Hindawi
Scientific Programming
Volume 2017, Article ID 6972461, 13 pages
https://doi.org/10.1155/2017/6972461

Research Article

Development of Multiple Big Data Analytics Platforms with Rapid Response

Bao Rong Chang, Yun-Da Lee, and Po-Hao Liao

Department of Computer Science and Information Engineering, National University of Kaohsiung, 700 Kaohsiung University Rd., Nanzih District, Kaohsiung 811, Taiwan

Correspondence should be addressed to Bao Rong Chang; brchang@nuk.edu.tw

Received 6 April 2017; Accepted 28 May 2017; Published 21 June 2017

Academic Editor: Wenbing Zhao

Copyright © 2017 Bao Rong Chang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The crucial problem of the integration of multiple platforms is how to adapt for their own computing features so as to execute the assignments most efficiently and gain the best outcome. This paper introduced the new approaches to big data platform, RHhadoop and SparkR, and integrated them to form a high-performance big data analytics with multiple platforms as part of business intelligence (BI) to carry out rapid data retrieval and analytics with R programming. This paper aims to develop the optimization for job scheduling using MSHEFT algorithm and implement the optimized platform selection based on computing features for improving the system throughput significantly. In addition, users would simply give R commands rather than run Java or Scala program to perform the data retrieval and analytics in the proposed platforms. As a result, according to performance index calculated for various methods, although the optimized platform selection can reduce the execution time for the data retrieval and analytics significantly, furthermore scheduling optimization definitely increases the system efficiency a lot.

1. Introduction

Big data [1] has been sharply in progress unprecedentedly in recent years and is changing the operation for business as well as the decision-making for the enterprise. The huge amounts of data contain valuable information, such as the growth trend of system application and the correlation among systems. The undisclosed information may contain unknown knowledge and application that are discoverable further. However, big data with the features of high volume, high velocity, and high variety as well as in face of expanding incredible amounts of data, several issues emerging in big data such as storage, backup [2], management, processing, search [3], analytics, practical application, and other abilities to deal with the data also face new challenges. Unfortunately, those cannot be solved with traditional methods and thus it is worthwhile for us to continue exploring how to extract the valuable information from the huge amounts of data. According to the latest survey reported from American CIO magazine, 70% of IT operation has been done by batch processing in the business, which makes it "unable to control processing resources for operation as well as loading" [4]. This becomes one of the biggest challenges for big data application.

Hadoop distributes massive data collections across multiple nodes, enabling big data processing and analytics far more effectively than was possible previously. Spark, on the other hand, does not do distributed storage [5]. It is nothing but a data processing tool, operating on those distributed data collections. Furthermore, Hadoop includes not only a storage component called Hadoop Distributed File System (HDFS), but also a processing component called MapReduce. Spark does not come with its own file management system. Accordingly, it needs to be integrated with Hadoop to share HDFS. Hadoop processing mostly static and batch-mode style can be just fine and originally was designed to handle crawling and searching billions of web pages and collecting their information into a database [6]. If you need to do analytics on streaming data, or to run required multiple operations, Spark is suitable for those. As a matter of fact,

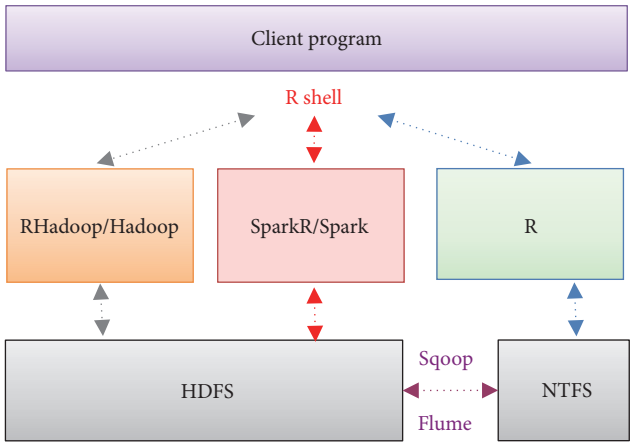


Figure 1: Data retrieval and data analytics Stack.

Spark was designed for Hadoop; therefore, data scientists all agree they are better together for a variety of big data applications in the real world.

Through establishing a set of multiple big data analytics platforms with high efficiency, high availability, and high scalability [7], this paper aims to integrate different big data platforms to achieve the compatibility with any existing business intelligence (BI) [8] together with related analytics tools so that the enterprise needs not change large amounts of software for such platforms. Therefore, the goal of this paper is to design the optimization for job scheduling using MSHEFT algorithm as well as to implement optimized platform selection, and established platforms support R command to execute data retrieval and data analytics in big data environment. In such a way the upper-level tools relying on relational database which has stored the original data can run on the introduced platforms through minor modification or even no modification to gain the advantages of high efficiency, high availability, and high scalability. I/O delay time can be shared through reliable distributed file system to speed-up the reading of a large amount of data. Data retrieval and data analytics stack has layered as shown in Figure 1. As a result, according to performance index calculated for various methods, we are able to check out whether or not the proposed approach can reduce the execution time for the data retrieval and analytics significantly.

2. Related Work in Big Data Processing

This paper has introduced data retrieval and data analytics using R programming in conjunction with RHadoop [9]/Hadoop [10] and SparkR [11]/Spark [12] platforms to build a multiple-platform big data analytics system. Furthermore, the use of distributed file system for fast data analytics and data storage reduces the execution time of processing a huge amount of data. First let us aim to understand the fundamental knowledge of Hadoop and Spark platforms and then build their extended systems RHadoop and SparkR for the purpose of fitting all kinds of relative problems on big data analytics. This section will introduce their related profiles and key technologies for both platforms accordingly.

2.1. Distributed Computing Framework with Hadoop. Hadoop is a well-known open source distributed computing framework as shown in Figure 2 that provides reliable, scalable, distributed computing, data storage, and cluster computing analytics of big data, including a MapReduce [13] for distributed computing, HDFS [14] distributed file system, and a distributed NoSQL database HBase [15] which can be used to store nonrelational data set. There are some tools that are based on Hadoop applications. First Apache Pig can perform complex MapReduce conversions on a huge amount of data using a simple scripting language called Pig Latin. Next Apache Hive [16] is a data warehousing package that lets you query and manage large datasets in distributed storage using a SQL-style language called HiveQL. Third Apache Sqoop is a tool for transferring large amounts of data between Hadoop and structured data storage as efficiently as possible. Further Apache Flume is a distributed and highly scalable log collection system that can be used for log data collection, log data processing, and log data transmission. Then Apache Zookeeper is a distributed application designed for the coordination of services, it is mainly used to solve the decentralized applications often encountered in some data management issues. Final Apache Avro is a data serialization system designed to support intensive data, the application of huge amounts of data exchange.

Examples of applications using Hadoop are given as follows. Caesars entertainment, a casino gaming company, has built a Hadoop environment [17] that differentiates customer groups and creates exclusive marketing campaign for each group. Healthcare technology company Cerner uses Hadoop to build a set of enterprise data centers [18] to help Cerner and their clients monitor the health of more than one million patients a day. The dating site eHarmony uses Hadoop to upgrade their cloud systems [19], enabling it to send millions of messages for matching friend dating every day.

2.2. Parallel Processing Framework with Spark. Spark is an open source parallel processing framework released by the Apache Software Foundation that supports in-memory processing and dramatically increases the execution speed of big data analytics, as shown in Figure 3. Spark is also designed for fast computing, high availability, and fault tolerance. Using its internal memory capabilities, Spark can be a great choice for machine learning and graph computation, as well as a great choice for big data analytics. Its main functions and positioning are the same as Hadoop MapReduce. Through In-memory cluster computing [20], it hopes to eliminate I/O latency caused by a lot of relay files swapped between memory and disk during MapReduce. Theoretically, the processing speed could be hundreds of times higher than the Hadoop. Spark is written in Scala, but also supports Scala, Java, and Python programming; the underlying storage system can also be directly compatible with HDFS.

Examples of Spark's applications are given as follows. Microsoft launched Spark for Azure HDInsight [21], allowing

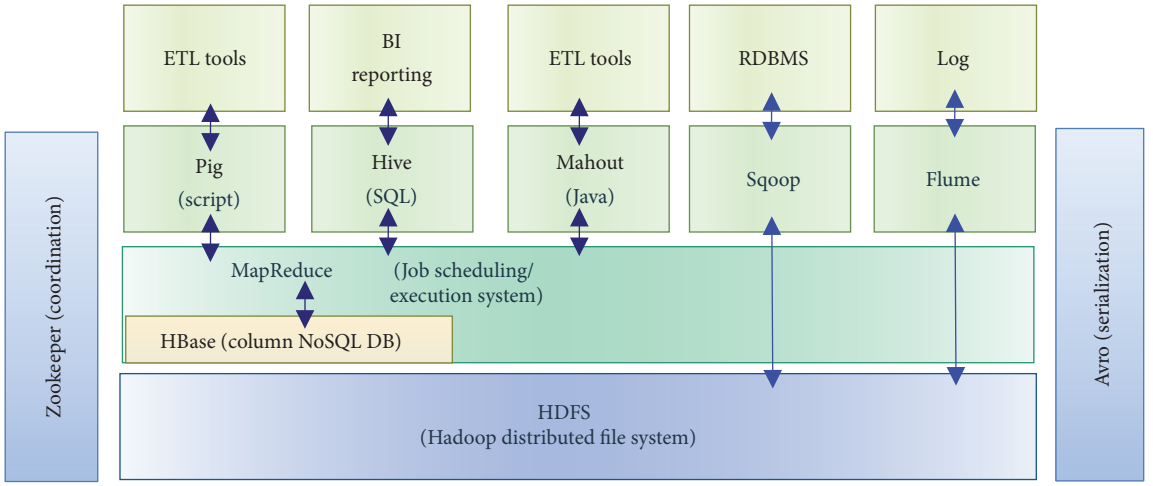


Figure 2: Hadoop framework.

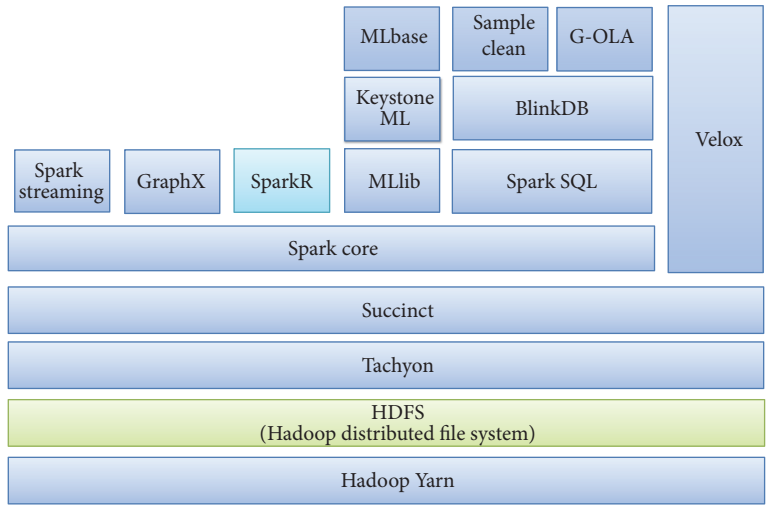


Figure 3: Spark framework.

users to use Spark for Azure HDInsight to solve big data challenges in near real-time, such as fraud detection, clickstream analytics, financial alerts, and more. Yahoo used Spark for the development of Audience Expansion in the application of advertising [22] to find the target user. Cloudera develops Spark Streaming's flexibility [23] to enable Cloudera's customers to build complete IoT applications on a unified platform.

2.3. Integrated Development Environment for R. Over the past decade, programming language R has been highly enhanced and greatly upgraded significantly to break the original limit in the past. In academy and industry, R becomes one of the most important tools for the research such as computational statistics, visualization, and data science. Millions of statisticians and data scientists use R to solve problems from counting biology to quantitative marketing. R has become one of the most popular programming language for the analytics of scientific data and finance. R is not only free, compact, and part of the open source that can run on many platforms, but also integrates data analytics and plotting functions all in one. It may add many additional packages to enhance system's functions, similarly comparable to the functions of commercial software, and can be viewed as one of major tools of contemporary data analytics. R is mainly used to analyze data, and thus the master node in a cluster installs R where big data access through HDFS has been available, or a stay alone computer for centralized processing installs R where small data access through NTFS has achieved. It is noted that data stored in NTFS can be transferred to HDFS via Sqoop [24]/Flume [25] or Hive.

2.4. RHadoop Based on Hadoop. Hadoop is capable of distributed computing and can store large amounts of data, but there is still a lot of information that needs to be analyzed professionally. However, R itself is not able to read the data size more than the size of memory in computer, and hence there is data size limit for processing big data. Therefore, it

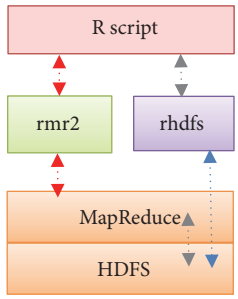


Figure 4: RHadoop framework.

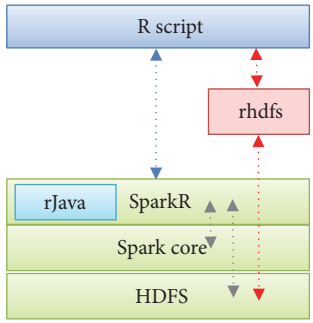


Figure 5: SparkR framework.

turns out the integration of Hadoop and R called RHadoop as a sort of data analytics service. In such a way, R will not only handle professional analytics, but it will also allow to easily utilize Hadoop features, such as the ability to access HDFS via rhdfs package and through the rmr2 package [26] to call MapReduce for accomplishing the distributed computing. The framework of RHadoop is shown in Figure 4.

2.5. SparkR Based on Spark. SparkR is an R suite developed by AMPLab that provides Spark with a Resilient Distributed Dataset (RDD) [27] API that allows R to carry out distributed computing using Spark. SparkR was merged into Spark in April 2015 and was released with Spark 1.4 in June 2015, so deploying SparkR requires installing Spark 1.4 or later and installing R related packages, including rJava [28] and rhdfs [29]. rJava lets R call objects, instances, and methods written in Java to make it less difficult for R to call Java-owned resources, such as Spark and Hadoop, and rhdfs, like RHadoop, to access HDFS. The framework of SparkR is shown in Figure 5. Although RHadoop mentioned above can activate distributed computing with R programming, its efficiency is not as good as SparkR. SparkR, adopting in-memory cluster computing, needs more memory resources than RHadoop. In order to avoid shutting down the task due to hardware resources limitation, both RHadoop and SparkR can be installed together for being interchangeably used at same site. In addition, in order to determine the most suitable analytical tools, we also need a matching algorithm to carry out the distributed computing successfully.

Table 1: Recipe of compatibility packages.

Software	Version
Hadoop (including RHadoop)	2.6.0
Spark (including SparkR)	1.4.0
R	3.2.2
Oracle Java (JDK)	8u66
Scala	2.10.4
rJava	0.9.7
rhdfs	1.0.8
rmr2	3.3.1

3. System Implementation Method

This paper aims to develop the optimization for job scheduling using MSHEFT algorithm so that system obtains the best throughput. After scheduling all of input queries in a job queue, system is then able to dispatch the job at top of the queue to one of big data analytics platforms through automatic platform selection. Regarding clustering and distributed parallel computing, a cloud computing foundation has been established to implement virtualization architecture because virtual machine has the feature of flexible control in hardware resource and thus it is quite suitable to act as a container provided an environment for the exploration of big data analytics.

3.1. Virtual Machine Deployment. Figure 6 shows a cloud computing [30] with high performance, high availability, and high scalability where server farm at the top layer and storage farm at the bottom layer are built for this study. In order to realize virtualization, an open source virtual machine management (VMM) or hypervisor Proxmox Virtual Environment (PVE) [31] based on KVM is used to implement virtual machine clustering; the status of virtual machine clustering can be effectively monitored through PVE, and the resource configuration of each virtual machine can be dynamically adjusted [32]. Since the platform performance is very closely related to I/O latency, the efficiency of both hard disk and network access should be increased in hardware configuration.

3.2. Recipe of Compatibility Packages. The most difficult aspect of integration of a lot of open source packages in a system is compatibility suite and that is one of the crucial problems of system integration as well. In this paper we proposed a recipe to resolve the challenge of suite compatibility. Several packages will be integrated to establish multiple big data analytics platforms in this paper and all of them are open source software, which are developed and maintained by different open source communities. A lot of software has complex dependency and compatibility problems. The recipe of packages proposed in this paper includes Hadoop, Spark, R, Scala, rJava, rhdfs, and rmr2, which are fully compatible for stable operation in the proposed approach as listed in Table 1.

3.3. Optimized Platform Selection. The program of automatic platform selection assigns a task to an appropriate big data

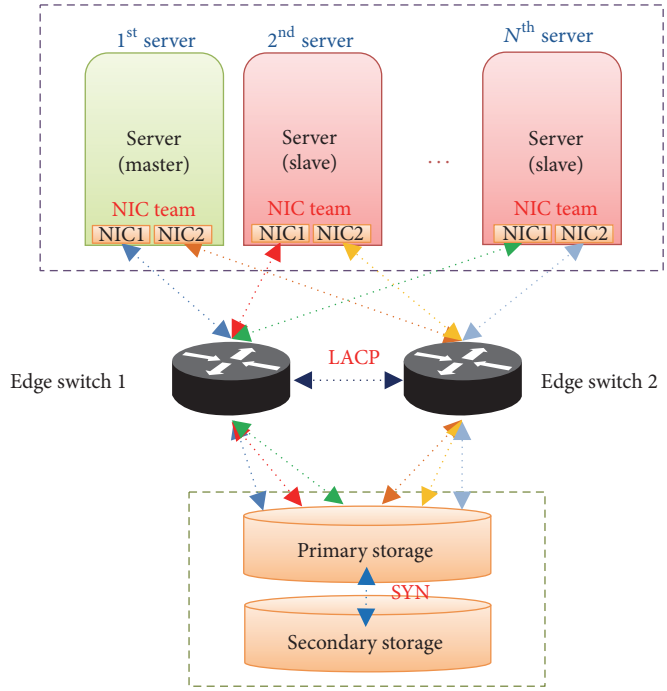


Figure 6: Cloud computing with high performance, high availability, and high scalability.

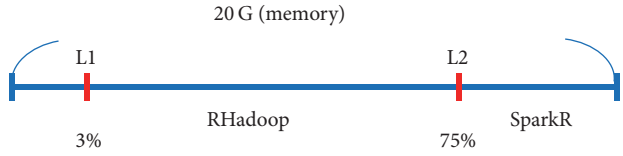


Figure 7: Automatic selection of suitable platform.

analytics platform according to the size of remaining amount of memory in a working virtual machine. The function and property for both RHadoop and SparkR are identical in a sense because they can access the same HDFS and support R syntax. Although these two platforms are the same function, they are different in the demand environment and executive manner. The memory size of 20 G for each server in the experiments is given, and it sets the remaining amount of memory size 0.6 G in a virtual machine in cluster denoted Level 1 (roughly 3% of total amount of memory) and 15 G Level 2 (approximately 75% of total amount of memory) as the cut-off points. In Figure 7, the program automatically chooses nothing to carry on the task as the remaining amount of memory is less than 3%; RHadoop would be applied as the remaining amount of memory lies between L1 and L2, and hence SparkR could be employed as the remaining amount of memory is higher than L2.

3.4. Optimization for Job Scheduling. Heterogeneous Earliest Finish Time (HEFT) [33] is an exploratory scheduling algorithm, which is used for scheduling the communication time of previous set of dependent task of heterogeneous network. HEFT is based on one of list scheduling algorithms, where their characteristics are to establish a priority list in the first step. According to the sorted priority list, HEFT assigns each task to a suitable CPU to make the task completed as soon as possible. The pseudocode of HEFT algorithm is shown in Algorithm 1. HEFT tries to search for local optimization and eventually makes the whole local optimums. In the test of automatic platform selection, the total of 20 GB memory is configured, and it is found that all of analytics platforms can be used when the remaining amount of memory is greater than or equal to L1; in addition, it is better to use RHadoop in case of being less than L2, and SparkR shall be used in case of being greater than L2. Job dispatched to RHadoop platform has run a kind of in-disk computing mode such that it may encounter data swap between disk and memory occasionally. Instead, in-memory computing mode has employed in SparkR platform and thus SparkR needs much more memory allocated for computing. HEFT algorithm is modified to Memory-Sensitive Heterogeneous Earliest Finish Time (MSHEFT) where the priority is considered first; then the size of data file is considered as the second condition, and finally an extra factor is considered, which is "remaining amount

(1) Compute rank_u for all nodes by traversing graph upward, starting from the exit node.
(2) Sort the nodes in a list by nonincreasing order of rank_u values.
(3) while there are unscheduled nodes in the list do
(4) begin
(5) Select the first task n_i in the list and remove it.
(6) Assign the task n_i to the processor p_j that minimizes the (EFT) value of n_i.
(7) end

Algorithm 1: The HEFT algorithm.

(1) Compute rank_u for all nodes by traversing graph upward, starting from the exit node.
(2) Sort the nodes in a list by nonincreasing order of rank_u values.
(3) while there are unscheduled nodes in the list do
(4) Compare priority.
(5) begin
(6) Compare job size
(7) Select the first task n_i in the list and remove it.
(8) begin
(9) if the remaining memory size > 0.6 GB
(10) begin
(11) what is the value of the remaining memory size?
(12) Assign the task n_i to the processor p_j that minimizes the (EFT) value of n_i.
(13) end if
(14) waiting the remaining memory size and go line 9.
(15) end
(16) end

Algorithm 2: The MSHEFT algorithm.

of memory." In Algorithm 2, the pseudocode of MSHEFT algorithm has been presented. Job processing flow chart is shown in Figure 8.

3.5. Execution Procedure. The execution procedure has been shown in Figure 9. With the user interface, the process is designated to monitor the status of each node in the server farm. MSHEFT algorithm for scheduling optimization together with platform selection has decided to choose an appropriate platform for execution according to the current status monitored through user interface. The proposed approach including MSHEFT algorithm plus platform selection can be denoted MSHEFT-PS in this paper. When the analytics task has finished, the results will be stored back to HDFS and the whole process will be terminated. In addition, job scheduling using first-come-first-serve FCFS will be adopted for each single analytics platform Rhadoop or SparkR, denoted FCFS-SP, in the experiment to check how it performs as a single platform applied. Furthermore, the platform selection mechanism integrated FCFS, denoted FCFS-PS, has also been employed to test the system performance under the condition of remaining amount of memory in a virtual machine in which a certain node has been resident.

3.6. Performance Evaluation. In order to compare the computation efficiency among the several algorithms, the performance index [2] has been evaluated based on the necessitated equations, which are derived first from measuring access time of data of a single item for a certain dataset on (1), next calculating average access time based on a variety of data size among the datasets on (2), then inducing a normalized performance index among the datasets on (3), and finally resulting in a performance index according to a series of tests on (4). In these equations we denote the subscript i the index of data size, j the index of dataset, and k the index of test condition and the subscript s indicates a single item in a specific dataset. Eq. (1) calculates the average access time (AAT) for each data size. In (1), AAT_{ijk} represents average access time with the same data size, and N_{ik} stands for the current data size. Eq. (2) calculates the average access times overall

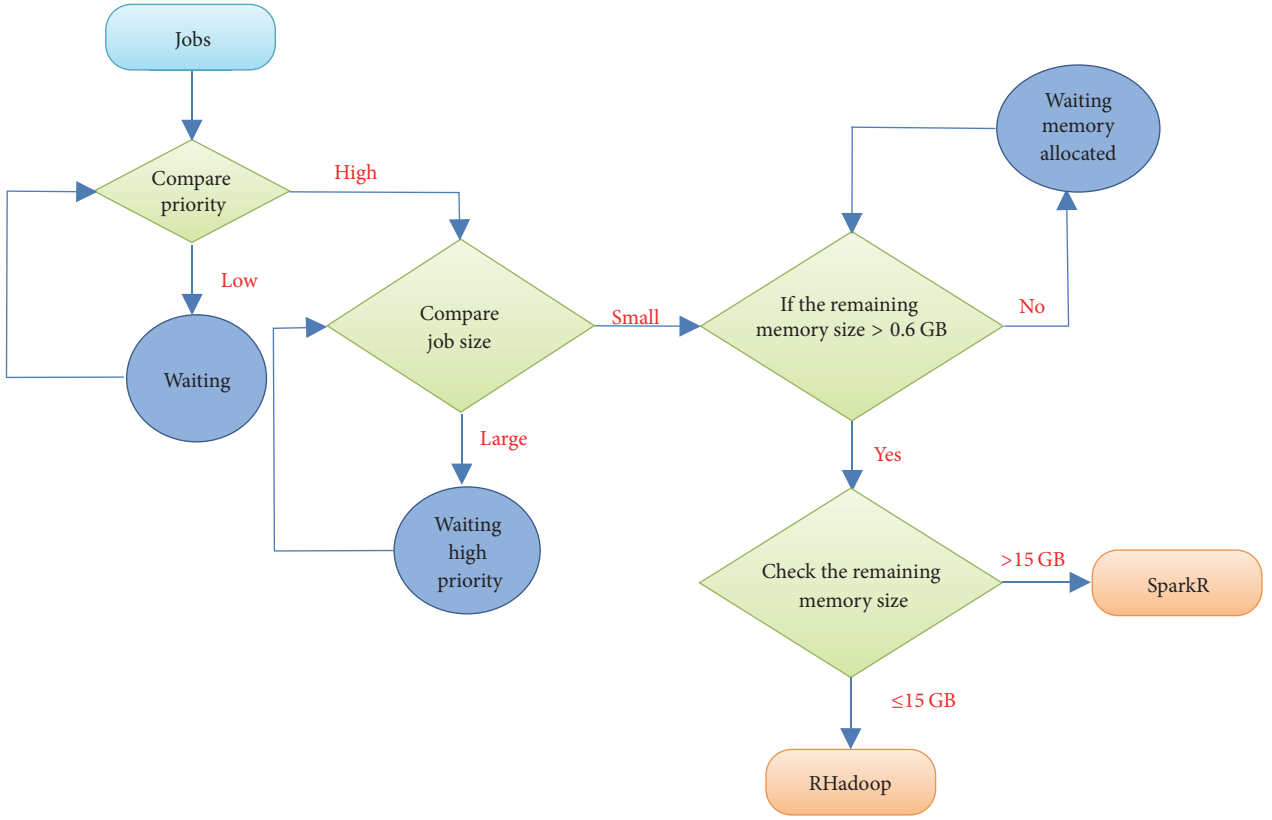


Figure 8: Job processing flow chart with MSHEFT algorithm and platform selection.

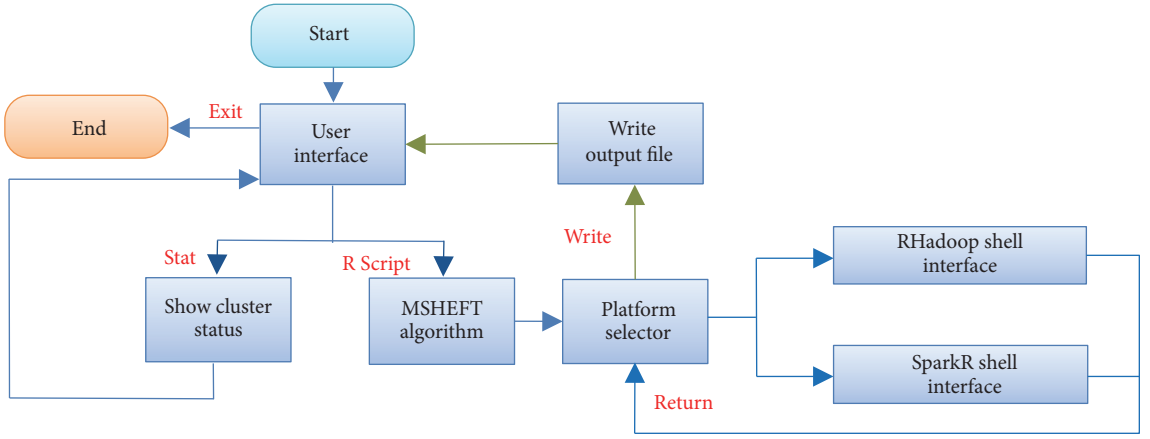


Figure 9: Execution procedure flow chart.

$\overline{\mathrm{AAT}_{s_{jk}}}$ for each test (i.e., write, read, and compute) on a specific platform, in which $\mathrm{AAT}_{s_{ijk}}$ represents the average access time of each dataset; please refer back to (1), and ω_i stands for weight for a weighted average. The following formula will evaluate the performance index (PI) [10]. Eq. (3) calculates the normalized performance index for a specific platform. Eq. (4) calculates the performance index overall for a specific platform, SF_1 is a constant value that is used here to quantify the value of performance index in the range 0–100, and W_k stands for weight for a weighted average.

$$\mathrm{AAT}_{s_{ijk}} = \frac{\mathrm{AAT}_{ijk}}{N_{ik}}, \quad \text{where } s = 1, 2, \ldots, d;\ i = 1, 2, \ldots, l;\ j = 1, 2, \ldots, m;\ k = 1, 2, \ldots, n, \tag{1}$$

$$\overline{\mathrm{AAT}_{s_{jk}}} = \sum_{i=1}^{l} \omega_i \cdot \mathrm{AAT}_{s_{ijk}}, \quad \text{where } s = 1, 2, \ldots, d;\ j = 1, 2, \ldots, m;\ k = 1, 2, \ldots, n;\ \sum_{i=1}^{l} \omega_i = 1, \tag{2}$$

$$\overline{\mathrm{PI}_{jk}} = \frac{1/\overline{\mathrm{AAT}_{s_{jk}}}}{\max_{h=1,2,\ldots m}\left(1/\overline{\mathrm{AAT}_{s_{hk}}}\right)}, \quad \text{where } j = 1, 2, \ldots, m;\ k = 1, 2, \ldots, n, \tag{3}$$

$$\mathrm{PI}_j = \left(\sum_{k=1}^{n} W_k \cdot \overline{\mathrm{PI}_{jk}}\right) \cdot \mathrm{SF}_1, \quad \text{where } j = 1, 2, \ldots, m;\ k = 1, 2, \ldots, n;\ \mathrm{SF}_1 = 10^2,\ \sum_{k=1}^{n} W_k = 1. \tag{4}$$

4. Experimental Results and Discussion

This section categories data into simulation data and actual data for test with two cases; the first case (Case 1) uses the test data generated randomly with Java programming; the second case (Case 2) adopts the actual data collected from the Internet. Proxmox Virtual Environment can be used to dynamically adjust the resource allocation to set up the experimental environments according to different memory remaining amounts, as listed in Table 2, so as to implement effect tests on various platforms.

4.1. Generated Data Set and Experimental Environment for Case 1. Case 1 tests each platform with first-come-first-serve algorithm to perform different sizes of test data, R commands having different complexity, and different priorities to all of queries so as to compare the execution time in various environments as is shown in Table 2. R commands for test are as shown in Table 3. In this experiment, there are three methods applied to test. The first approach uses first-come-first-serve algorithm (FCFS) for each single platform RHaoop or SparkR, denoted FCFS-SP. The second one is an optimized platform selection (PS) utilized to choose an appropriate platform for execution according to the remaining amount of memory in a virtual machine but it is still based on FCFS, thus denoted FCFS-PS. The third method introduced the optimization for job scheduling using MSHEFT algorithm employed to reschedule all of input queries in an ascending order in a job queue according to the smallest size of data file first. Once a job has been dequeued and launched, it based on PS will also choose an appropriate platform for execution, thereby denoted MSHEFT-PS. In short, three approaches including FCFS-SP, FCFS-PS, and MSHEFT-PS will be implemented in this paper. The test methods are shown in Table 4. With four fields, test data have been randomly generated with Java programming where the first column is the name of the only key string, the second column is random integer from 0 to 99, the third column is a random integer from 100 to 199, and the fourth column is the generated integer sequence number. Designated data size for test is shown in Table 5.

4.2. Experimental Results in Case 1. As a result, two platforms, RHadoop and SparkR, have performed for several test data sets with different priorities, data sizes, and R commands. As listed in Table 6, the proposed approach MSHEFT-PS has been implemented in the different order of jobs in a queue when comparing with the other methods. Performance comparisons of test are shown in Figures 10, 11, 12, 13, 14, and 15. The average execution time of proposed approach MSHEFT-PS is faster than the other methods, FCFS-SP and FCFS-PS. The normalized performance index and performance index are listed in Tables 7 and 8. This shows that the proposed approach outperforms the others in Case 1.

4.3. Data Collection and Experimental Environment for Case 2. Case 2 has collected the actual data sets and the designated data size for test as shown in Table 9. The concerned approaches as listed in Table 3 have applied for measuring the average execution time according to different data themes. Similarly, Table 1 has listed two test environments and R command I test is listed in Table 2 as well.

4.4. Experimental Results in Case 2. Executable job list in Case 2 is shown in Table 10. Performance comparisons of test are shown in Figures 16 and 17. The experimental results show that the average execution time of the proposed approach MSHEFT-PS is much lower than the other methods, FCFS-SP and FCFS-PS, over three different conditions. The normalized performance index and performance index are listed in Table 11. Notice that the performance of our proposed approach is superior to the others in Case 2.

4.5. Discussion. There is no specific mechanism so far to estimate job execution time for Rhadoop or SparkR. According to the report in Apache Spark website at https://spark.apache.org/, it noted that run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Technically speaking, SparkR job execution will similarly be up to 100x faster than RHadoop job execution in-memory, or up to 10x on disk as mentioned above. In this paper, the experiments show that run program for a specific job using SparkR is up to 9.7x faster than RHadoop. However, the average in SparkR job execution is nearly 3.9x faster than RHadoop job execution.

5. Conclusion

This paper found that even though the analytics platforms have the same configuration and functions, their performance still has resulted in different efficiency in different experimental conditions when applying scheduling optimization for multiple big data analytics platforms. The performance efficiency can be greatly improved by making the optimization for job scheduling, automatically detecting clustering state, and then choosing an appropriate platform for job processing. According to the experiments in Case 1 with simulation data and Case 2 with actual data, it is found that the remaining amount of memory is less and the scale of

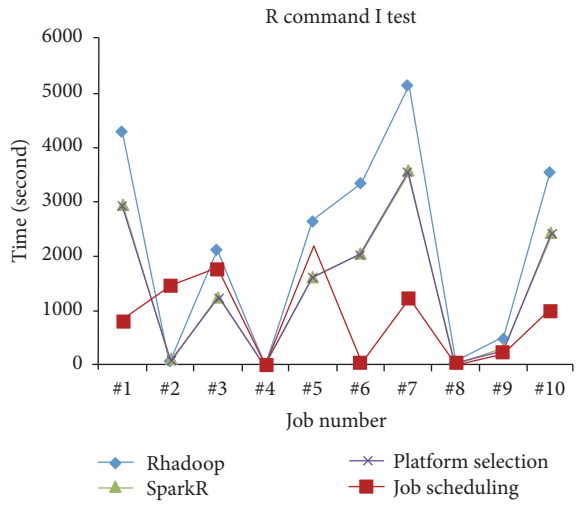


Figure 10: Execution time of R command I in test environment I of Case 1.

Table 2: Test environment.

Environment	Description
Test environment I	Adjust 10 GB memory space and give it to a virtual machine executing big data processing
Test environment II	Configure 20 GB memory space of a virtual machine executing big data processing

Table 3: R command test.

Command	Description
R command I	Only search special field
R command II	Only search special field, and add comparison conditions
R command III	Search special field, add comparison conditions, and execute the commands with while or for

Table 4: Test method.

Method	Description
FCFS-SP	Use command of enforced R to execute such platform, and then input R command
FCFS-PS	Directly input R command
MSHEFT-PS	Use command of set to set working quantity, and then input R command

data set is larger, which will much more highlight the importance of scheduling optimization and platform selection. In addition to the job scheduling using MSHEFT algorithm and optimized platform selection proposed in this paper, this system is capable of integrating new analytics platform to it by adding new big data analytics tools with related R shells to system, without any further changes in others.

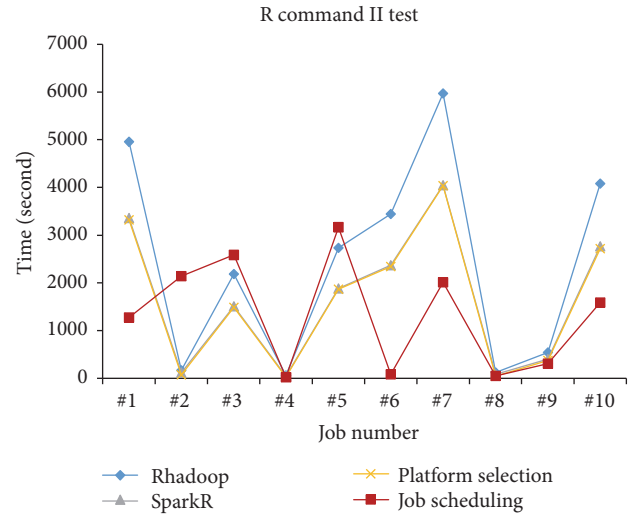


Figure 11: Execution time of R command II in test environment I of Case 1.

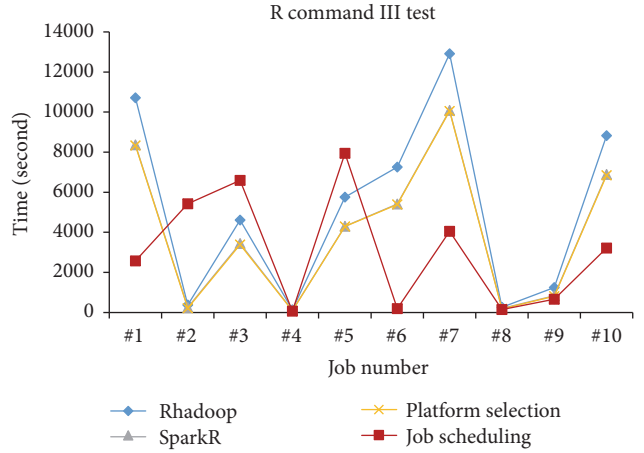


Figure 12: Execution time of R command III in test environment I of Case 1.

Table 5: Designated data size and its priority in Case 1.

Sequence	Priority	Data size	Code name
1	1	850 G	A
2	3	30 G	B
3	1	400 G	C
4	2	10 G	D
5	5	500 G	E
6	3	630 G	F
7	2	1 T	G
8	4	20 G	H
9	5	100 G	I
10	1	700 G	J

Table 6: Executable job list in Case 1.

Method	Job									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
FCFS-SP	A	B	C	D	E	F	G	H	I	J
FCFS-PS	A	B	C	D	E	F	G	H	I	J
MSHEFT-PS	C	J	A	D	G	B	F	H	I	E

Table 7: Normalized performance index in Case 1.

Operation	FCFS-SP-RHadoop	FCFS-SP-SparkR	FCFS-PS	MSHEFT-PS
R command I	0.319	0.787	0.799	1.000
R command II	0.441	0.884	0.895	1.000
R command III	0.481	0.880	0.885	1.000

Table 8: Average normalized performance index and performance index in Case 1.

Method	Average normalized performance index	Performance index
FCFS-SP-RHadoop	0.413	41.34
FCFS-SP-SparkR	0.850	85.03
FCFS-PS	0.859	85.94
MSHEFT-PS	1.000	100.00

Table 9: Designated data size and its priority in Case 2.

Sequence	Priority	Data size	Data theme	Code name
1	4	10 G	World-famous masterpiece	WC
2	1	250 G	Load of production machine: Overlaoding	OD
3	2	250 G	Load of production machine: Underloading	UD
4	3	1 T	Qualified rate of semiconductor products	YR
5	1	750 G	Correlation among temperature and people's power utilization	TE
6	4	750 G	Correlation among rainfall and people's power utilization	RE
7	1	100 G	Flight information in the airport	AP
8	5	500 G	Traffic violation/accidents	TA

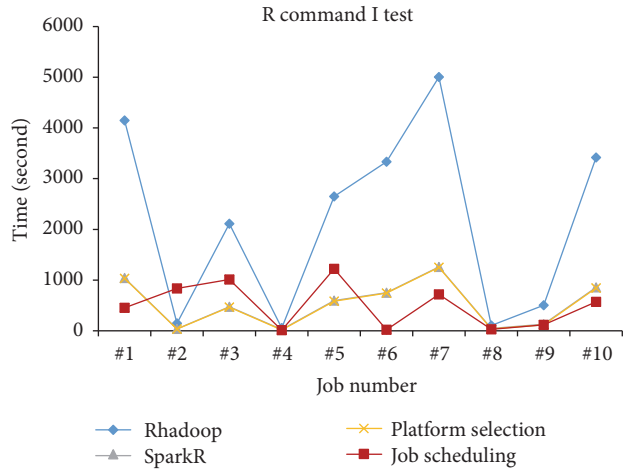


Figure 13: Execution time of R command I in test environment II of Case 1.

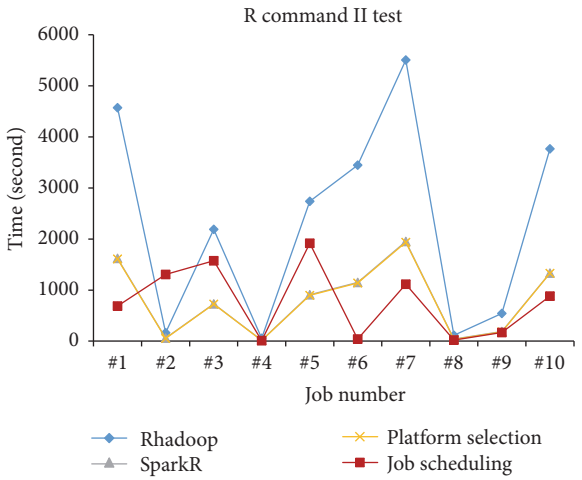


Figure 14: Execution time of R command II in test environment II of Case 1.

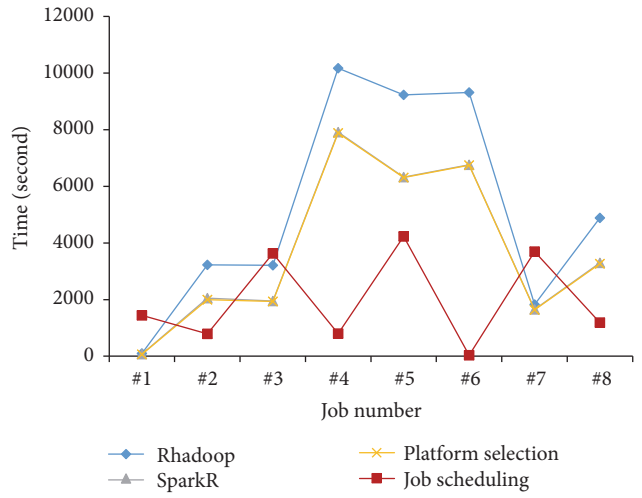


Figure 16: Execution time of experimental environment I in Case 2.

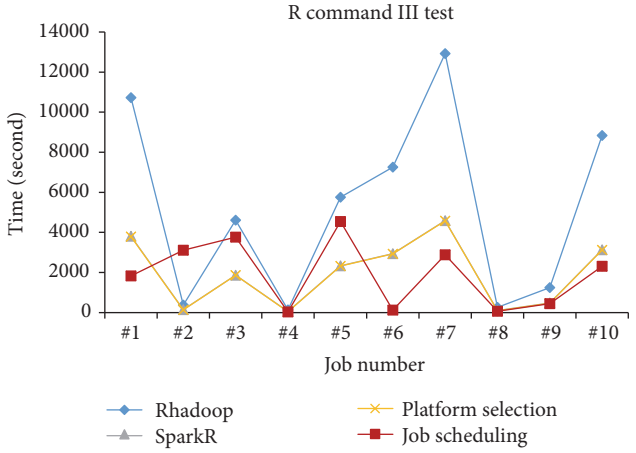


Figure 15: Execution time of R command III in test environment II of Case 1.

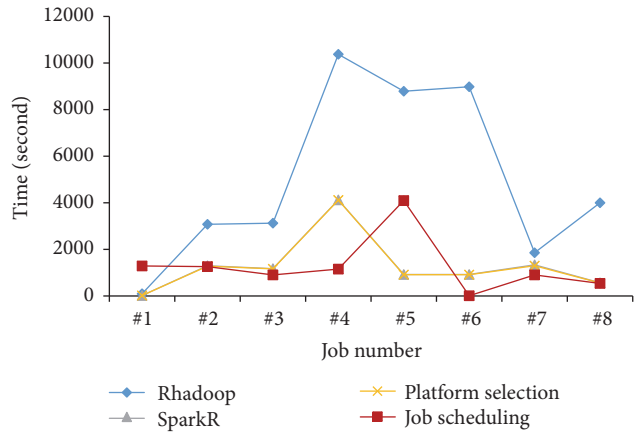


Figure 17: Execution time of experimental environment II in Case 2.

Table 10: Executable job list in Case 2.

Method	Job							
	#1	#2	#3	#4	#5	#6	#7	#8
FCFS-SP	WC	OD	UD	YR	TE	RE	AP	TA
FCFS-PS	WC	OD	UD	YR	TE	RE	AP	TA
MSHEFT-PS	AP	OD	TE	UD	YR	WC	RE	TA

Table 11: Average normalized performance index and performance index in Case 2.

Method	Average normalized performance index	Performance index
FCFS-SP-RHadoop	0.314	31.42
FCFS-SP-SparkR	0.753	75.32
FCFS-PS	0.760	76.02
MSHEFT-PS	1.000	100.00

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is fully supported by the Ministry of Science and Technology, Taiwan, under Grant nos. MOST 105-2221-E-390-013-MY3 and MOST 104-2622-E-390-006-CC3.

References

[1] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business intelligence and analytics: from big data to big impact," *MIS Quarterly: Management Information Systems*, vol. 36, no. 4, pp. 1165–1188, 2012.

[2] B. R. Chang, H.-F. Tsai, and C.-L. Guo, "High performance remote cloud data center backup using NoSQL database," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 7, no. 5, pp. 993–1005, 2016.

[3] B.-R. Chang, H.-F. Tsai, and H.-T. Hsu, "Secondary index to Big Data NoSQL Database—Incorporating solr to HBase approach," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 7, no. 1, pp. 80–89, 2016.

[4] C. D. Wickens, "Processing resources in attention dual task performance and workload assessment," 1981, Office of Naval Research Engineering Psychology Program, No. N-000-14-79-C-0658.

[5] P. Mika and G. Tummarello, "Web semantics in the clouds," *IEEE Intelligent Systems*, vol. 23, no. 5, pp. 82–87, 2008.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing*, pp. 95–101, Portland, Ore, USA, 2010.

[7] B.-R. Chang, H.-F. Tsai, Y.-C. Tsai, C.-F. Kuo, and C.-C. Chen, "Integration and optimization of multiple big data processing platforms," *Engineering Computations (Swansea, Wales)*, vol. 33, no. 6, pp. 1680–1704, 2016.

[8] S. Chaudhuri, U. Dayal, and V. Narasayya, "An overview of business intelligence technology," *Communications of the ACM*, vol. 54, no. 8, pp. 88–98, 2011.

[9] D. Harish, M. S. Anusha, and K. V. Daya Sagar, "Big data analytics using RHadoop," *International Journal of Innovative Research in Advanced Engineering*, vol. 2, no. 4, pp. 180–185, 2015.

[10] M. Adnan, M. Afzal, M. Aslam, R. Jan, and A. M. Martinez-Enriquez, "Minimizing big data problems using cloud computing based on Hadoop architecture," in *Proceedings of the 2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy), HONET-PfE 2014*, pp. 99–103, Charlotte, NC, USA, 2014.

[11] X. Yang, S. Liu, K. Feng, S. Zhou, and X.-H. Sun, "Visualization and adaptive subsetting of earth science data in HDFS: a novel data analytics strategy with hadoop and spark," in *Proceedings of the 2016 IEEE International Conferences on Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications*, pp. 89–96, Atlanta, Ga, USA, 2016.

[12] Apache Spark, 2017, https://spark.apache.org/.

[13] M. Maurya and S. Mahajan, "Performance analysis of MapReduce programs on Hadoop cluster," in *Proceedings of the 2012 World Congress on Information and Communication Technologies, WICT 2012*, pp. 505–510, Trivandrum, India, 2012.

[14] A. Kala Karun and K. Chitharanjan, "A review on hadoop—HDFS infrastructure extensions," in *Proceedings of the 2013 IEEE Conference on Information and Communication Technologies, ICT 2013*, pp. 132–137, Tamil Nadu, India, 2013.

[15] L. George, *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*, O'Reilly Media, Inc, Sebastopol, Calif, USA.

[16] A. Thusoo, J. S. Sarma, N. Jain et al., "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[17] Caesars Entertainment, 2017, https://www.cloudera.com/about-cloudera/press-center/press-releases/2015-05-05-cloudera-intel-accelerate-enterprise-hadoop-adoption-industry-partnership.html.

[18] Cerner, 2017, https://www.cloudera.com/customers/cerner.html.

[19] eharmony, 2017, http://www.eharmony.com/engineering/mapping-love-with-hadoop/#.WKCRgTt9600.

[20] J. Heinrich and B. Broeksema, "Big data visual analytics with parallel coordinates," in *Proceedings of the Big Data Visual Analytics, BDVA 2015*, Tasmania, Australia, 2015.

[21] Azure HDInsight, 2017, https://azure.microsoft.com/zh-tw/services/hdinsight/.

[22] G. Li, J. Kim, and A. Feng, "Yahoo audience expansion: migration from hadoop streaming to spark," in *Proceedings of the Spark Summit 2013*, San Francisco, Calif, USA, 2013, Yahoo, 2017, https://spark-summit.org/2013/li-yahoo-audience-expansion-migration-from-hadoop-streaming-to-spark/.

[23] Cloudera Spark Streaming, 2017, https://blog.cloudera.com/blog/2016/05/new-in-cloudera-labs-envelope-for-apache-spark-streaming/.

[24] M. S. Aravinth, M. S. Shanmugapriyaa, M. S. Sowmya, and M. E. Arun, "An efficient hadoop frameworks sqoop and ambari for big data processing," *International Journal for Innovative Research in Science and Technology*, vol. 1, no. 10, pp. 252–255, 2015.

[25] S. Hoffman, *Apache Flume: Distributed Log Collection for Hadoop*, Packt Publishing Ltd, Maharashtra, India, 2013.

[26] A. Gahlawat, "Big data analytics using R and Hadoop," *International Journal of Computational Engineering and Management*, vol. 1, no. 17, pp. 9–14, 2013.

[27] M. Zaharia, M. Chowdhury, T. Das et al., "Fast and interactive analytics over Hadoop data with Spark," *USENIX Login*, vol. 37, no. 4, pp. 45–51, 2012.

[28] S. Urbanek, M. S. Urbanek, and S. J. JDK, "Package 'rJava'," 2017, http://www.rforge.net/rJava/.

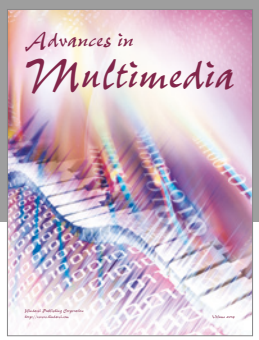
[29] S. Salian and D. G. Harisekaran, "Big data analytics predicting risk of readmissions of diabetic patients," *International Journal of Science and Research*, vol. 4, no. 4, pp. 534–538, 2015.

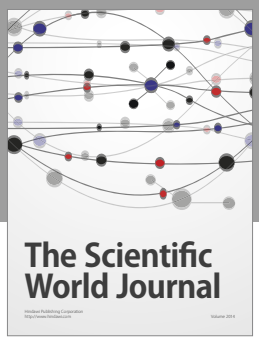
[30] B. R. Chang, H.-F. Tsai, and C.-M. Chen, "Empirical analysis of cloud-mobile computing based VVoIP with intelligent adaptation," *Journal of Internet Technology*, vol. 17, no. 5, pp. 993–1002, 2016.

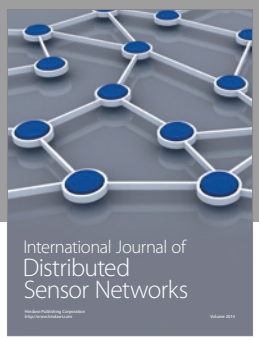
[31] Proxmox Virtual Environment, 2017, https://pve.proxmox.com/.

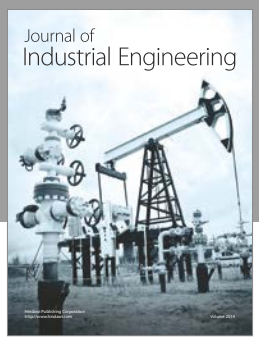
[32] B. R. Chang, H.-F. Tsai, and Y.-C. Tsai, "High-performed virtualization services for in-cloud enterprise resource planning system," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 5, no. 4, pp. 614–624, 2014.

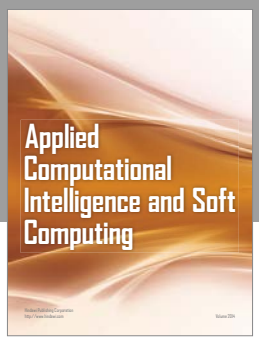
[33] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

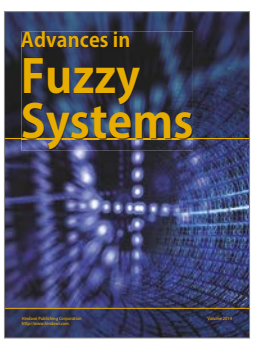


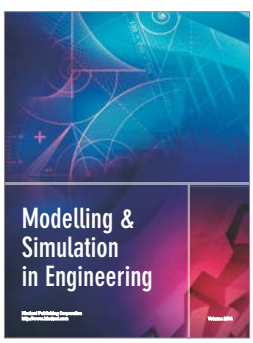


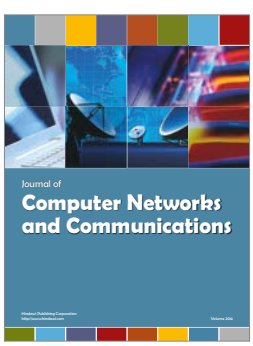












Hindawi
Submit your manuscripts at
https://www.hindawi.com