analyzes the information in the datacenter. To setup efficient Hadoop clusters, a collaboration among servers, networking, and application teams is required. It may take several days to weeks if the orchestration and robust tools are not well organized [34]. SDN and its associated management tools are used to execute the job efficiently in the Hadoop clusters. The management tool includes failure management [35], bandwidth-aware scheduling [36], and Hadoop mapreduce [37].

An overview of the path setup between the master and slaves is shown in Figure 5. OpenFlow switches are placed on both sides, and these switches are connected to an SDN controller. An SDN controller is connected to a path computation element (PCE) [38–41], which selects the routes. The master node and PCE are connected to the Hadoop scheduler for job and task scheduling. For master node communication with the Hadoop clusters, the SDN switches, located on the two sides, are connected through optical paths, L2-VPN or the Internet. The SDN controller dynamically controls the route from the master to slave to permit the master to communicate with different slaves. Each slave has its own SDN switch, which is connected to another SDN switch placed on the master side. PCE determines the routes based on traffic conditions.

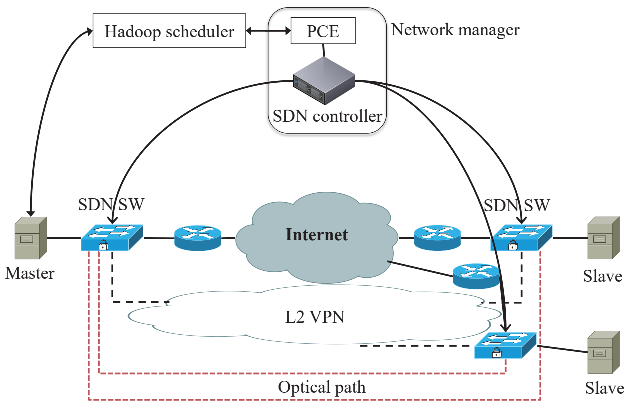
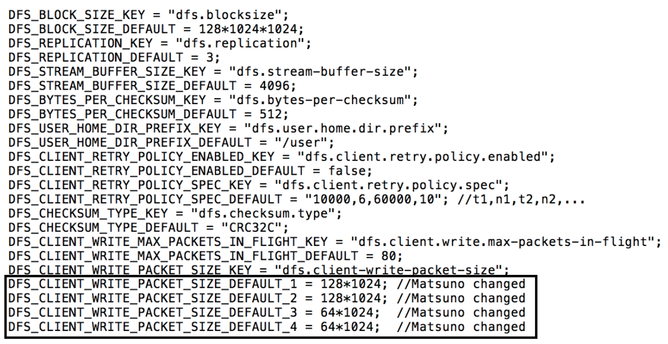


Fig. 5 Overview of the path setup from master to slaves [22].

We modified the Hadoop system to divide the job with suitable splitting ratios according to the availability of network resources and the processing capability of the computers. The source code of Hadoop 2.7.1 [42], which contains several directories, e.g. client, MapReduce, HDFS, and assemblies, was installed for our experiment. The configurations of the tasktracker and jobtracker in the HDFS directory were the focus of the study. We modified the DFSConfigKeys.java file to change the block size of the tasks; the default block size of tasks is 64 MB. The modified DFSConfigKeys.java file is shown in Fig. 6; the code displayed in a black box was reconfigured. The default block size of the tasks in DFSConfigKeys.java can be restructured using an automated function that depends on the input parameters.



```
DFS_BLOCK_SIZE_KEY = "dfs.blocksize";
DFS_BLOCK_SIZE_DEFAULT = 128*1024*1024;
DFS_REPLICATION_KEY = "dfs.replication";
DFS_REPLICATION_DEFAULT = 3;
DFS_STREAM_BUFFER_SIZE_KEY = "dfs.stream-buffer-size";
DFS_STREAM_BUFFER_SIZE_DEFAULT = 4096;
DFS_BYTES_PER_CHECKSUM_KEY = "dfs.bytes-per-checksum";
DFS_BYTES_PER_CHECKSUM_DEFAULT = 512;
DFS_USER_HOME_DIR_PREFIX_KEY = "dfs.user.home.dir.prefix";
DFS_USER_HOME_DIR_PREFIX_DEFAULT = "/user";
DFS_CLIENT_RETRY_POLICY_ENABLED_KEY = "dfs.client.retry.policy.enabled";
DFS_CLIENT_RETRY_POLICY_ENABLED_DEFAULT = false;
DFS_CLIENT_RETRY_POLICY_SPEC_KEY = "dfs.client.retry.policy.spec";
DFS_CLIENT_RETRY_POLICY_SPEC_DEFAULT = "10000,6,60000,10"; //t1,n1,t2,n2,...
DFS_CHECKSUM_TYPE_KEY = "dfs.checksum.type";
DFS_CHECKSUM_TYPE_DEFAULT = "CRC32C";
DFS_CLIENT_WRITE_MAX_PACKETS_IN_FLIGHT_KEY = "dfs.client.write.max-packets-in-flight";
DFS_CLIENT_WRITE_MAX_PACKETS_IN_FLIGHT_DEFAULT = 80;
DFS_CLIENT_WRITE_PACKET_SIZE_KEY = "dfs.client-write-packet-size";
DFS_CLIENT_WRITE_PACKET_SIZE_DEFAULT_1 = 128*1024; //Matsuno changed
DFS_CLIENT_WRITE_PACKET_SIZE_DEFAULT_2 = 128*1024; //Matsuno changed
DFS_CLIENT_WRITE_PACKET_SIZE_DEFAULT_3 = 64*1024;  //Matsuno changed
DFS_CLIENT_WRITE_PACKET_SIZE_DEFAULT_4 = 64*1024;  //Matsuno changed
```

Fig. 6 Source code modified in DFSConfigKeys.java.

5.2 Experimental setup for Hadoop

A virtual scenario was created with fourteen machines using VMware Player software [43]. We considered one machine among fourteen machines as the master server and the remaining thirteen machines as slave servers, as shown in Fig. 7. Windows 7 was used as the operating system (OS) of the host. VMware Player [43,44] was installed as the guest OS on the host. The specifications of the machines used in our experiments are shown in Table 2. We used a Machin-like formula [45,46], which is a standard technique for calculating an inverse trigonometric function. The sum of infinite series was also calculated. One master server and ten slave servers were used to evaluate 100 million digits of π. To create a heterogeneous cluster, we assigned the same π calculation job as a load to the slave servers, except for the first and second slave machines, to make them overloaded.

To compare our proposed Hadoop system with the conventional Hadoop system, which splits a job into equal tasks without considering the computer processing performance, we consider a system with five machines: one master server and four slave servers. We consider five cases, which are shown in Fig. 8. Case 1: Homogeneous cluster with the conventional Hadoop system: 64 MB is considered as the block size for all tasks. Case 2: Heterogeneous cluster with the conventional Hadoop system: 64 MB is considered as the block size for all tasks. Case 3: Heterogeneous cluster with the modified Hadoop system. In this case, we adjust the task block size with the constraints of 128 MB for first and second slaves and 64 MB for the third and fourth slaves. Case 4: Heterogeneous clusters with the modified Hadoop system. In this case, we adjust the task block size with constraints of 64 MB for the first and second slaves and 32 MB for the third and fourth slaves. Case 5: Heterogeneous clusters with the modified Hadoop system. In this case, we adjust task the block size with constraints of 128 MB for the first and second slaves, 64 MB for the third slave and 32 MB for the fourth slave. The mathematical model determines the splitting

Table 2 Virtual machine specifications

	Master server
CPU	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz
Number of cores	2
Memory	2GB
	Slave servers 01 to 10
CPU	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz
Number of cores	1
Memory	2GB
	Slave server 11 at NICT
CPU	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz
Number of cores	1
Memory	2GB
	Slave server 12 at Keio
CPU	Xeon E5345 2.33 GHz
Number of cores	1
Memory	2GB
	Slave server 13 at UVA
CPU	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz
Number of corers	1
Memory	2GB

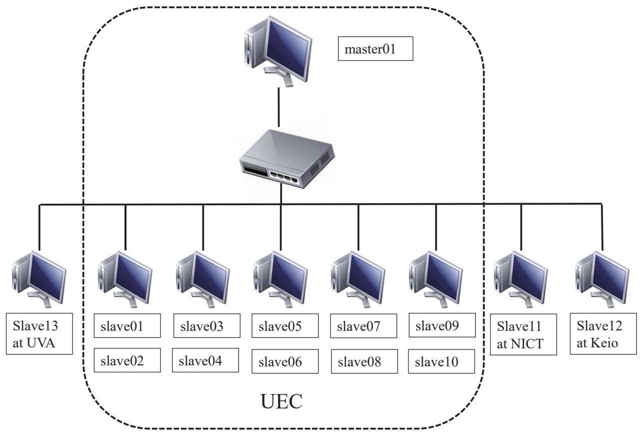


Fig. 7 Network diagram.

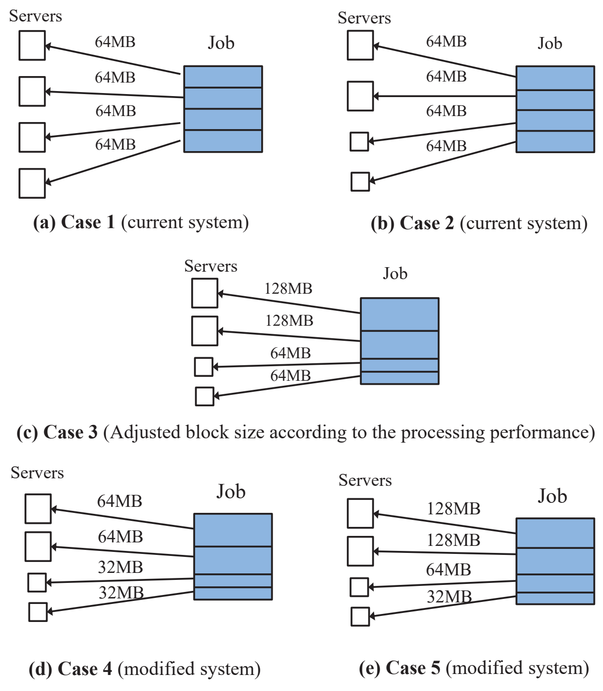

Servers
Job
64MB
128MB
32MB
(a) Case 1 (current system)
(b) Case 2 (current system)
(c) Case 3 (Adjusted block size according to the processing performance)
(d) Case 4 (modified system)
(e) Case 5 (modified system)

Fig. 8 Five experimental cases [22].

ratio of a task. These experiments confirm the impact of changing the task size according to a simple ratio. The slaves and master regularly exchange their survival information using the heartbeat. The slave's processing capacity can also be conveyed simultaneously with the heartbeat.

5.3 Experimental setup among different campuses

A layer 2 network was setup among Keio University, NICT, and UEC. UEC was connected to NICT through SINET 5 and JGN-X. Through NICT, UEC was also connected to Keio University. Finally, UEC was connected to UVA through Keio University. We intentionally did not connect each location directly through the layer 2 network because this network structure was used to measure the network delay from the master located at UEC to the slave located at UVA, as shown in Fig. 9.

We measured the round-trip time and throughput using ping and iperf. The throughput and round-trip time from UEC to NICT, from NICT to Keio University, and from UEC to Keio University via NICT were 383 Mb/s and 2.1 ms, 162 Mb/s and 5.7 ms and 126 Mb/s and 7.5 ms, respectively.

Table 3 shows the internet protocol (IP) address of the machines at different locations around the world.

Table 3 IP Addresses

Network address	10.1.1.0/24
NICT	10.1.1.186
UEC master	10.1.1.254
UEC slave	10.1.1.253
Keio University	10.1.1.131
UVA	10.1.1.120

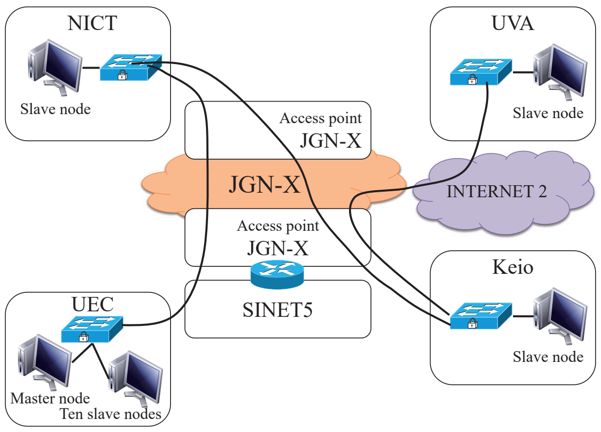


Fig. 9 Network topology.

6 Performance analysis

This section evaluates the performance of the proposed Hadoop system using both experimental and simulation studies.

6.1 Simulation results

This subsection compares the processing times of the conventional Hadoop system and the proposed Hadoop system via simulation. In our simulation, we consider 18 computers as slave servers, of which nine are modern high-performance computers and the other nine outdated low-performance computers. We summarize the simulation parameters in Table 4. In this evaluation, we assume that $f(x_i, c_i, d_i)$ is proportional to x_i, i.e., $f(x_i, c_i, d_i) = g(c_i, d_i)x_i \quad \forall i \in I$, where the factor of proportionality, $g(c_i, d_i)$, depends on c_i and d_i. $g(c_i, d_i)$ is a given parameter. r, x_i and m_i are decision variables. The value of $g(c_i, d_i)$ is the lowest normalized processing time, which is i=1. In this simulation, we consider heterogeneous clusters, where the machines have different specifications. We define a machine as a modern machine when $g(c_i, d_i) \leq 10$; otherwise, it is treated as an outdated machine.

The normalized processing times of r for conventional Hadoop and the proposed Hadoop for different numbers of slaves are shown in Fig. 10. We normalize

Table 4 Simulation parameters

i	$g(c_i, d_i)$	i	$g(c_i, d_i)$
1	1.000	10	10.000
2	1.527	11	11.054
3	2.054	12	12.108
4	2.581	13	13.162
5	3.108	14	14.216
6	3.635	15	15.270
7	4.162	16	16.324
8	4.689	17	17.378
9	5.216	18	18.432

the processing time of r by dividing by that of one slave server. When the number of slave servers increases, the normalized processing time of r for the proposed Hadoop system decreases owing to the appropriate allocation of the job to slave servers via splitting ratios. Furthermore, at the ninth slave server, the normalized processing time of r for the conventional Hadoop system suddenly increases because the conventional system does not work well when poor-performance slave servers are included.

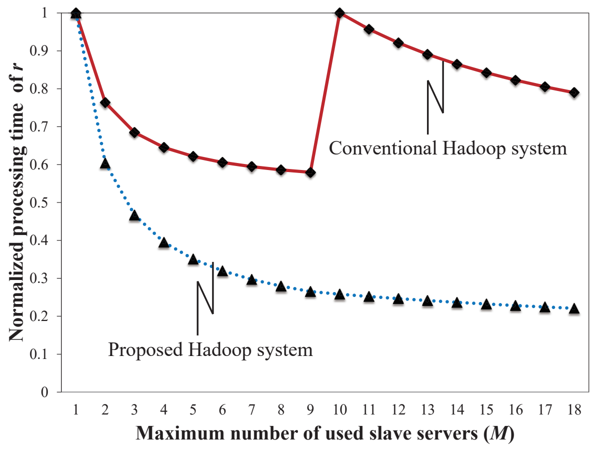


Fig. 10 Processing time of the proposed and conventional Hadoop systems.

6.2 Experimental results for a single job

The functions $f()$ and $y()$ of the proposed model, presented in section 3.2, are mapped in our experiment. We assessed the delay considering the location: a higher delay indicates a longer distance between the master and slave. In this subsection, we first focus our discussion on the experimental results for a single job. Then, we discuss the performance of the proposed and conventional schemes in the context of multiple jobs.

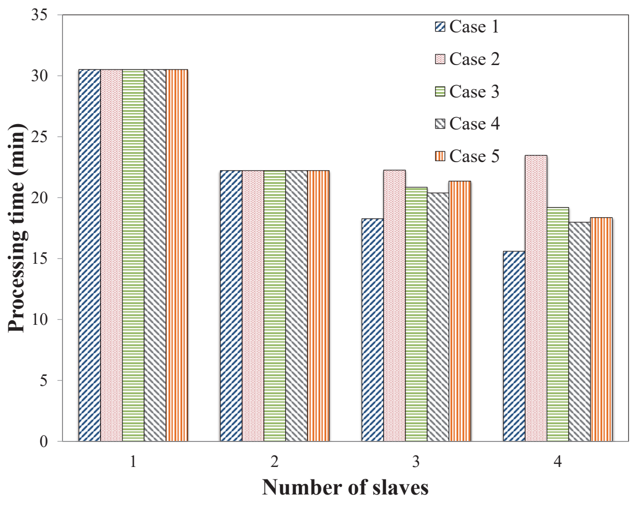


Fig. 11 Comparison of the five cases.

6.2.1 Experimental results with five machines

This section presents the experimental results of our proposed Hadoop system in terms of the job processing time. Figure 11 compares the performances of the five cases. In case 1, the job processing time decreases as the number of slave servers increases due to the proper distribution of the computational load among slave servers with an appropriate task size in the conventional Hadoop system. Nonetheless, when as number of slave servers increases, the processing time in case 2 increases because the additional load is assigned to the third and fourth slave servers. Therefore, the scale-out operation with four slaves is not successful. This issue is resolved by case 3, which divides a job into different sized tasks according to the availability of network resources and the computer processing capability. In case 3, the processing time decreases as the number of slave servers increases. When the number of slave servers increases in case 4, the processing time decreases. The processing time in case 5 decreases as the number of slave servers increases. The decreasing processing times are a result of the appropriate load distribution among slave servers achieved by the modified Hadoop system. Furthermore, in case 4, the processing time is shorter than those of other cases with heterogeneous Hadoop clusters as the tasks are allocated appropriately among the four slave servers. To extend this analysis, we measure the memory usage on each slave server for case 4. The memory usage of the first and second servers, which are high-performance machines, is 87%, which is equivalent to 1.74 GB. The memory usage of the third and fourth servers, which are low-performance machines, is 51%, which is equivalent to 1.02 GB. According to these measurements, the low-performance machines are given smaller tasks. In heterogeneous clusters, the results of case 4 are better than the results of the other cases. The results of case 4 are better than those of case 2 because the task size in case 4 is properly distributed by an appropriate splitting ratio based on the specifications of the machines. On the other hand, case 4 outperforms cases 3 and 5 because the task size in cases 3 and 5 is larger than that of case 4; therefore, the overhead processing time is increased in cases 3 and 5. By allocating smaller tasks to the low-performance slaves, the advantages of Hadoop are realized.

The processing time in case 1 is the best, followed by those of case 4, case 2, case 3, and case 5. In this experiment, only four slaves, which are placed at UEC, are considered. An experiment with more slaves placed in the different parts of the globe is desirable and is performed in the following subsection.

6.2.2 Experimental results with thirteen slave machines

This subsection evaluates the performance of the proposed Hadoop system and compares it to the conventional Hadoop system. Case 1, case 2, and case 4 are considered. Cases 3, 4, and 5 in the previous subsection involve the proposed Hadoop system. Case 4 provides the best performance among all cases using the proposed system. Therefore, we select case 4 for comparison with the conventional system. We use fourteen machines in this experiment. The first machine is used as a master and is located at UEC; the remaining thirteen machines are used as slaves. The first ten slave machines are located at UEC, and the eleventh, twelfth and thirteenth slave machines are located at NICT, Keio University, and the University of Virginia, respectively.

Figure 12 shows the experimental results of our proposed Hadoop system. In case 1, when machines from the additional locations are added, the processing time decreases because the load is distributed among slave servers with an appropriate task size in the conventional Hadoop system. Case 2, implements heterogeneous clusters with the conventional Hadoop system; the low-performance machines (i.e., the third to the thirteenth) result in a heterogeneous cluster. In this case, we observe three situations. (i) When we add the third to sixth machines, the processing time increases because these slave machines are heavily loaded. The system does not scale-out using slave machines one to six. The first and second slave machines wait until the heavy load is completely processed by the third to sixth slave machines. (ii) When we add the seventh to tenth machines, the processing time decreases because the heavy load is distributed among the other slave machines. The scale-out of the system works properly. (iii) When we add the eleventh to thirteenth slave machines, the processing time increases because network delay is included in the processing time. Case 4 implements heterogeneous clusters with the proposed Hadoop system. In this case, we adjust the task block size so that the first and second slaves are given 64-MB blocks while the third to thirteenth slaves are given 32-MB blocks. When the other slave machines at the additional locations are added, the processing time decreases due to the proper allocation of tasks among the slave servers.

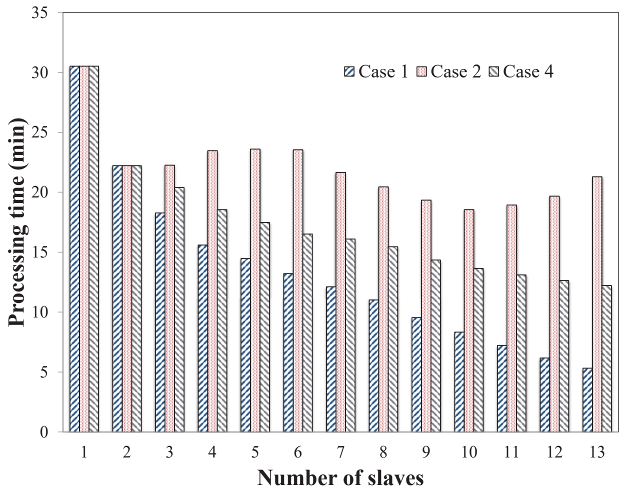


Fig. 12 Comparison of the single-job results for case 1, case 2, and case 4.

6.3 Experimental results for multiple jobs

This subsection performs a scale-out experiment considering multiple types of jobs: (i) π calculation, (ii) WordCount [47], and (iii) TeraSort [48]. The experimental environments are the same as those used for the π calculation considered in subsection 6.2.

Figure 13 shows the experimental results of our proposed Hadoop system considering multiple jobs. The observation of case 1 for multiple jobs is the same as that for a single job, see Fig. 12, except for the slave located at UVA. When we use the slave located at UVA, the TeraSort job, which has a larger impact on network delay than that of the π calculation considered in subsection 6.2, causes the processing time to increase. The observation of case 2 for multiple jobs is the same as that for a single job. The observation of case 4 for multiple jobs is the same as that for a single job, except for the slave located at UVA, because the impact of the TeraSort job on the network delay is larger than that of the π calculation considered in subsection 6.2; therefore, the incorporation of the slave located at UVA increases the processing time.

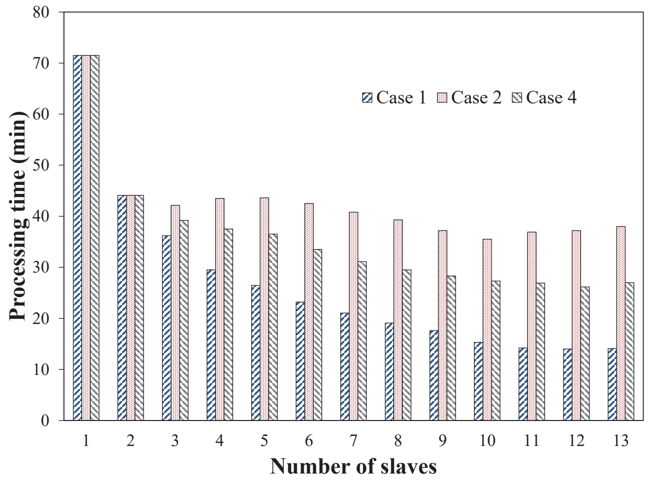


Fig. 13 Comparison of the multiple-job results for case 1, case 2, and case 4.

6.4 Processing time measurement for various slave locations

This subsection measures the processing times for various slave locations and investigates how the processing time is affected by the location. The master is placed at UEC to measure the processing times. We consider two slave servers with identical performance. The submitted job is to determine 100 million digits of π.

The processing time for the π calculation is 22 min 21 sec when both slaves are placed at UEC. When one slave is placed at UEC and the other is placed at NICT, the processing time is 22 min 32 sec. When one slave is placed at UEC and the other is placed at Keio University, the processing time is 22 min 46 sec. Finally, the processing is 23 min 56 sec when one slave is placed at UEC and the other is placed at UVA. As the distance between the master and a slave increases, the processing time also increases.

7 Conclusions

This paper proposed a Hadoop system to reduce the job processing time by considering both the slave server's processing capacity and network delay for wide area networks. The task allocation scheme in the Hadoop system divides each individual job into different tasks using suitable splitting ratios, and then allocates the different tasks to different slaves according to the computational capability of each server and the availability of network resources. The performance of the proposed Hadoop system was experimentally evaluated with fourteen machines using the scale-out approach. The configurations of the jobtracker and tasktracker in Hadoop system were the focus of our experiment. We allocated larger task blocks to high-performance slaves and smaller task blocks to low-performance slaves in heterogeneous Hadoop clusters. We created a scenario where high-performance slave servers execute more work than low-performance slave servers. The performance of the proposed Hadoop system was evaluated through experiments and simulations. In both cases, the proposed Hadoop system outperforms the conventional Hadoop system in terms of processing time.

In this work, we modified the DFSConfigKeys.java file to change the block size of the tasks; the default block size of the tasks in DFSConfigKeys.java is configured manually based on system requirment. The default block size of the tasks in DFSConfigKeys.java can be restructured using an automated function that depends on the input parameters, which is left as part of our future work.

References

1. Manikandan, S. & Ravi, S. (2014). Big Data Analysis Using Apache Hadoop. In *International Conference on IT Convergence and Security (ICITCS)* (pp. 1-4).
2. Dong, F. & Akl, S. G. (2006). Scheduling algorithms for grid computing: State of the art and open problems. Tech. Rep.
3. Apache Hadoop. http://hadoop.apache.org/.
4. Adnan M., Afzal M., Aslam M., Jan R., & Martinez-Enriquez A. (2014). Minimizing Big Data Problems using Cloud Computing Based on Hadoop Architecture. In *11th Annual High-capacity Optical Networks and Emerging/Enabling Technologies (HONET)* (pp. 99-103).
5. Cloudera Impala Project. http://impala.io/.
6. Cao, Z., Lin, J., Wan, C., Song, Y., Taylor, G., & Li, M. (2017). Hadoop-based framework for big data analysis of synchronised harmonics in active distribution network. In *IET Generation, Transmission & Distribution, 11*(16), 3930-3937. https://doi.org/10.1049/iet-gtd.2016.1723.
7. White, T. (2012). Hadoop: The Definitive Guide, Third Edition. O'Reilly Media, Inc.
8. Martin, B. (2014). SARAH-Statistical Analysis for Resource Allocation in Hadoop. In *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 777-782).
9. Chen, D., Chen, Y., Brownlow, B.N., Kanjamala, P.P., Arredondo, C.A.G., Radspinner, B.L., & Raveling, M.A. (2017). Real-Time or Near Real-Time Persisting Daily Healthcare Data Into HDFS and ElasticSearch Index Inside a Big Data Platform. In *IEEE Transactions on Industrial Informatics, 13*(2), 595-606. https://doi.org/10.1109/TII.2016.2645606

10. Palanisamy, B., Singh, A., & Liu, L. (2014). Cost-effective Resource Provisioning for MapReduce in a Cloud. In *IEEE Transactions on Parallel and Distributed Systems 26*(5), 1265-1279. https://doi.org/10.1109/TPDS.2014.2320498.
11. Zhao, Y., Wu, J., & Liu, C. (2014). Dache: A data aware caching for big-data applications using the MapReduce framework. In *Tsinghua Science and Technology 19*(1), 39-50. https://doi.org/10.1109/TST.2014.6733207.
12. Jung, H., & Nakazato, H. (2014). Dynamic Scheduling for Speculative Execution to Improve MapReduce Performance in Heterogeneous Environment. In *IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 119-124).
13. Hsiao, J. & Kao, S. (2014). A usage-aware scheduler for improving MapReduce performance in heterogeneous environments. In *International Conference on Information Science, Electronics and Electrical Engineering (ISEEE)* (pp. 1648-1652).
14. Zhu, N., Liu, X., Liu, J., & Hua, Y. (2014). Towards a Cost-efficient MapReduce: Mitigating Power Peaks for Hadoop Clusters. In *Tsinghua Science and Technology 19*(1), 24-32. https://doi.org/10.1109/TST.2014.6733205.
15. Xu, X., Cao, L., & Wang, X. (2014). Adaptive Task Scheduling Strategy Based on Dynamic Workload Adjustment for Heterogeneous Hadoop Clusters. In *IEEE Systems Journal 10*(2), 471-482. https://doi.org/10.1109/JSYST.2014.2323112.
16. Yao, Y., Wang, J., Sheng, B., Lin, J., & Mi, N. (2014). HaSTE: Hadoop YARN Scheduling Based on Task-Dependency and Resource-Demand. In *IEEE 7th International Conference on Cloud Computing (CLOUD)* (pp. 184-191).
17. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R.H., & Stoica, I. (2008). Improving MapReduce Performance in Heterogeneous Environments. In *8th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, (pp. 29-42).
18. Xiong, R., Luo, J., & Dong, F. (2014). SLDP: A Novel Data Placement Strategy for Large-Scale Heterogeneous Hadoop Cluster. In *Second International Conference on Advanced Cloud and Big Data (CBD)* (pp. 9-17).
19. Guo, Z. & Fox, G. (2012). Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)* (pp. 714-716).
20. Matsuno, T., Chatterjee, B.C., Oki, E., Okamoto, S., Yamanaka, N., & Veeraraghavan, M. (2015). Task Allocation Scheme for Hadoop in Campus Network Environment. In *IEICE Society Conference* (pp. B-12-20).
21. Matsuno, T., Chatterjee, B.C., Oki, E., Okamoto, S., Yamanaka, N., & Veeraraghavan, M. (2015). Resource Allocation Scheme for Hadoop in Campus Networks. In *21st Asia-Pacific Conference on Communications (APCC) (APCC 2015)* (pp. 596-597).
22. Matsuno, T., Chatterjee, B.C., Oki, E., Okamoto, S., Yamanaka, N., & Veeraraghavan, M. (2016). Task Allocation Scheme Based on Computational and Network Resources for Heterogeneous Hadoop Clusters. In *IEEE 17th International Conference on High Performance Switching and Routing (HPSR)* (pp. 200-205).
23. Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010). Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *5th European Conference on Computer Systems (EuroSys '10)* (pp. 265-278).
24. Tan, J., Meng, X., & Zhang, L. (2013). Coupling task progress for mapreduce resource-aware scheduling. In *IEEE INFOCOM* (pp. 1618-1626).
25. Seo, S., Jang, I., Woo, K., Kim, I., Kim, J.S., & Maeng, S. (2009).Hpmr: Prefetching and pre-shuffling in shared mapreduce computation environment. In *IEEE International Conference on Cluster Computing and Workshops* (pp. 1-8).
26. Jin, J., Luo, J., Song, A., Dong, F., & Xiong, R. (2011). Bar: an efficient data locality driven task scheduling algorithm for cloud computing. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 295-304).
27. Fischer, M.J., Su, X., & Yin, Y. (2010). Assigning tasks for efficiency in hadoop: Extended abstract. In *Twenty-second Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '10)* (pp. 30-39).
28. Wang, G., Ng, T.E., & Shaikh, A. (2012). Programming your network at run-time for big data applications. In *First Workshop on Hot Topics in Software Defined Networks (HotSDN '12)* (pp. 103-108).
29. Qin, P., Dai, B., Huang, B., & Xu, G. (2017). Bandwidth-Aware Scheduling With SDN in Hadoop: A New Trend for Big Data. In *IEEE Systems Journal, 11*(4), 2337-2344. https://doi.org/10.1109/JSYST.2015.2496368.
30. Zhu, T., Feng, D., Wang, F., Hua, Y., Shi, Q., Liu, J., Cheng, Y., & Wan, Y. (2017). Efficient Anonymous Communication in SDN-Based Data Center Networks. In *IEEE/ACM Transactions on Networking, 25*(6), 3767-3780. https://doi.org/10.1109/TNET.2017.2751616.
31. Ruffini, M., Slyne, F., Bluemm, C., Kitsuwan, N., & McGettrick, S. (2015). Software Defined Networking for Next Generation Converged Metro-Access Networks. In *Optical Fiber Technology, 26*(A), 31-41. https://doi.org/10.1016/j.yofte.2015.08.008.
32. OpenFlow. http://archive.openflow.org/.
33. Oki, E. (2013). Linear Programming and Algorithms for Communication Networks. CRC Press.
34. When SDN meets Hadoop big data analysis, things get dynamic. Retrieved January 20, 2018 from http://searchsdn.techtarget.com/opinion/When-SDN-meets-Hadoop-big-data-analysis-things-get-dynamic.
35. Kitsuwan, N., McGettrick, S., Slyne, F., Payne, D.B., & Ruffini, M. (2015). Independent transient plane design for protection in OpenFlow-based networks. In *IEEE/OSA Journal of Optical Communications and Networking, 7*(4), 264-275. https://doi.org/10.1364/JOCN.7.000264.
36. Qin, P., Dai, B., Huang, B., & Xu, G. (2017). Bandwidth-Aware Scheduling With SDN in Hadoop: A New Trend for Big Data. In *IEEE Systems Journal, 11*(4), 2337-2344. https://doi.org/10.1109/JSYST.2015.2496368.
37. Zhao, S. & Medhi, D. (2017). Application-Aware Network Design for Hadoop MapReduce Optimization Using Software-Defined Networking. In *IEEE Transactions on Network and Service Management, 14*(4), 804-816. https://doi.org/10.1109/TNSM.2017.2728519.
38. Le Roux, J.L., Ed. (2007). Path Computation Element Communication Protocol (PCECP) Specific Requirements for Inter-Area MPLS and GMPLS Traffic Engineering. IETF RFC 4927. https://tools.ietf.org/html/rfc4927.
39. Lee, Y., Le Roux, JL., King, D., & Oki, E. (2009). Path Computation Element Communication Protocol (PCEP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization. IETF RFC 5557. https://tools.ietf.org/html/rfc5557.
40. Oki, E., Inoue, I., & Shiomoto, K. (2007). Path Computation Element (PCE)-Based Traffic Engineering in MPLS

and GMPLS Networks. In *IEEE Sarnoff Symposium* (pp. 1-5).
41. Oki, E., Takada. T., Le Roux, JL., & Farrel, A. (2009). Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering. IETF RFC 5623. https://tools.ietf.org/html/rfc5623.
42. Apache Hadoop source code. Retrieved November 29, 2016 from http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.1/hadoop-2.7.1-src.tar.gz/.
43. VMware solution. Retrieved January 24, 2016 from http://www.vsolution.jp/.
44. Ishii, M., Han, J., & Makino, H. (2013). Design and Performance Evaluation for Hadoop Clusters on Virtualized Environment. In *International Conference on Information Networking (ICOIN)* (pp. 244-249).
45. Pi program. Retrieved January 24, 2016 from http://h2np.net/pi/mt-bbp.c.
46. Machin-Like Formulas. Retrieved November 29, 2016 from http://mathworld.wolfram.com/ Machin-LikeFormulas.html.
47. WordCount. Retrieved November 29, 2016 from http://hadoop.apache.org/docs/stable/ hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.
48. Apache Hadoop examples. Retrieved November 29, 2016 from http://hadoop.apache.org/docs/r2.7.2/api/org/apache/hadoop/examples/terasort/package-summary.html.

Tomohiro Matsuno received the B.E. degree in Information Science and Engineering from the University of Electro-Communications, Tokyo, Japan, in 2015. He is currently engaged with research work in the Graduate School in the Department of Communication Engineering and Informatics, the University of Electro-Communications, Tokyo, Japan. His research interests include resource allocation. He is a student member of IEICE, Japan.

Bijoy Chand Chatterjee received the Ph.D. degree from the Department of Computer Science & Engineering, Tezpur University in the year of 2014. From 2014 to 2017, he was a Postdoctoral Researcher in the Department of Communication Engineering and Informatics, the University of Electro-Communications, Tokyo, Japan, where he was engaged in researching and developing high-speed flexible optical backbone networks. Currently, he is working at Indraprastha Institute of Information Technology, Delhi, as a DST Inspire Faculty and Visiting Faculty. Dr. Chatterjee was the recipient of several prestigious awards, including DST Inspire Faculty Award in 2017, ERCIM Postdoctoral Research Fellowship by the European Research Consortium for Informatics and Mathematics in 2016, UEC Postdoctoral Research Fellowship by the University of Electro-Communications, Tokyo, Japan in 2014, and IETE Research Fellowship by the Institution of Electronics and Telecommunication Engineers, India in 2011. His research interests include QoS-aware protocols, cross-layer design, optical networks and elastic optical networks. He is a professional member of IEEE and life member of IETE, India.

Nattapong Kitsuwan received B.E. and M.E. degrees in electrical engineering (telecommunication) from Mahanakorn University of Technology, King Mongkut's Institute of Technology, Ladkrabang, Thailand, and a Ph.D. in information and communication engineering from the University of Electro-Communications, Japan, in 2000, 2004, and 2011, respectively. From 2002 to 2003, he was an exchange student at the University of Electro-Communications, Tokyo, Japan, where he performed research regarding optical packet switching. From 2003 to 2005, he was working for ROHM Integrated Semiconductor, Thailand, as an Information System Expert. He was a post-doctoral researcher at the University of Electro-Communications from 2011 to 2013. He worked as a researcher for the Telecommunications Research Centre (CTVR), Trinity College Dublin, Ireland from 2013 to 2015. Currently, he is an assistant professor at the University of Electro-Communications, Tokyo, Japan. His research focuses on optical network technologies, routing protocols, and software-defined networks.

Eiji Oki is a Professor at Kyoto University, Japan. He received the B.E. and M.E. degrees in instrumentation engineering and a Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. In 1993, he joined Nippon Telegraph and Telephone Corporation (NTT) Communication Switching Laboratories, Tokyo, Japan. He has been researching network design and control, traffic-control methods, and high-speed switching systems. From 2000 to 2001, he was a Visiting Scholar at the Polytechnic Institute of New York University, Brooklyn, New York, where he was involved in designing terabit switch/router systems. He was engaged in researching and developing high-speed optical IP backbone networks with NTT Laboratories. He was with The University of Electro-Communications, Tokyo, Japan from July 2008 to February 2017. He joined Kyoto University, Japan in March 2017. He has been active in standardization of path computation element (PCE) and GMPLS in the IETF. He wrote more than ten IETF RFCs. Prof. Oki was the recipient of the 1998 Switching System Research Award and the 1999 Excellent Paper Award presented by IEICE, the 2001 Asia-Pacific Outstanding Young Researcher Award presented by IEEE Communications Society for his contribution to broadband network, ATM, and optical IP technologies, the 2010 Telecom System Technology Prize by the Telecommunications Advanced Foundation, IEEE HPSR 2012 Outstanding Paper Award, IEEE HPSR 2014 Best Paper Award Finalist, First Runner Up, and the IEICE 2014 Achievement Award. He has authored/co-authored four books, *Broadband Packet Switching Technologies*, published by John Wiley, New York, in 2001, *GMPLS Technologies*, published by CRC Press, Boca Raton, FL, in 2005, *Advanced Internet Protocols, Services, and Applications*, published by Wiley, New York, in 2012, and *Linear Programming and Algorithms for Communication Networks*, CRC Press, Boca Raton, FL, in 2012. He is a Fellow of IEEE and IEICE Fellow.

Malathi Veeraraghavan is a Professor in the Charles L. Brown Department of Electrical & Computer Engineering at the University of Virginia (U.Va). Dr. Veeraraghavan received her BTech degree from Indian Institute of Technology (Madras) in 1984, and MS and Ph.D. degrees from Duke University in 1985 and 1988, respectively. After a ten-year career at Bell Laboratories, she served on the faculty at Polytechnic University, Brooklyn, New York from 1999-2002, where she won the Jacobs award for excellence in education in 2002. She served as Director of the Computer Engineering Program at U.Va from 2003-2006. Her current research work is on topics in high-speed networking, which includes optical, datacenter, virtual-circuit, and Grid networks, vehicular networks, and future Internet architectures. She holds twenty-nine patents, has over 90 publications and has received six Best paper awards. She served as the Technical Program Committee Co-Chair for the High-Speed Networking Symposium in IEEE ICC 2013, as Technical Program Committee Chair for IEEE ICC 2002 and Associate Editor for the IEEE/ACM Transactions on Networking. She was General Chair for IEEE Computer Communications Workshop in 2000, and served as an Area Editor for IEEE Communication Surveys. She served as Editor of IEEE ComSoc e-News and as an Associate Editor of the IEEE Transactions on Reliability from 1992-1994.

Satoru Okamoto is a Project Professor of the Keio University, Kanagawa, Japan. He received his B.E., M.E. and Ph.D. degrees in electronics engineering from Hokkaido University, Hokkaido, Japan, in 1986, 1988 and 1994. In 1988, he joined Nippon Telegraph and Telephone Corporation (NTT), Japan, where, he conducted research on ATM cross-connect system architectures, photonic switching systems, optical path network architectures, photonic network management systems, and photonic network control technologies. In 1999, he investigated the HIKARI router (photonic MPLS router) and the MPLambdaS principle. He is now researching future IP + optical network technologies, and application over photonic network technologies. He has led several GMPLS-related interoperability trials in Japan, such as the Photonic Internet Lab (PIL), the Optical Internetworking Forum (OIF) Worldwide Interoperability Demo, and the Kei-han-na Interoperability Working Group. He is a vice co-chair of the Interoperability Working Group of the Kei-han-na Info-communication Open Laboratory. He is now promoting several research projects related to photonic networks. He has published over 80 peer-reviewed journal and transaction articles, written over 150 international conference papers, and been awarded 50 patents including 5 international patents. He received the Young Researchers Award and the Achievement Award in 1995 and 2000 respectively from the IEICE of Japan. He also received the IEICE/IEEE HPSR2002 Outstanding Paper Award, Certification of Appreciation ISOCORE and PIL in 2008, IEICE Communications Society Best Paper Award and IEEE ISAS2011 Best Paper Award in 2011, and Certification of Appreciation for iPOP Conferences 2005-2014 in 2014. He was an associate editor of the IEICE Transactions on Communications (2006-2011) as well as the chair of the IEICE Technical Committee on Photonic Network (PN) (2010-2011), and was an associate editor of the Optical Express of the Optical Society of America (OSA) (2006-2012). He is an IEICE Fellow and an IEEE Senior Member.

Naoaki Yamanaka graduated from Keio University, Japan where he received B.E., M.E. and Ph.D. degrees in engineering in 1981, 1983 and 1991, respectively. In 1983 he joined Nippon Telegraph and Telephone Corporation's (NTT's) Communication Switching Laboratories, Tokyo Japan, where he was engaged in research and development of a high-speed switching system and high-speed switching technologies for Broadband ISDN services. Since 1994, he has been active in the development of ATM base backbone network and system including Tb/s electrical/optical backbone switching as NTT's Distinguished Technical Member. He moved to Keio University, Department of Information and Computer Science in 2004. He is now researching future optical IP network, and optical MPLS router system. He is currently a Professor in Dept. of Information and Computer Science, Keio University, Japan and representative of Photonic Internet Labs. He has published over 112 peer-reviewed journal and transaction articles, written 82 international conference papers, and been awarded 174 patents including 17 international patents

Dr. Yamanaka received Best of Conference Awards from the 40th, 44th, and 48th IEEE Electronic Components and Technology Conference in 1990, 1994 and 1998, TELECOM System Technology Prize from the Telecommunications Advancement Foundation in 1994, IEEE CPMT Transactions Part B: Best Transactions Paper Award in 1996 and IEICE Transaction Paper award in 1999. Dr. Yamanaka is Technical Editor of IEEE Communication Magazine, Broadband Network Area Editor of IEEE Communication Surveys, Former Editor of IEICE Transaction, TAC Chair of Asia Pacific Board at IEEE Communications Society as well as Board member of IEEE CPMT Society. Dr. Yamanaka is an IEEE Fellow and IEICE Fellow.