Int J Parallel Prog (2018) 46:508–527
https://doi.org/10.1007/s10766-017-0498-x

Multilevel Data Processing Using Parallel Algorithms for Analyzing Big Data in High-Performance Computing

Awais Ahmad[1] · **Anand Paul**[2] · **Sadia Din**[2] · **M. Mazhar Rathore**[2] · **Gyu Sang Choi**[1] · **Gwanggil Jeon**[3]

Received: 26 December 2016 / Accepted: 11 March 2017 / Published online: 27 March 2017

Abstract The growing gap between users and the Big Data analytics requires innovative tools that address the challenges faced by big data volume, variety, and velocity. Therefore, it becomes computationally inefficient to analyze such massive volume of data. Moreover, advancements in the field of Big Data application and data science poses additional challenges, where High-Performance Computing solution has become a key issue and has attracted attention in recent years. However, these systems are either memoryless or computational inefficient. Therefore, keeping in view the aforementioned needs, there is a requirement for a system that can efficiently analyze a stream of Big Data within their requirements. Hence, this paper presents a system

✉ Awais Ahmad
aahmad.marwat@gmail.com

✉ Gwanggil Jeon
gjeon@inu.ac.kr

Anand Paul
paul.editor@gmail.com

Sadia Din
research.2486@gmail.com

M. Mazhar Rathore
rathoremazhar@gmail.com

Gyu Sang Choi
castchoi@ynu.ac.kr

1 Department of Information and Communication Engineering, Yeungnam University, Gyeongbuk, Republic of Korea

2 School of Computer Science and Engineering, Kyungpook National University, Daegu, Republic of Korea

3 Department of Embedded Systems Engineering, Incheon National University, Incheon, Korea

architecture that enhances the working of traditional MapReduce by incorporating parallel processing algorithm. Moreover, complete four-tier architecture is also proposed that efficiently aggregate the data, eliminate unnecessary data, and analyze the data by the proposed parallel processing algorithm. The proposed system architecture both read and writes operations that enhance the efficiency of the Input/Output operation. To check the efficiency of the proposed algorithms exploited in the proposed system architecture, we have implemented our proposed system using Hadoop and MapReduce. MapReduce is supported by a parallel algorithm that efficiently processes a huge volume of data sets. The system is implemented using MapReduce tool at the top of the Hadoop parallel nodes to generate and process graphs with near real-time. Moreover, the system is evaluated in terms of efficiency by considering the system throughput and processing time. The results show that the proposed system is more scalable and efficient.

Keywords Big Data · HPC · Parallel Processing algorithm · Four-tier system architecture

1 Introduction

Recently, much focus is given to Big Data since it is one of the influential research areas in 2020 horizon. The model of the Big Data analytics relies on the data acquisition and aggregation of the enormous volume of data that supports innovation in the upcoming years. The data is considered to be big when it meets the basic requirements of traditional three v's. These three v's are volume, variety, and velocity [1]. The basis of the Big Data exploitation is to enable the existence of big data sets with a view to extracting information, which ultimately results in better business value chains. According to the latest report published by International Data Corporation (IDC), the quantity of the data will increase fourth-four time bigger in the next up-coming years. Such increase is noticed as 0.8–35 ZB [1].

Having fact that scientific applications tend toward more data-intensive due to the desires of scientific sightings. For instance, Sloan Digital Skype Survey (SDSS) and Arctic Systems Reanalysis (ASR) [2] are the two case studies that generate an extensive amount of Big Data, i.e., one hundred terabytes and twenty-three terabyte of data in an individual cycle. It is also noticed that the size of these applications will continue to increase in a near future. Therefore, various applications, such as internet of things (IoT), smart cities, and wireless sensor network can be used for data acquisition, aggregating, and analyzing data liable on the framework. Moreover, Gartner Group predicts that 26 billions of the things will be connected by 2020. Hence, it provides a great deal of opportunities for the scientific discoveries. Apparently, it poses a lot of challenges in the community of high-performance computing. In this regard, the Input/Output (I/O) systems are one of the well-known challenges and are getting attraction with notable methods proposed. The aforementioned valuations are examples of Big Data acquisition, aggregation, and analysis, and it can be dealt with the massive volume in a larger variety, aggregating data with high velocity to define value-added services and other applications.

The coupling between Big Data and High-Performance Computing (HPC) is strong [1,3,4]. Since there is no such widespread approach that supports HPC in real-time. For instance, data acquisition, data aggregation, and data analysis and their exploitation. Some of the recent trends are focused on the data acquisition and data aggregation from the data generation tiers [5], aggregation tiers [6], exploitation tier [7], and lastly, the analysis tiers [1,8]. These mentioned approaches are challenging tasks than locating, identifying, understanding or citing data [9]. In the case of large-scale data, the above-mentioned factors shall occur in a mechanized way since there is a requirement of diverse data structure and semantics that need to be expressed in a way so that the computer can understand the data. Though, to analyze data having one data sets, some traditional and simple intelligently designed databases might work. However, if the size of the data is increased than the basic requirements of memory, then the existence technique might fail to store and process parallel in a traditional computing machine. Therefore, to design a system that intelligently tackles the Big Data by using HPC will have advantages over others traditional processing tools.

Having understood from the fact that traditional Big Data analytics are not just a useful remedy to analyze a stream of data in an efficient manner. Therefore, various platforms are introduced by different vendors that are used for HPC [10]. These platforms are software only or they simply provide analysis services. Moreover, these platforms are designed for latest technologies that are used for the analytical purpose only. These technologies include web traffic, global positioning system (GPS), and IoT data. Moreover, several other platforms are also available in the market that is used for the particular application.

Among various I/O systems, a great potential has been seen in active storage systems that address the I/O bottleneck challenges for scientific application. Such potential is increased with the increase in the demand for the data. Though, the prototype of the active storage [11,12], the main focus is given to read-intensive operations. Since it provides an easy way to identify common operations, i.e., lookup, amongst several data analysis approaches, kernel analysis is predefined in libraries, known as processing kernels [12]. On the other hand, various other techniques for writing-intensive applications are not well addressed that are common in scientific areas. Since rapid advancement in the size of the output, write performance operations I/O systems becomes more important [10]. Moreover, due to incredible growth in this scientific application, several other challenges are noticed. These challenges include storing huge amount of data and memory allocation to this application, processing and analyzing these data without having intelligent technique. Several other techniques, such as prototype reduction (PR) is one of the useful remedies for class distribution. Prototype reduction splits the original data into different subsets, which can be individually addressed. After that, PR combines each moderately compact set into a global solution. In addition, torrents of event data are essential to be distributed among various databases, which required large mining algorithms needed to be distributed in the networks of the computer. However, the design of existing active storage techniques have some major drawbacks, which are (i) to process kernel design pattern is not suitable for write operations, (ii) usually, write operations require more computation power than read operations, and (iii) to process massive volume of data is sometimes hard to processes by using some traditional intelligence and processing tools.

Motivating from the aforementioned limitations discussed above, we propose a novel architecture, called multilevel data processing using parallel algorithms for analyzing big data using high-performance computing. The active Input/Output works on three levels. The first level is used for data generation. The second level is used for data storage, which is used for getting data from the data nodes and stores it in a sequential manner. Moreover, the data storage level is also comprises processing servers, in which we came up with the algorithms to enhance the efficiency of the Hadoop server. The processing server efficiently processes and analyzes Big Data in limited resources. Finally, the third level is the interpretation level, which is used for getting the processed data from the layer-two and displays it for the user.

The proposed architecture is used to analyze real-time as well as offline Big Data. The contribution of the paper is manifold, which are:

(1) The data is aggregated by our proposed architecture in larger blocks.
(2) The aggregated data is arranged in a sequential manner so that it may be useful to store in the parallel storage devices in a server.
(3) The data blocks are partitioned in equal size so that it may increase the processing efficiency.
(4) Finally, the parallel algorithms are used to process the data and perform analysis, which enhances the system efficiency to be called as high-performance computing.

The remainder of this paper is organized as follows. In Sect. 2, we give a detailed related work. In Sect. 3 presents the motivation. In Sect. 4, we presents multilevel architectural model for Big Data analytics. In we briefly explained proposed four-tier architecture design the requirements for the Big Data. In Sect. 5, we present a detailed analytical and simulation results using Hadoop and MapReduce. Finally, Sect. 6 offers a conclusion.

2 Related Work

MapReduce programming paradigm producing a large amount of datasets that is responsible for the extensive diversity of real world tasks [13]. MapReduce divides input data into small independent chunks which dealt in a parallel manner completely. The MapReduce architecture classifies the maps outputs and sends to the reduce job. Basically, the input and output of the task are kept in the file system. MapReduce is a parallel programming model, which perform three main task at the same time i.e., simplicity, load balancing, and fault tolerance. The Google File System (GFS) normally inspired from the MapReduce model gives the reliability and efficiency of data storage required for large databases applications [14].

MapReduce model motivated by functional languages. Functional languages have a map and reduce primeval exist in functional languages. Depending on the framework requirement numerous executions can be feasible in MapReduce platform. Few recently executions are existed in literature work e.g., networked machines clustering [13], shared memory multi-core system techniques [15,16], graphic processors and asymmetric multi-core processors approach [17].

Google launched one of the most famous implementation that exploit a huge number of clusters of computers, which are joint through switch Ethernet. Basically, Google MapReduce scheme reduce the cost of clusters of machines for wide range distributed applications. MapReduce approach makes simpler and easier the establishment process. It is based on real-time execution and it does not define preplan execution scheduling of the node [18].

MapReduce paradigm is able to perform a parallel execution on distributed nodes. The core purpose of MapReduce model is to clarify huge data implementation as well as low-cost cluster machines. It is also obtained fault toleration and balancing the load for each cluster to make this simpler and easier for operators. Map and Reduce are two basic entities of the MapReduce model. Google is the owner of indigenous MapReduce so it is not for public use [18]. Even though, the idea of MapReduce primeval generally simply the distributed computing system. The original MapReduce structure is very important to obtain required and efficient performance [19]. The Google's MapReduce Structure has originally distributed file system which identifies the data location and accessibility [13].

The objective of acquiring distributed data execution to processed parallel data for thousands of computing machines can be accomplished by joining the MapReduce programming model and an effectual distributed file system. By applying this approach, data can be processed on large scale i.e., terabyte and petabyte along with better system performance, reliability and more effectiveness of the system. MapReduce minimize the accessing and loading data time greater than 50% because it has data optimization efficiency and reliability [20]. MapReduce is a flexible and scalable processing tool introduced by Google. It is able to process a huge amount of low-cost computing node's data on same time [21]. Lately, MapReduce has become the center of attraction among the scientific and commercial community for its best performance [22–25].

3 Motivation

Usually, the framework of Hadoop MapReduce run over Hadoop Distributed File Server (HDFS), which has the advantage of multiple local disks on a computer node providing better data-locality [26]. Though, majority of the HPC clusters [27,28] used to follow traditional Beowulf architecture [29,30]. In such systems, the computer nodes are provided with a very light weight operating system, or sometimes with a limited capacity of local storage [31]. Simultaneously, they are all connected to a parallel file system, called as Lustre. Lustre provides an efficient and scalable data storage facility. Figure 1 delineates an example scenario of deployment of Lustre system and the operation of YARN MapReduce on modern HPC clusters.

The major drawback of this architecture has the limited capacity of local disks since they inhibit the working of MapReduce on large data sets. These inconsistencies lead toward attenuation of MapReduce running on HPC clusters. Moreover, recent studies also verify that MapReduce does not provide significant results when it combines with HPC cluster [10,32,33]. These limitations lead us toward a question whether storage system with Lustre gives us local storage capabilities that facilitate MapReduce, which gives us efficient results on HPC clusters?

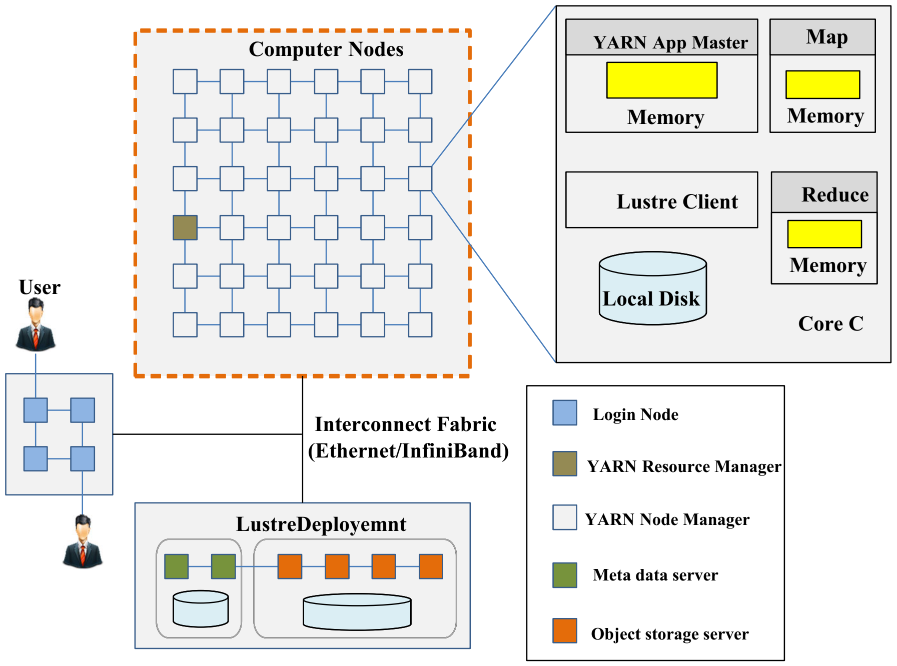

Fig. 1 YARN MapReduce running over a typical Lustre

The majority of the Lustre system installed on HPC cluster using Lustre as a local storage. This local storage is for traditional MapReduce functions where these functions can be completed in two steps, i.e., read and write operations. These operations give us high-speed data shuffles path since read and write have high throughput on Lustre systems. Though, the time required for a transmission inside Lustre depends on many factors, such as interconnection of clusters, data load, and other variations, etc. These factors, when combined together, generate an overhead on the traditional MapReduce functions [10]. Moreover, recent studies have also been proposed to enhance the function of MapReduce design [34–36] in order to speed-up the function of MapReduce. However, these systems are facing some other limitations related to the local memory, processing big data sets, dividing the job of the map and reduce function, results in storage in a real-time scenario, etc.

Motivated by these factors, the proposed multilevel architectural design for processing big data using parallel algorithms is a useful remedy to provide local memory where the map and reduce efficiently perform their functionality. Moreover, the data is divided into small chunks that can be stored in the local memory in order to perform I/O jobs more efficiently.

4 Proposed Multilevel System Design

The multilevel active storage and processing aim to extend the existing storage and processing systems. The system architecture is composed of four layers. Each layer is supported by different functionalities enables read and write operations high effectively. In this section, we will first introduce the layered architecture that supports complete system design for high-performance computing. Afterward, we will present the design and working of the designed system.

4.1 IV-Tier Layered Architecture

Based on the needs of analyzing big data, we propose an IV-Tier architecture model. The designed model assists different objects to interact with each other using the shared medium. The proposed architectural model integrate various data generated by difference application, under the same domain, i.e., social internet of things, which supports the research community to provide the generalized framework and architecture that can help the domestic users in the case of security, healthcare, elderly age people and kids, and transportation system, machine-to-machine network, wireless sensor network, and vehicular network, etc. As Fig. 1 shows that the proposed IV-Tier architectural model consists of four layers.

Tier I Data Generation handles data generation through various objects and then collecting and aggregating that data. Since a different number of objects are involved in generating the data. Therefore, an enormous number of heterogeneous data is produced with various formats, a different point of origin, and periodicity. Moreover, various data have security, privacy, and quality requirements. Also, in sensor's data, the Metadata is always greater than the actual measure. Therefore early registration and filtration technique are applied at this layer, which filters the unnecessary Metadata, as well as redundant data, is also discarded.

Tier-II This layer provides end-to-end connectivity to various devices. Moreover, data is aggregated at this point generated from various devices and arrange them in the proper format.

Tier-III Data storage and Processing Layer is the primary layer of the whole system architecture, which handles the processing of data. Since we need a real-time stream of the data and offline data analysis. Therefore, we need a third party real-time tool to combine with the processing server to provide the real-time implementation. To provide real-time implementations, Strom, Spark, VoltDb, and Hupa can be used. For instance, to be very specific in the case of data analysis, the implementation part could be achieved by using MapReduce. At this layer, the same structure of MapReduce and HDFS is used. With this system, we can also use HIVE, HBASE, and SQL supposed for managing Database (in-memory or Offline) to store historical information.

Tier-IV Service layer is the lowermost layer responsible for incorporating the third party interfaces to objects and human. This layer can be used autonomously as a single site, merged with other locations, or deployed in cloud interface. There are different other features as well. For instance, the unique global ID management is the key element in the application layer that handles identifying the object throughout the

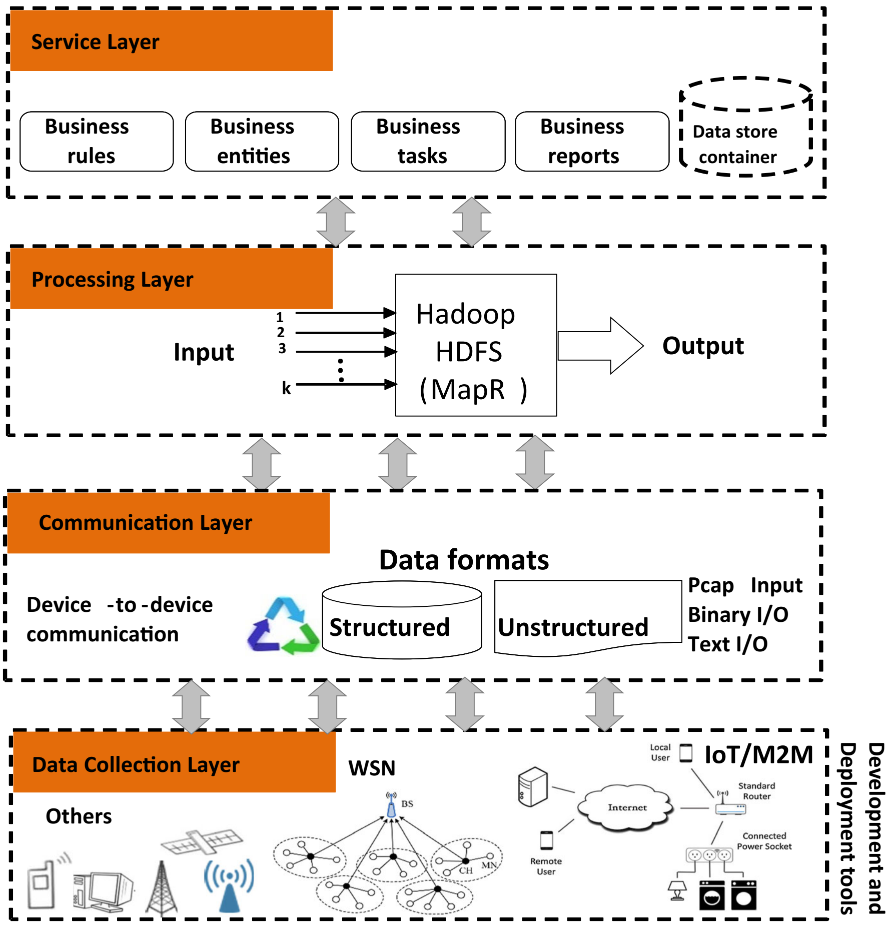

Fig. 2 Four-tier communication model

universe. Vendor control is another feature deals with the definition of the activities duly performed by different objects. The proposed architectural layers involve different objects that need intelligent power to interact with a human. For this reason, a smart algorithm is required at the application level that could efficiently and effectively interact with the human. Various tasks could be performed by these features, such as request generator, session initiating, setting up communicating rules, interact with heterogeneous objects and terminating the session (Fig. 2).

4.2 System Architecture for High-Performance Computing

Figure 3 shows the high-level system architecture for large-scale data processing services using parallel algorithms in designing HPC system. The proposed system architecture for analyzing Big Data is divided into four layers, i.e., data collection

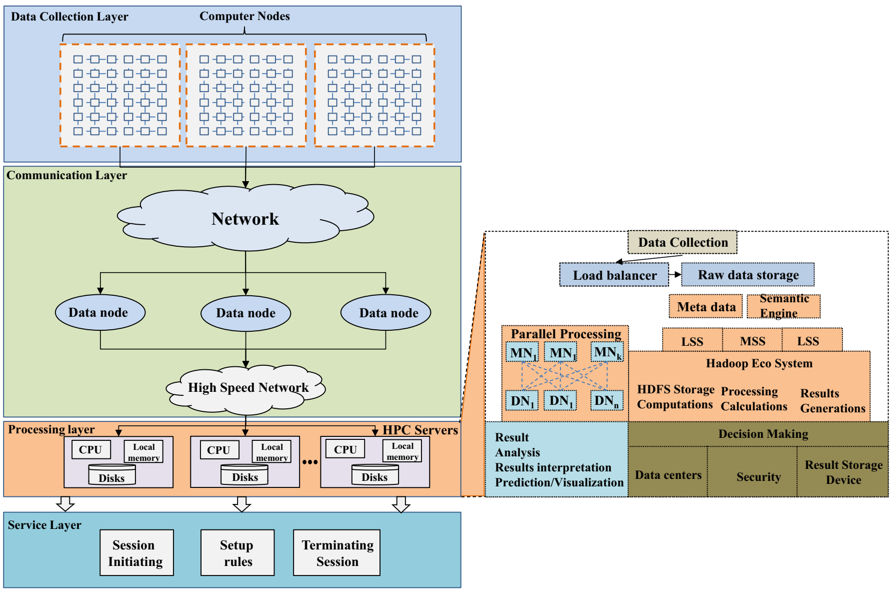

Fig. 3 Proposed Big Data Architecture using HPC system

layer, communication layer, processing layer, and service layer. Each layer is responsible for performing individual tasks. These layers are described as below

1. Data Collection Layer

This layer is responsible for data generation. Devices involved in this layer uses communication medium, such as the Internet, ZigBee, Wi-Fi, etc. to communicate with each other. The communication mediums handle collecting data from all the objects and then relay it towards the communication layer. Initially, metadata is collected whose nature heterogeneous. Moreover, this layer is also capable of finding the redundant data. For this technique, some related techniques are used to find the redundant data [1,3]. Afterward, those metadata, as well as redundant, are discarded. In the literature, the design of HPC usually tackles the entire amount of data, which uses additional system requirements. Therefore, the proposed system for HPC does not encounter processing of raw data, redundant data, or metadata. After elimination of redundant data, the useful amount of data is classified by the identifier and message type. Once the data is classified, the data is converted to machine readable form, which provides easy solution for the processing to understand it very well and process accordingly.

2. Communication Layer

This layer is responsible for transmission of data from source to the design HPC system for analysis purpose. It used high-speed Internet, GPRS, 3G, 4G/LTE, or WIMAX as a source of the medium provider. In addition, it uses Wi-Fi or Bluetooth communication technology to transfer data from source to designed server if the devices and system are kept near to each other. All the communication with the various units of the analysis system is done by Ethernet. In this layer, we exploit the nature of graph that generates or updates each time when new data is added to the system. Initially, it creates a new graph but at later stages, when it meets any incoming data, it just updates the graph by either adding a new node, new edge or updating the weights on the edge. It uses an efficient searching mechanism, which uses indexing to search particular edge to be updated when required. Graph building layer also increases the efficiency of the system by making the graph be processed on multiple parallel servers simultaneously while dividing the graph into various independent mutually exclusive parts/subgraphs. The exploitation of graphs in the system assists the HPC to process the data efficiently. It is important to note that unlike the previous system for HPC, our designed system is not only deals with the processing efficiency. However, its main task is to analyze the huge amount of data in limited resources of the Hadoop server.

3. Processing Layer

This layer is responsible for processing sub-graphs which were sent to the processing server by the intermediate layers. From the literature, it is recognized that the traditional approaches do not efficiently analyze Big Data. Therefore, a new system with novel algorithms is required that can efficiently analyze and process Big Data in real-time as well as offline. Therefore, keeping in view the aforementioned requirements, the proposed system architecture is powered by the processing layer, which acts as a core component for designing HPC system. The processing layer initially performs load balancing algorithm. Load balancing is used for distributing load to each HPC server in equal size. This equal size distribution enhances the system efficiency and all the server will process the equal amount of output, and generate an output at the same time. After load balancing, the data chunks (referred to as sub-graphs) are sent to the raw data storage device. Raw data storage device is used for storing sub-graphs in a sequential form. This unit helps HPC system if the data has some missing values; it can re-accessed from the previous layers. After storing the data, if there is metadata (after arranging data in graphs form) than it is the time to discard those data. Moreover, the semantic engine will check whether the graphs those are ready for processing is at low scale, medium scale, or large scale. The semantic engine pass-on their instruction to the Hadoop system to react accordingly (i.e., assign memory and processing power).

Now the data is ready for processing in the designed HPC server. The designed system is powered parallel algorithms, which equally and parallel process the data. Once the data is processed, it is required to the store the results in a local disk, where they can be used for future usage. In related work as stated in Sect. 2, the majority of the techniques are based on very minimum local memory. Thus, it is really hard to store the results or perform a traditional MapReduce function. In order to cope with such situation, we came up with the extension of the HPC system that provides enough local memory to store the data and result. For this purpose. The incorporation of results storage device plays an important role. In some cases using HPC systems, sometimes we do not get what we actually want, or the results are corrupted. Hence, we need to process the whole data again. Thus, reducing the computational and processing efficiency of the system. Therefore, our designed system stores the results, analyze the results and then it is displayed to the users

4. Service Layer

The service layer is provided with different features, such as session initiating, defining rules, providing security, cloud support, and others. As the data is processed by parallel algorithms is completed. Therefore, there is a need to send the results toward the service layer. As the results arrive at the service layer, the aforementioned functions started their job. On the basis of the contents, the rule is defined. These rules can be 'where to store the data when the whole process will finish, etc'. Moreover, database management features are also provided that can handle the database and the storage system. These databases are used to store various records and their relationships.

5 Implementation Results and Analysis

The main purpose of the proposed scheme is to enhance the efficiency of the HPC system based on Hadoop system. In order to validate the system, we consider some example scenarios. These example scenarios help us in implementing our proposed architecture for HPC systems. For this purpose, this section provides the details about the analyzing datasets of various sizes, the tools we use for the analysis, and the implementation of the proposed algorithm on Hadoop system.

5.1 Datasets, Tools, and Implementation Environment

In this, describe the detailed description about the data sets we used for our analysis. Please note that the datasets that are used for analysis are only to check the system performance of the proposed HPC system based on Hadoop. The datasets are taken from the European Space Agency (ESA) [37]. Moreover, other datasets are related to the healthcare system, which comprises Glucose level of the patients, ECG, temperature, and various other activities [1].

ESA gets their datasets from two satellites, as shown in Table 1. These satellites are mounted 800 KM above the surface of the earth [38]. In these datasets, different products are analyzed, which are Sea, Ice, and River, as shown in Fig. 4. Moreover, ESA is monitoring various countries the globe, i.e., S. Africa, Mauritania, and the USA, etc. In Fig. 10, product 10 covers the land, ice, sea from Canada. Product 7 covers the sea and land between Spain and Morocco. Finally, product 9 and product 1 is from USA and Vietnam. These in-depth details about the datasets are highly recommendable to test the proposed HPC system using Hadoop environment. We developed and test the proposed algorithm to extract the features of the river using the divide and combine mechanism on corei5 3.20 GHz $\times$ 4, UBUNTU 14.04 local machine with Hadoop single node setup having 4 GB RAM and Gallium 0.4 on AMD OLAND graphics.

5.2 Proposed Algorithm for HPC System Using Parallel Algorithms

In the processing layer, analyze function is used to break down the data into smaller chunks so that interested feature shall be detected in an efficient way. Initially, data is arranged in a fixed size cell, called block, and in each block, there are various statistical

Table 1 Parameters and their description

Parameters	Descriptions
∂_{b_size}	Block size threshold
Set_{rivers}	Set of concerned point (CP) detected. I.e., $R_1, R_2, R_3, \ldots R_n$ and each R_i is also a set of ordered 3- tuples that contain the pixels information of the river $Ri = \{(x, y, value) / (x, y) \in R_i\}$
B_1, B_2	Blocks obtained by dividing the matrix
Width_M	Width of image matrix
Height_M	Height of image matrix
$\overline{X}$_B, S.D_B	Mean and Standard Deviation of pixel values of block and calculated as: $\overline{X}\_B = \frac{\sum Pixel\ values\ in\ block\ B}{no.\ of\ pixels}\ S.D_B = \sqrt{\frac{\sum(Pixel\ value\ i - \overline{X}_{Bi})^2}{no.\ of\ pixels}}$
ED	The Euclidian distance between two pixels. $\mathrm{ED(P1, P2)} = \sqrt{(P_1x - P_2x)^2 + (P_1y - P_2y)^2}$ Where p_1: $\mathrm{pixel_1 position} = (P_1x, P_1y)$, P_2 : $\mathrm{pixel_2 position} = (P_2x, P_2y)$
$\partial_{Min\_RB\_SD}$	Minimum S.D threshold set for the block that has a concerned point
∂_{Mean_Diff}	Threshold set for mean absolute difference between $\overline{X}$_B and Mean of concerned point pixel values
NP_RDS	Threshold set for detecting minimum number of pixel in a concerned set of single block
Threshold set for detecting minimum number of pixel in a concerned point set of single block	Threshold set for detecting minimum number of pixel in a concerned point set of single block

parameters are associated with it. Hence, if the data block does not meet the threshold value, then it means that the block does not contain any of the interested features. Moreover, if the interested parameter is found in the data block, then the anomalies as well as false-positive are minimized by exploiting mean difference comparison. The difference is calculated between mean values of the interested region and the whole block by ($\left|\overline{X}_{\mathrm{RDC}} - \overline{X}_{\mathrm{B}}\right|$). Moreover, the pixel value for each interested region is denoted by ((NP_RDC). These parameters are mapped with the different threshold in order to minimize false-positive in the data sets, as shown in Fig. 5.

The combination of results is made in the processing layer of the multiple processors. Since the data is divided for each parallel processing, performing the above-mentioned algorithm at each level. Now, we need to combine the results, which are being generated by each processor. For this reason, recursive operations are made on each consecutive data blocks. For instance, if one set remains empty than another set of the data block can be returned. Apparently, the proposed algorithm checks the Euclidean Distance (ED) between the interested rejoins of different data sets. Such difference is made to know about the continuous features. In order to examine the continuous features, our proposed HPC system based Hadoop performed significant

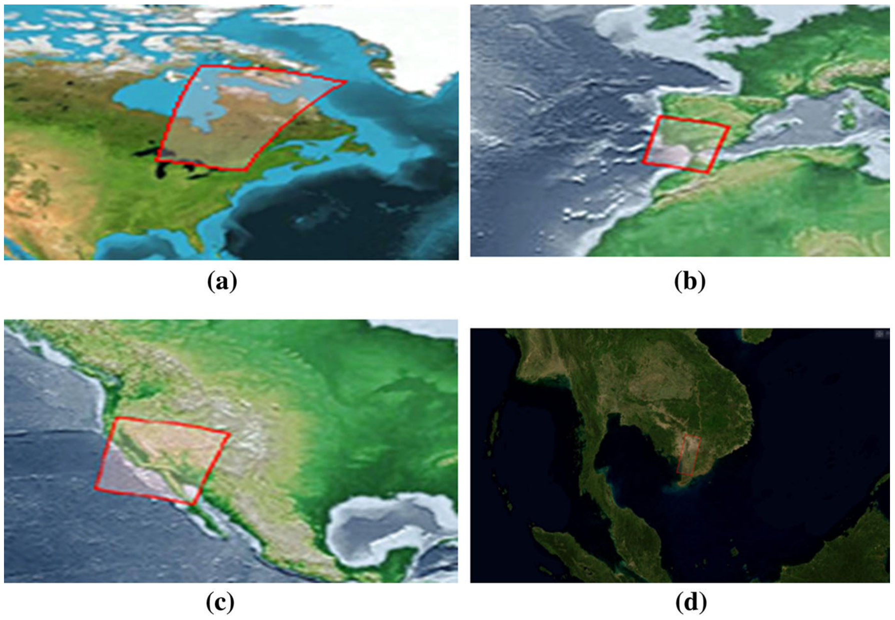

Fig. 4 Datasets locations. **a** Product 10, **b** Product 7, **c** Product 9 and **d** Product 1

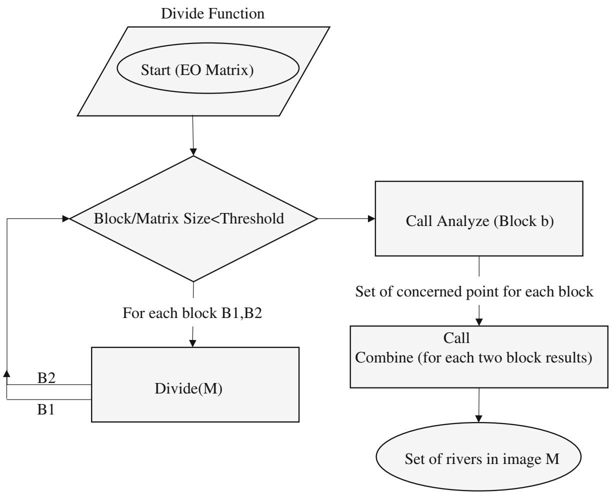

Fig. 5 Divide function flows chart

calculations, and hence enhancing the system efficiency. After calculating ED, bother interested rejoins at each data block are merged together, as shown in Figs. 6 and 7.

1. Proposed Algorithm

The proposed algorithm for the proposed HPC system based on Hadoop server uses recursive mechanisms in order to achieve division of the data, analysis of the data, and

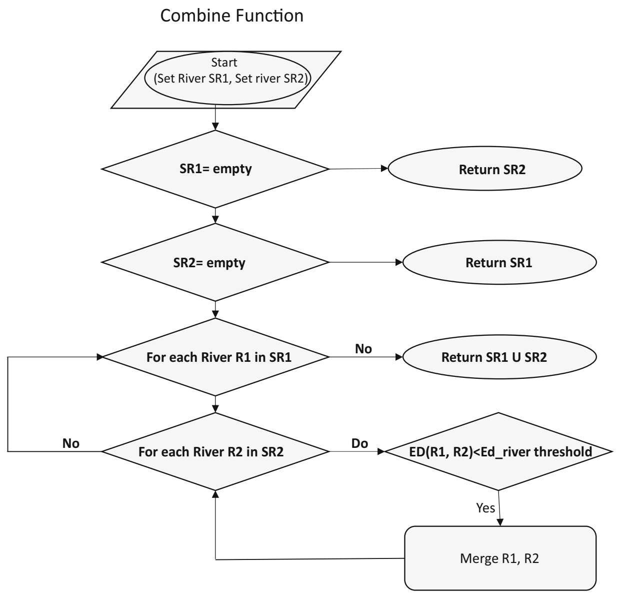

Fig. 6 Combine function flowchart

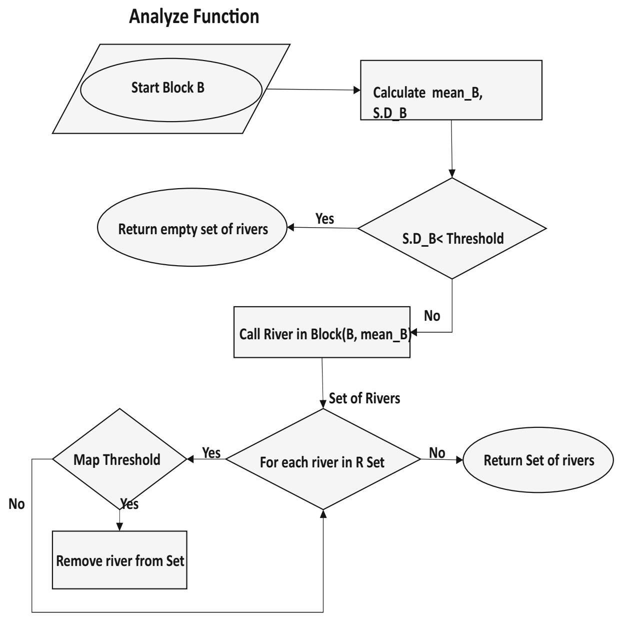

Fig. 7 Analyze function flowchart

combine the data. In the algorithm, the first statement is the division function (i.e., if-statement). This statement checks the incoming data chunks in the processing server. In the given algorithm, if the size of the data blocks is less than the defined threshold than the division process will stop and analysis function will start. Apparently, each data block is equally divided (horizontal and vertical) depending on the width and height of the data block. Moreover, the division function is called again in order to check the block size. This will happen when we have large data sets, and we need to analyze the data more efficiently. After division into the certain threshold, the division process stops. If the algorithm detects the region of interest, then it compares and analyzes the pixel value information and the entire data block with the mean values. Moreover, a REPTree method is exploited to divide the data in a fixed size up to certain extent. And finally, the combine function is used to collect and combine the results generated by each parallel server. The proposed parallel algorithms for HPC are described in Algorithm 1.

5.3 Results and Discussion

In order to test and validate our proposed HPC algorithm, the datasets mentioned above are implemented using Java iterations and the Hadoop with proposed algorithm for HPC system. The proposed algorithm using enhanced MapReduce in the light of the proposed algorithm is more efficient that the simple Java iteration implementation as shown in Fig. 8. in this figure, the graph shows average processing time to process 1MB of data using proposed algorithm as well as Java iterations. From the figure, it is clearly seen that the proposed algorithm for HPC system require approximate half of the second to process 1 MB of data. Apparently, Java iterations require more time to process the same amount of data. In addition, ASA–APS requires more time for both cases since the size of the data is too much. Moreover, products of the ASA–WSM are processed quite efficiently since the size of the data is very less. It is concluded from the figure that size of the data plays an important role in any system. Likewise, if the size increases, the performance of Java iteration drastically reduces, whereas, the performance of the proposed HPC system is giving us better results.

Moreover, the proposed algorithm for HPC system is compared with the size of the data. As shown in Fig. 9, if the size of the data is kept smaller for processing (for both Java and proposed algorithm), in this case, Java implementation generate quite good results than proposed scheme. On the other hand, if the size of the data increases, the parallel processing nature of the proposed scheme efficiently process and analyze data with the HPC systems. Since the parallel processors are waiting for the data. When each server receives data, then the overall performance of the proposed scheme significantly increases. Apparently, the Java iteration performs very poorly when the data size is increased.

Moreover, we also evaluate the throughput of the proposed system by increasing the size of the data sets. As shown in Fig. 10, the throughput is directly proportional to the size of the data sets. When there is an increase in the size of the data sets, the throughput also increases, hence increase the system sufficiency.

In order to test and validate with various other datasets, we measured the processing time as good throughput on healthcare datasets as shown in Figs. 11 and 12. In the figure, the proposed scheme with proposed parallel algorithm for HPC system takes few seconds to process GBs of data. To be more specific, it takes seventy seconds

Algorithm I: Parallel Processing Algorithm for HPC System

```
Divide (image_matrix M) {
1: If (size (M) <=$\boldsymbol{\partial}_{B\_size}$) {
    Set_ CP = Analyze (M);
    Return Set_ CP;     } //end of if
2: If (Width_M<Height_M) {   // divide m into two parts
vertically
 B1= M [0- Width_M /2] [Height_M]; //first half of M
 B2=M [Width_M /2- Width_M] [Height_M];//2nd half of M}
3: Else {
  B1= M [width_M] [0-Height_M/2]; //Upper half of M
  B2=M [width_M] [Height_M/2- Height_M]; // Lower half of
M} // end of if else
4: Set_ CP 1=Divide (B1);
5: Set_ CP 2=Divide (B2);
6: Conquer (Set_ CP1, Set_ CP2); //combining blocks and
results of blocks. }//end of Divide
Analyze (Image_Matrix_Block B) {
1:  Calculate$\bar{X}$_B, S.D_B;
2:  If (S.D_B <$\boldsymbol{\partial}_{Min\_RB\_SD}$) {// Block does not have any river.
     Set_ CPDataClass_Set= Φ
     Return Set_ CPDataClass_Set ;} // end of if
3:  Set_ CPDataClass_Set = CP_in_BlocK (B, $\bar{\boldsymbol{X}}$_B);
4:  For each (CPDataClass RDC: Set_ CPDataClass_Set) {
      If (($|\bar{X}$_RDC $-$ $\bar{X}$_B$|<\boldsymbol{\partial}_{Mean\_diff}$) || (NP_RDC < $\boldsymbol{\partial}_{\mathrm{NP\_RDC}}$)
   Remove RDC from Set_ CPDataClass_Set ;}
5:ReturnSet_ CPDataClass_Set; // Return set of CP detected.}
CP_in_Block (Matrix block B, Double $\bar{\boldsymbol{X}}$_B) {
1: Define Set_R= Φ;
2: Divide B into sub_blocks S_B of size 10 x 10;
3:  For each (S_B) Do {
    Calculate $\bar{X}$_SB, SD_SB, |$\bar{\boldsymbol{X}}$_B-$\bar{X}$_SB|;
    If (REPTree ($\bar{\boldsymbol{X}}$_SB, SD_SB, |$\bar{\boldsymbol{X}}$_B-$\bar{X}$_SB|) == river) {
      Set_R= Set_R U S_B ;}}
4:  ∀ SBi, SBj ∈ Set_R where (i≠j), if (ED (SBi, SBj) <=2) the
merger SBi, SBj
5: $return$ Set_R}
Combine (Set Set_ CP1, Set set_Rivers2) {
1: If (Set_ CP1== Φ) then return Set_ CP2;
2: If (Set_ CP2== Φ) then return Set_ CP1; //if either set is
empty then no need to combine.
3:   For each (CP R1: Set_ CP 1)
       For each (CP R2: Set_ CP 2) {
          If(ED (R1, R2) <$\boldsymbol{\partial}_{Ed\_\mathrm{CP}}$) then R1 + R2;
//Combine R1 and R2, remove individual entries of R1 and R2
            }//end of for each loop
4: Return (Set_ CP1 U Set_ CP2);
Combine CP Set1 and Set_ CP 2
}
```

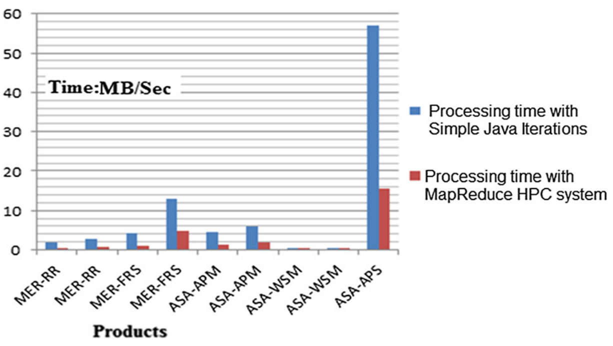

Fig. 8 Processing time is taken by our algorithm for various products

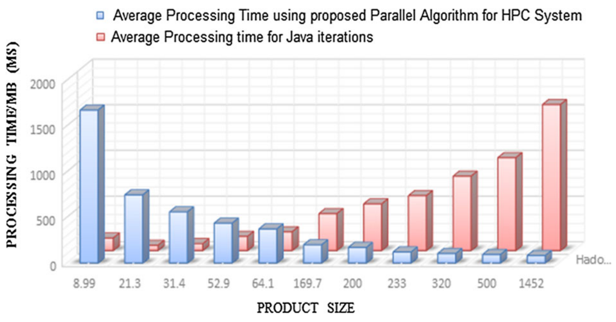

Fig. 9 Processing time is taken by the proposed system using simple Java and Hadoop

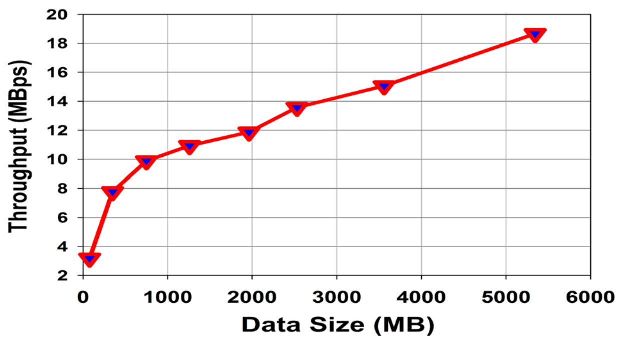

Fig. 10 Throughput of the proposed system

to process 2 GB data on a single node of the parallel processor. Moreover, if we increase the size of the datasets that due to the nature of the proposed parallel and distributed system of the parallel processors, the throughput is maximized. Therefore, it is concluded from these results that the proposed system with parallel processors for HPC system give us very efficient results than ordinary simple processing tools.

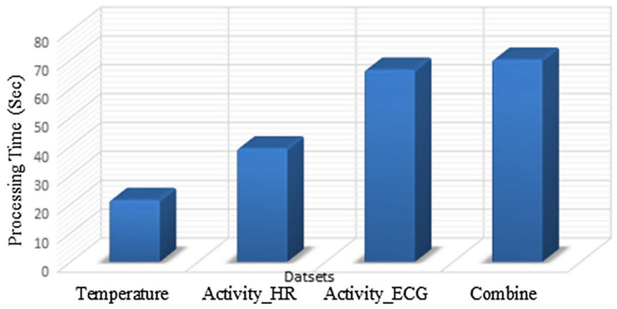

Fig. 11 Processing time of healthcare datasets

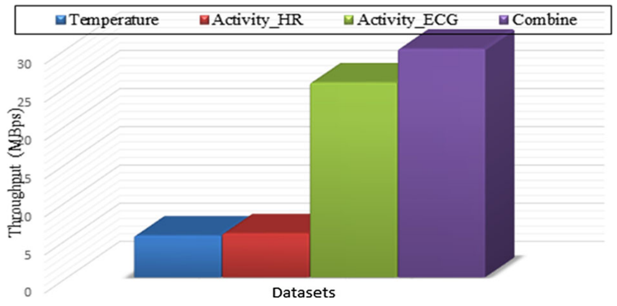

Fig. 12 Throughput of our proposed scheme

6 Conclusion

Undeniably, scientific discoveries and latest innovations can benefit significantly from huge volume aggregated data and simulated data. Moreover, data scientists can gain insights and apprehend the singularities behind the data more efficiently. However, it is based on the assumptions that the designed HPC system can deliver effective and efficient I/O system. In prior research, various works has been done on the I/O systems to make it more effective, enhance the data storage and computational capabilities, which give a significant amount of results based on the data processing. However, still, the prior systems are lacking such capabilities. Therefore, in this paper, the proposed scheme based on a parallel algorithm that effectively enhances the computational capabilities of HPC servers. The proposed scheme is based on the four layers architectural model then aggregated the data, remove erroneous or redundant data, which is useful for the enhancement of computational capabilities. Moreover, an architecture that analyzes Big Data is also proposed based on the parallel algorithm that is exploited on Hadoop server giving high-performance computing. The whole system is implemented using enhanced MapReduce with the additional feature of parallel processing algorithms to process large graphs and MapReduce to process other data with Hadoop ecosystem to achieve the efficiency and real-time processing. The results proved that the use of the parallel algorithm with MapReduce and Hadoop ecosystem dramatically increase the efficiency of the whole system.

Acknowledgements This work is supported by BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (21A20131600005) and NRF Grant funded by the Korean Government (NRF-2015R1D1A1A01058171).

References

1. Ahmad, A., Paul, A., Rathore, M.M.: An efficient divide-and-conquer approach for big data analytics in machine-to-machine communication. Neurocomputing **174**, 439–453 (2016)
2. NOAA. Overview of Current Atmospheric Reanalysis. http://reanalyses.org/atmosphere/overview-current-reanalyses (2016)
3. Ahmad, A., Paul, A., Rathore, M., Chang, H.: An efficient multidimensional big data fusion approach in machine-to-machine communication. ACM Trans. Embed. Comput. Syst. (TECS) **15**(2), 39 (2016)
4. Rathore, M.M., Ullah, A.P., Ahmad, A., Chen, B.-W., Huang, B., Ji, W.: Real-time big data analytical architecture for remote sensing application. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **8**(10), 4610–4621 (2015)
5. Haderer, N., Romain, R., Seinturier, L.: Dynamic deployment of sensing experiments in the wild using smartphones. In: IFIP International Conference on Distributed Applications and Interoperable Systems, pp. 43–56. Springer, Berlin, Heidelberg (2013)
6. Mosser, S., Fleurey, F., Morin, B., Chauvel, F., Solberg, A., Goutier, I.: Sensapp as a reference platform to support cloud experiments: from the internet of things to the internet of services. In: 2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 400–406. IEEE (2012)
7. Mosser, S., Logre, I., Ferry, N., Collet, P.: From sensors to visualization dashboards: need for language composition. In: Globalization of Modeling Languages workshop (GeMOC'13) (2013)
8. Awais, A., Paul, A., Rathore, M.M., Chang, H.: Smart cyber society: integration of capillary devices with high usability based on cyber–physical system. Future Gen. Comput. Syst. **56**, 493–503 (2016)
9. Labrinidis, A., Jagadish, H.V.: Challenges and opportunities with big data. Proc. VLDB Endow. **5**(12), 2032–2033 (2012)
10. Chen, C., Lang, M., Chen, Y.: Multilevel active storage for big data applications in high performance computing. In: 2013 IEEE International Conference on Big Data, pp. 169–174. IEEE (2013)
11. Felix, E.J., Fox, K., Regimbal, K., Nieplocha, J.: Active storage processing in a parallel file system. In: Proceedings of the 6th LCI International Conference on Linux Clusters: The HPC Revolution, p. 85 (2006)
12. Thakur, R., Gropp, W., Lusk, E.: Data sieving and collective I/O in ROMIO. In: The Seventh Symposium on the Frontiers of Massively Parallel Computation, 1999. Frontiers' 99, pp. 182–189. IEEE (1999)
13. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
14. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. ACM SIGOPS Oper. Syst. Rev. **37**(5), 29–43 (2003)
15. Yoo, R.M., Romano, A., Kozyrakis, C.: Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system. In: IEEE International Symposium on Workload Characterization, 2009. IISWC 2009, pp. 198–207. IEEE (2009)
16. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating mapreduce for multi-core and multiprocessor systems. In: 2007 IEEE 13th International Symposium on High Performance Computer Architecture, pp. 13–24. IEEE (2007)
17. Rafique, M.M., Rose, B., Butt, A.R., Nikolopoulos, D.S.: Supporting MapReduce on large-scale asymmetric multi-core clusters. ACM SIGOPS Oper. Syst. Rev. **43**(2), 25–34 (2009)
18. Lee, K.H., Lee, Y.J., Choi, H., Chung, Y.D., Moon, B.: Parallel data processing with MapReduce: a survey. AcM sIGMoD Rec. **40**(4), 11–20 (2012)
19. Shim, K.: MapReduce algorithms for big data analysis. Proc. VLDB Endow. **5**(12), 2016–2017 (2012)
20. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: Planet: massively parallel learning of tree ensembles with mapreduce. Proc. VLDB Endow. **2**(2), 1426–1437 (2009)
21. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Commun. ACM **53**(1), 72–77 (2010)

22. Ekanayake, J., Pallickara, S., Fox, G.: Mapreduce for data intensive scientific analyses. In: IEEE Fourth International Conference on eScience, 2008. eScience'08, pp. 277–284. IEEE (2008)
23. Rathore, M.M., Ahmad, A., Paul, A., Rho, S.: Exploiting encrypted and tunneled multimedia calls in high-speed big data environment. Multimed. Tools Appl. 1–26 (2017)
24. Paul, A., Ahmad, A., Rathore, M.M., Jabbar, S.: Smartbuddy: defining human behaviors using big data analytics in social internet of things. IEEE Wirel. Commun. **23**(5), 68–74 (2016)
25. Rathore, M.M., Paul, A., Ahmad, A., Jeon, G.: IoT-based big data: from smart city towards next generation super city planning. Int. J. Semant. Web Inf. Syst. **13**(1), 28–47 (2017)
26. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
27. Stampede at TACC. http://www.tacc.utexas.edu/resources/hpc/stampede
28. Gordon at San Diego Supercomputer Center. http://www.sdsc.edu/us/resources/gordon/
29. Gropp, W., Lusk, E., Sterling, T.: Enabling Technologies in Beowulf Cluster Computing with Linux, 2nd edn, vol. 3. The MIT Press, Cambridge, MA, London, England, p. 14 (2003)
30. Sterling, T.L., Salmon, J., Becker, D.J., Savarese, D.F.: How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters. MIT Press, Cambridge, MA (1999)
31. Engelmann, C., Ong, H., Scott, S.L.: Middleware in modern high performance computing system architectures. In: International Conference on Computational Science, pp. 784–791. Springer, Berlin, Heidelberg (2007)
32. Castain, R.H., Kulkarni, O.: MapReduce and Lustre: Running Hadoop in a High Performance Computing Environment. https://intel.activeevents.com/sf13/connect/sessionDetail.ww?SESSIONID=1141
33. Wasi-ur Rahman, Md., Lu, X., Islam, N.S., Rajachandrasekar, R., Panda, D.K.: MapReduce over Lustre: Can RDMA-Based Approach Benefit? In: tEuropean Conference on Parallel Processing, pp. 644–655. Springer, Berlin (2014)
34. Wasi-ur-Rahman, Md., Islam, N.S., Lu, X., Jose, J., Subramoni, H., Wang, H., Panda, D.K.: High-performance RDMA-based design of Hadoop MapReduce over InfiniBand. In: 2013 IEEE 27th International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), pp. 1908–1917. IEEE (2013)
35. Wasi-ur Rahman, Md., Lu, X., Islam, N.S., Panda, D.K.: HOMR: a hybrid approach to exploit maximum overlapping in MapReduce over high performance interconnects. In: Proceedings of the 28th ACM international conference on Supercomputing, pp. 33–42. ACM (2014)
36. Lu, X., Islam, N.S., Wasi-Ur-Rahman, Md., Jose, J., Subramoni, H., Wang, H., Panda, D.K.: High-performance design of Hadoop RPC with RDMA over InfiniBand. In: 2013 42nd International Conference on Parallel Processing, pp. 641–650. IEEE (2013). doi:10.1109/ICPP.2013.78
37. Available online: 14/10/2014, 2312. https://earth.esa.int/
38. ESA: ENVISAT Altimetry Level 2 User Manual V1.4 2011. [Available online: 15/10/2014, 0333] https://earth.esa.int/pub/ESA_DOC/ENVISAT/RA2-MWR/PH_light_1rev4_ESA.pdf