An efficient divide-and-conquer approach for big data analytics in machine-to-machine communication

Awais Ahmad, Anand Paul *, M. Mazhar Rathore

The School of Computer Science and Engineering, Kyungpook National University, Daegu 702-701, Republic of Korea

A R T I C L E    I N F O

Article history:
Received 1 December 2014
Received in revised form 24 March 2015
Accepted 14 April 2015

Keywords:
M2M
Big Data
Divide-and-conquer
Data fusion domain

A B S T R A C T

Machine-to-Machine (M2M) communication relies on the physical objects (e.g., satellites, sensors, and so forth) interconnected with each other, creating mesh of machines producing massive volume of data about large geographical area (e.g., living and non-living environment). Thus, the M2M is an ideal example of Big Data. On the contrary, the M2M platforms that handle Big Data might perform poorly or not according to the goals of their operator (in term of cost, database utilization, data quality, processing and computational efficiency, analysis and feature extraction applications). Therefore, to address the aforementioned needs, we propose a new effective, memory and processing efficient system architecture for Big Data in M2M, which, unlike other previous proposals, does not require whole set of data to be processed (including raw data sets), and to be kept in the main memory. Our designed system architecture exploits divide-and-conquer approach and data block-wise vertical representation of the data-base follows a particular petitioner strategy, which formalizes the problem of feature extraction applications. The architecture goes from physical objects to the processing servers, where Big Data set is first transformed into a several data blocks that can be quickly processed, then it classifies and reorganizes these data blocks from the same source. In addition, the data blocks are aggregated in a sequential manner based on a machine ID, and equally partitions the data using fusion algorithm. Finally, the results are stored in a server that helps the users in making decision. The feasibility and efficiency of the proposed system architecture are implemented on Hadoop single node setup on UBUNTU 14.04 LTS core™i5 machine with 3.2 GHz processor and 4 GB memory. The results show that the proposed system architecture efficiently extract various features (such as River) from the massive volume of satellite data.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Big Data is one of the central and influential research challenges for the 2020 horizon. The archetype relies on the acquisition and aggregation of the massive volume of data to support innovation in the upcoming years. The data is considered as big when it meets the requirements of "four V's", such as Volume, Variety, Velocity, and Value. The groundwork of Big Data exploitation is to empower the existence data sets to extract new information, helps in enrichment of business values chains. According to the IDC group, the quantity of world data will be 44 times bigger in the next few years (such as 0.8–35 zettabytes). Therefore, in this context, the machine-to-machine archetype relies on the world of interconnected object [1], which can be used for acquisition, aggregation and analyzing the data depending on context. In these days', vehicles, smart phones, buildings, satellites, sensors, etc. collects various information about physical environment (i.e., large geographic area), where majority of them are generating zettabytes of the sensed data. For instance, the Gartner group predicts that up to 26 billion of the machines will be connected to the internet by 2020. Furthermore, Intechno Consulting estimates that up to 180 billion Euros will be generated worldwide. The aforementioned estimations are the examples of the Big Data aggregation and analysis as it can be dealt with the large Volume in large Variety, aggregated data with a high Velocity to define application with added-Value [2].

The coupling between M2M and the Big Data communities is strong [3–5]. As such, there is no widespread approach that supports data acquisition, data aggregation, and data analysis from numerous objects (such as, satellites, sensors, and so forth), and their exploitations. Based on the aforementioned needs, recent research efforts are focused on the data acquisition from the data generation tiers [6], the aggregation tiers [7], or the exploitation [8], and lastly, the data analysis tiers. Such approaches are somehow challenging task in
The incredible growth in the data also posing new challenges, such as, how to aggregate massive volume of data? How to store such data in a limited amount of memory allocated for the particular task? Moreover, how to process and analyze these data when there is no such intelligent algorithm is available? Moreover, large-scale data cannot be tackled by standard reduction techniques since their runtime becomes impractical. Several other approaches have been developed that helps in enabling data reduction techniques, which deal with this problem. In the case of Prototype Reduction (PR), the data level partitioning is based on a distributed partitioning model that sustains the class distribution. Such type of reduction splits the original data into various subsets that could be individually addressed. Afterward, it combines each partially reduced set into a global solution. Furthermore, torrents of event data are required to be distributed over various databases, and large process mining problems need to be distributed over a network of computer. Several other approaches could be found in the literature [2,4,25,27]. However, the generic divide-and-conquer approach based on fusion technique could be the optimal solution for the said challenges.

Hence, in a nutshell, the following two main problems appear when we increase the data set size during analysis.

- The existing Big Data architectures are not capable of processing and analyzing a large amount of data, i.e., the data that is generated by the various remote sensory satellites.
- Continuous feature extraction, such as rivers or highways detection from remote sensory Big Data is a challenging issue. Such scheme requires efficient algorithms in handling large scale earth observatory datasets on a limited timescale.

Therefore, in this work, we propose a system architecture designed for analyzing Big Data in M2M using a divide-and-conquer mechanism that welcomes real-time and offline data. The proposed system architecture handles the drawbacks mentioned above. To do so, various machine are used for Earth Observatory System (i.e., satellites and sensors), which are used to collect data, and directly transmits the data to the ground base station. The ground base station pre-process the raw data in which they extract useful information from the raw data in order to create data blocks (called granules). Afterward, these data blocks are transmitted to Data Aggregation Unit (DAgU), where it aggregates all the data blocks, and arrange them in a sequential manner based on their unique machine ID. Furthermore, fusion algorithm is employed, which is used for partitioning the data blocks. The exploitation of the fusion algorithm is used to disseminate equal size data in each divide-and-conquer servers. Such technique not only helps in enhancing the efficiency of the divide-and-conquer servers but also helps the system in fast data processing. Finally, the partitioned data blocks are sent to the Decision Making Unit (DMU), which are used for analysis as well storing the results. The Decision Making server can utilize those results depending on the requirement of the user.

The proposed architecture welcomes both real-time as well as offline data (e.g., GPRS, xDSL, or WAN). The contribution of the proposed system architecture is summarized as follows.

1. The data aggregation technique concatenates all the data being generated by various machines in larger block.
2. The larger data blocks are arranged in a sequential manner based on the machine ID. Afterward, these data block are partitioned into smaller data blocks (granules).
3. These granules are then sent to D&CPU, in which each granule is sent to one server for final processing.
4. The result storage device helps the user to get their desired results at any times, which can be used for future comparison, if needed.

The proposed divide-and-conquer data analytical architecture for Big Data in M2M has several advantages, such as, at data acquisition stage, the data is concatenated to form a Big Data block that helps the system to combine the same data type, the fusion domain helps in enhancing the efficiency of D&CPU by dividing the data into smaller data blocks. Each block is then sent to a single server for further processing, which helps in increasing the processing efficiency, and finally, users can use the desired results for comparison purpose.

The remainder of this paper is organized as follows. In Section 2, we give a detailed survey of the Big Data. In Section 3, we briefly explained the requirements for the Big Data using divide-and-conquer approach for M2M. In Section 4, we describe the proposed system architecture for divide-and-conquer approach in M2M. In Section 5, we present a detailed analytical and simulation results using Hadoop. Finally, Section 6 offers a conclusion and future work of the paper.

2. Related work

Big Data and its analysis are at the verge of modern science and business, where author highlights the identity of number of sources on Big Data such as online transactions, emails, audios, videos, search queries, health records, social networking interactions, images, click-streams, logs, posts, search queries, health records, social networking interactions, mobile phones and applications, scientific equipment, and sensors [3]. The proposed model is using conventional database tools. The challenge is to capture, form, store, manage, share, analyze and visualize the Big Data. In addition, the characteristics of Big Data, such as a variety, volume, and velocity, the three main characteristics of Big Data are elaborated briefly in Section 1.

The concept of Big Data is stimulating a broad range of curiosity in the industrial sector [11]. The report provides concrete examples of Big Data generated by sensors. For instance, manufacturing companies use various machines (e.g., sensors) which are embedded in monitoring usage patterns, predicting maintenance problems and enhancing the product quality in their machinery equipment. On analyzing data streams generated by the embedded machines, allows manufacturers to improve their products in their machinery. A massive volume of data is generated by numerous machines deployed in the supply lines of utility providers, which are constantly monitoring the production quality.
A. Ahmad et al. / Neurocomputing [ ] 3

safety, maintenance, and so forth. The electronic sensors that are frequently monitoring the mechanical and atmospheric conditions are high-quality example of sensors generating a bulk of Big Data. Furthermore, sensors are used for healthcare sectors (for monitoring biometrics on human body, health care diagnoses, patients' conditions, treatment phases, and so forth) are huge source of information for Big Data presented in [11]. However gathering the sensors’ data from numerous sensors in an energy efficient method remains, beyond expertise’s of the report.

Cloud-based federated framework for sensor services objective is to enable a seamless exchange of feeds from large numbers of heterogeneous sensors [10]. In literature, densely distributed wireless sensor network is used for various applications generating Big Data is also discussed. In health care sectors, the information (e.g., blood pressure and heart rate provided by numerous sensors are used to understand remote medical care services [11,12]. In addition, for arranging prompt dispatch of ambulances, the use of patients’ location information is also been carried out. In [13,14] researchers observed various animal habitats by gathering massive volume of data from location sensors attached to animals since densely deployed wireless sensor network in large geographic location yields in devastating amount of data. Due to a limited range of wireless communication, wireless sensor network may be divided into sub-networks since it is difficult to aggregate Big Data generated by the densely distributed wireless sensor network.

Literature studies state that, conventional research works is widely used for data gathering using the mobile sink in wireless sensor network. One of the most prominent and earliest studies in mobile sink scheme is Data Mobile Ubiquitous LAN Extensions (MULEs) [15]. Data mules can be the second alternative of the mobile sink. First the sensors nodes are divided into clusters, and the second task is to make a decision about the route for perambulation of each cluster. Regardless of the sensors node location, the mobile sink node divides sensor nodes into grids, which patrols the grids by using a random walk between the neighboring grids with assumption of simple data collection scheme [15]. Nevertheless, this type of clustering might result in inefficient data gathering since it is not based on the nodes’ location. In a cluster, if there is no sensor node, perambulation of empty cluster would lead to wastage of valuable time, and it also degrades the efficiency of the system. Moreover, randomness patrolling results in unbalancing visits to the clusters with different sensor nodes. As a result, mobile sink fails to collect the information.

One of the most famous clustering algorithms techniques used in wireless sensor network with static sink node is Low-Energy Adaptive Clustering Hierarchy (LEACH) [16]. In LEACH, the clustering algorithm is carried out by each sensor node in the network. In this scheme, residual energy of each sensor node are exchanged since sensor node having higher residual energy can be in a better position of becoming a cluster head. The energy consumption of each sensor node becomes eventually equal by performing periodical re-clustering. Nevertheless, LEACH has several limitations, for instance, wireless sensor network are deployed in wide areas are not able to use the algorithm since it is based on the assumption that each node can communicate with all other nodes in the network. In addition, LEACH unsurprisingly null over the restriction of the node’s communication range, like most of the other distributed algorithms such as K-hop Overlapping Clustering Algorithm (KOCA) [17] and k-hop connectivity ID (k-CONID) [18], respectively. The focus of KOCA algorithm is on multiple overlapping clusters, and its designed is based on a probabilistic cluster head selection and nodes’ location. Furthermore, in k-CONID, the nodes randomly exchange their id. The node having minimum id within a k-hop is then chosen as a cluster head.

For distributed clustering algorithm, it is difficult to minimizing data transmission in wireless sensor network. The distributed clustering algorithm cannot achieve considerable optimization since wireless sensor network is physically divided into sub-networks, where a node cannot have the information about its surrounding nodes in the network. With a view to achieving minimum energy clustering, centralized clustering algorithm would be more efficient. Furthermore, for mobile sink, the centralized clustering algorithm is carried out by a supernode can be the appropriate option. Some of the existing centralized clustering algorithms are Power-Efficient Gathering in Sensor Information Systems (PEGASIS) [19] and KAT mobility (K-Means And TSP mobility) [20]. PEGASIS algorithm is based on position that constructs chain clusters of nodes and repeats cluster head selection for efficiency. Uniform energy consumption is also achieved in PEGASIS algorithm, and it also considers limitation of the communication range. On the other hand, by using this algorithm, it is difficult to get minimum energy consumption since the clustering algorithm employs greedy algorithm techniques. However, by means of K-means algorithm, KAT mobility divides the nodes into clusters since K-means algorithm is based on location of the nodes. The structure of KAT mobility algorithm is designed without considering the communication range limitation, and it also results in an optimal cluster that reduces energy consumption. As a consequence, the failure probability of the mobile sink is such circumstances very high.

Recent Innovations in Big Data sensing and computer technology modernize the way remote data collected, processed, analyzed, and managed [21–23]. Mainly, integration of most recent designed sensors to various platforms of earth and planetary observatory is now a day’s generating a continuous stream of data. We referred the continuous flow of data to ‘Big Data’ that is leading us to a new world of challenges [10]. Such consequences of transformation of remotely sensed data to the scientific understanding are a critical task [11]. The rate at which volume of the remote access data is increasing, number of individual users as well as organizations are now demanding an efficient mechanism to collect, process and analyze and store these data and its resources.

Recent research techniques for analyzing Big Data in M2M can be classified into two types, namely real-time Big Data analysis and offline Big Data analysis. However, to achieve both, our proposed system architecture provides a suitable platform for delivering desirable results using both types of data analysis.

3. Divide-and-conquer based Big Data architecture requirements

With a view to supporting scenarios expressed in the previous sections, we categorized the following requirements to be supported by proposed Big Data architecture for M2M communication. The following requirements are not only specific to divide-and-conquer approach in M2M but can also be applied to any M2M/IoT platforms.

Heterogeneity: The designed system must handle various machines, i.e., satellites, sensors, different data formats, and protocols. Reconfiguration: The designed system shall be deployed in a broad range environments. Thus, one must be able to reconfigure it remotely.

Scalability: The designed system must be scaled in two dimensions such as vertical and horizontal scalability. Vertical scalability is used for storage purpose, e.g., enhancement of database size. Whereas, the horizontal scalability is referred to processing purpose (e.g., load balancing at filtration and load balancer device in system architecture)
Services: The designed system must be endowed with a method that supports those users, who would like to take back the aggregated data, at the right level of abstraction.

The contribution of the paper is delineated in Fig. 1, i.e., divide-and-conquer approach for Big Data in M2M. The proposed architecture is comprehensive as it addresses the complete spectrum of the elements involved in it.

- **Satellites**: In this study, we consider various machines (e.g., satellites, sensors, ground station, and receiving antenna), transforming wireless signals to machine readable data. Typically, these machines are used to transform such quantity (e.g., river, jungles, glaciers, deserts, etc.) into electrical resistance values.
- **Data bus**: Data bus is used in Data Aggregation Unit (DAU), which helps in the transmission of the fused data to fusion result storage device.
- **Fusion results storage device**: This device bridges the DAU with the D&CPU, which can store fused data that can be efficiently retrieved by the filtration server.
- **Filtration and load balancer device**: This device is used to filter the fused data, i.e., the designed divide-and-conquer algorithm is applied to each granule data, and distributes each granule equally in all servers for further analysis.
- **Processing Servers**: Processing server follows the technique of the divide-and-conquer mechanism, in which each server analyze the data and then conquer the data to produce desired results.
- **Result storage device**: This device is meant for storing the result of D&CPU, which can be used by the decision-making server; based on user requirements.
- **Decision-making Server**: Decision making server follow various algorithms designed by the user.

The designed architecture fulfills all the requirements for analyzing Big Data in M2M; however, some challenges must be considered while designing the real-time scenario. There is no such standard for sensors [2], or satellite manufacturers, each of them have their format of data or for configuration of sensors, satellites or other machines, depending on the user requirements. Therefore, implementing a sensor network [2] or a satellite network is error prone and time-consuming when the ultimate object is to collect massive datasets for further exploitations. On the other hand, sometimes the brands are obsolete and no more available to customers. These challenges are out of the scope for this work, as our focus is on remote access data by using satellites.
4. Big Data in M2M

In this section, we proposed divide-and-conquer approach for Big Data analytics in M2M communication. Firstly, we argue the motivation that justify our proposal. Then, a detail description is given about the System Model. Finally, a detail description is given in depth.

4.1. System model

Set of machines (satellites) are deployed in an earth arbitrary system, denoted by $\mathcal{U} = \{S_1, S_2, \ldots, S_N\}$ for collecting data using a set of sensors or cameras from a Gaussian random field such that each source $Z_i, i \in \mathcal{U}$, is a Gaussian random variable with mean $\mu_i$ and variance $\sigma_i^2$, and collection of data $Z_i = \{Z_0, Z_1, \ldots, Z_k\}$ follows multi-variant Gaussian distribution. The covariance matrix of the N Gaussian sources is denoted as $\sum = \sum_{i=1}^{N} \sigma_i^2 \delta_{ij}$ with $\sigma_i$ being covariance between machine $i$ and $j$ as follows:

$$\sigma_{ij} = \sigma_i \sigma_j e^{-\frac{d_{ij}^2}{\Delta^2}}$$

(1)

where $c$ is the correlation exponent and $d_{ij}$ denotes the Euclidean distance between machines $i$ and $j$.

In view of the fact that Gaussian source is continuous. Therefore, we have made an assumption about the data source, which is quantized with already pre-configured quantization level $\Delta$. However, for digitized observation of machine $i$ is denoted by $Z_{ij}^*$ [24] (Interested reader are referred to [24] for more details). Having a fact that small quantization level, the joint entropy $H(Z_{ij}^*)$ of the quantized source $Z_{ij}^*$ can be expressed as:

$$H(Z_{ij}^*) = \frac{1}{2} \log_2 \left[ \frac{2 \pi e}{3^2 \sigma_i^2} \right]$$

(2)

4.2. Divide-and-conquer approach using Big Data in M2M

The term Big Data covers diverse technologies same as cloud computing. The input of Big Data comes from social networks (Facebook, Tweet, LinkedIn, etc.), web servers, satellite imagery, sensory data and banking transactions, etc.

Regardless of very recent emergence of Big Data architecture in scientific applications, numerous efforts towards Big Data analytics architecture can already be found in the literature. Among numerous others [25–29], the divide-and-conquer based Big Data analytical architecture in M2M is proposed to analyze the Big Data in an efficient manner. Fig. 1 delineates 'n' number of satellites that obtain the Earth Observatory Big Data images with sensors or conventional cameras through which scenarios are recorded by using radiations. Special techniques are applied to process and interpret remote sensing imagery for the purpose of producing conventional maps, thematic maps, resource surveys, etc. We have classified divide-and-conquer based Big Data analytical architecture into four units, such as, Data Acquisition Unit (DAcU), Data Aggregation Unit (DAgU, Divide), Divide-and-Conquer based Processing Unit (D&CPU), and Decision Making Unit (DMU). The functionalities and working of the said parts are described below.

4.2.1. Data acquisition unit

The deployment of the satellites around the globe encourages the growth of Earth Observatory System as cost effective parallel data acquisition, which is used to gratify for explicit computational requirements. This standard has already been provided by the Space Science Society for parallel process in this context [30]. The traditional data processing technologies could not provide sufficient power in order to process massive amount of data that is collected by the satellites. Hence, satellites instruments for an Earth Observatory System can be a better choice for data acquisition in a more sophisticated manner. With a view for such background, there is a need to design a system that could efficiently collect and process the data. Therefore, the designed architecture introduces DAcU for Big Data applications. The designed attribute gathers data from various satellites around the globe with the help of ground base station as shown in Fig. 2.

The data is transmitted to the ground base station using downlink channel. This transmission is achieved either direct communication or with the help of relay satellites with an appropriate tracking antenna and communication link in the wireless atmosphere. We assume that the data must be corrected with different methods for removal of distortion caused by motion of the platform relative to the earth, platform attitude, earth curvature, nonuniformity of illumination, variations in sensor characteristics. We also assume that the correction of erroneous data is done at satellites end. In addition, the transformation of raw data into image form is performed by the satellites using Doppler or SPECAN algorithms [31]. However, the biggest challenge should be noticed that how the ground base station distinguishes satellites data? In order to cope with such constraint, the satellites transmit their data to the ground station with a unique satellite ID. We divided the DAcU into two processing unit, such as offline data processing and real-time data processing. In offline data processing unit, the data is pre-processed and store in data center for storage. Whereas, the data centers cannot store real-time data, as it will go directly to the DAgU after raw data preprocessing.

For effective data analysis, the raw data pre-processing processes the data under many situations to integrate the data from

Fig. 2. Remote sensing earth observatory image.

Please cite this article as: A. Ahmad, et al., An efficient divide-and-conquer approach for big data analytics in machine-to-machine communication, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.04.109
various sources. Some relational raw data pre-processing techniques are applied, such as data integration, data cleaning, and data redundancy elimination. The incorporation of raw data pre-processing technique at DACU combine satellites data with their unique satellite ID and store them in its servers. This technique is used to create equal size of data blocks (granules). With a view to enhancing the efficiency of divide-and-conquer mechanism (described in later section), the raw data pre-processing technique not only decreases storage capacity, but it also helps in dividing the Big Data block into equal size, which helps in improving the analysis accuracy at each of its server.

After the raw data pre-processing phase, the data blocks are then aggregated at DAgU. The detailed functionalities of DAgU are described in the next section.

4.2.2. Data Aggregation Unit

This section describes the mechanism of data aggregation, in which the Big Data blocks are sent to DAgU. At this stage, aggregation driven algorithm is needed to aggregate the data blocks and generates single data block of the same satellites using their unique ID. Therefore, to address the features mentioned above, two main techniques are introduced. Firstly, a method is required for aggregation of same satellite data. Secondly, Data Fusion Domain (DFD) is used that enhances the computational capabilities of the system.

In the proposed divide-and-conquer based Big Data architecture, DAgU sends the Big Data blocks to the Data Aggregation Domain (DAD), where it performs aggregation on the received data blocks. Since the aggregated data is not in organized and compiled form. Therefore, there is a need to organize the aggregated data into the proper format for further processing and to store them. The storage capability helps the aggregated data for providing the sequence to the data blocks, e.g., > i.e., the intermingled data is separated from each other on the basis of satellite ID. It combines the same aggregated data captured by the same satellite. In the proposed architecture, aggregation and compilation server is supported by various algorithms that compile, organize, store, and transmit the results. Again, the algorithm varies from requirement to requirement and depends on the analysis needs. This technique not only enhances the scalability of the proposed architecture but also play a fundamental role in the divide-and-conquer domain for equal division of data among its servers.

After data aggregation, the sequence data blocks are transmitted to DFD. The detailed explanation of DFD is described below.

a. Data fusion domain

In order to make everything clear in Big Data M2M communication, special requirements are required to integrate and combine the multifaceted data in order to fuse knowledge [32–34]. This is a pre-process analysis technique for the divide-and-conquer mechanism that would improve the computational efficiency of the entire system. This knowledge can endow with unambiguous advice and direct either ours or machine's measurements. For this reason, it is essential to process this inundation of multifaceted data to facilitate the intelligence of M2M.

Let assume the M2M Big Data modeling in which massive volume of data is received from N number of machines. Also, let us assume that the nature of the collected data is heterogeneous and has been converted and interoperated semantically. For instance, satellite $\xi (1 \leq \xi \leq 2)$ are the measurements types of data. The Z number of machine (satellites) have K types of data measurements. In the given scenario, each type of data measurements for a single machine, there are T measured values collected at T times, $t_i (1, 2, 3 \ldots T)$. we use the $T^* \sum_{i=1}^{Z} K_i Z$ matrix R to express $\sum_{i=1}^{Z} K_i$ types of measured values derived from $S$ number of machines at T time difference [34].

With a view to verifying association between multifaceted data measurements and certain states of the facilities, the machine data type measurements can be articulated as attributes, such as $a_0, a_2, a_3, \ldots, a_6$ whereas states can be articulated as the decision attribute set D.

i. Big data fusion

Keeping in view the enhancement of the computation efficiency of analyzing multifaceted data in M2M, the incorporation of data fusion algorithm can play a vital role to achieve the desired output. The data fusion algorithm is based on partitioning of larger data sets as shown in Fig. 1. The idea of data fusion is based on division of the massive volume of multifaceted data divided into smaller block (granules). These smaller data blocks are easy to store in fusion result storage device (described in the next section). This technique is not only enhancing the efficiency of data analysis, but it also reduces the load on the divide-and-conquer mechanism (load refers to equal distribution of large data sets among divide-and-conquer servers). In addition, it is also possible that the proposed architecture could also process the data with missing values. We have divided data fusion technique into following stages.

Stage 1: In data fusion technique, data pre-processing is capable of performing two tasks. First task is to normalize the data blocks in the information system $S$, which can help the data to be compared. This comparison mechanism is used for division of data blocks in equal size. The second task is to replace the missing values or incomplete information with "\*". This technique helps in enhancing the processing speed in the fusion algorithm.

Stage 2: In this technique, the multifaceted partitioned data block is based on the partition of attributes governed by certain values, yields the division of high dimension multifaceted data into lower dimension smaller data sets.

For one machine, if we use $K_i * T$ matrix W, which can be articulated to measure the values derived from $K_i$ of measurements at T different times i.e., $S = (W_{11}, W_{22}, W_{33}, \ldots W_{ij})$. As shown in Fig. 1, we achieve $K_i$ types of data measurements at same amount of time from Z sensors that can be expressed by the matrix X, therefore, the above equation can be written as, $S = (X_{11}, X_{22}, X_{33}, \ldots, X_{ij})$. Hence, the desired result can be achieved i.e., high dimensional data can be broken down into Z or K blocks.

4.2.3. Divide-and-Conquer Based Processing Unit (D&CPU)

After the data (fusion results) being received by D&CPU, the first job is to store the result in Fusion Domain Storage Device. Such storage device stores the data blocks in their servers so that it can be efficiently retrieved by the processing servers located in D&CPU. After storing the results, these results are sent to filtration and load balancer server. Filtration and load balancer server performs two tasks i.e., filter the data and equally distribute the fusion results in all divide-and-conquer servers. The rationale behind filtration and load balancer server is to separate the fusion results from each other so that it can be equally distributed in all the servers. Such technique helps the divide-and-conquer servers to process the data efficiently for performing parallel tasks. In the proposed architecture, the employment of divide-and-conquer algorithm is not specific, however, it depends on the application scenario. In our scenario, we have designed a divide-and-conquer algorithm that extracts the features of Land and River in the data sets obtained from European Satellite Agency. Details about the proposed algorithm are presented in section 5.3.
4.2.4. Decision-Making Unit (DMU)

In the proposed architecture, DMU is the final stage of data analysis and process, which is classified into three components, such as, fusion results storage device, decision-making server, and communications infrastructure. When the results are ready for compilation, the divide-and-conquer server in D&CPU sends their results to the fusion result storage device in DMU for storage purposes. Since the divide-and-conquer server results are not in organized and compiled form, therefore, there is a need to organize the results in a proper form for further processing and to store them. In the proposed architecture, Result and Storage server is supported by various algorithms that organize, store, and transmit the results to Decision Making Server with the intention that can be used by any server for its processing at any time. Again, the algorithms depend on user’s requirement of analysis.

As the Decision Making Server is supported by various algorithms, which request from the results storage about various parameters in order to make various decisions (e.g., in our analysis, we extract the features of the river). The decision algorithm should be intelligent and efficient enough that could efficiently produce results to find the hidden parameters as well as to make everything clear. The decision part in the proposed architecture is very important since a small mistake in the algorithm can degrade the performance of the entire analysis. After the decision has been made, DMU finally displays the desired output so that any application can utilize these decisions at real-time or offline for the developmental purpose, such as, business software, general purpose community software, or other social networks, etc. that need those findings (i.e., decision making).

5. Big Data Satellite product analysis for continuous feature extraction

The main purpose of the analysis is to extract continuous features presented in the satellite images, such as, rivers, main highways, etc. For sake of simplicity, we are focusing on the analysis of River detection. In this section, we are discussing the datasets, tools being used during the analysis and implementation of the algorithms using Hadoop. We also give the details about the experimental environment in which we develop and evaluate our proposed algorithm. Afterward, the analysis finding and its discussion are described in the next section.

5.1. Datasets, analysis tools, and implementation environment

Datasets are taken from European Space Agency (ESA) [35] for analysis and testing that contain various earth observatory satellite products by monitoring different locations on earth. Two main satellite sensors’ data i.e., Advanced Synthetic Apertures Radar (ASAR) and medium resolution imaging spectrometer (MERIS), of ENVISAT mission, is taken for analysis as shown in Table 1. ENVISAT was working and monitoring earth from approximately 800 KM above the surface [36]. Different types of products that are subjected to the area covered are examined, such as, Sea area, Land area, Ice area, etc. as shown in Fig. 3. ESA monitored products contains satellite image data of various countries, such as, European Countries i.e., Italy, Greece, Spain, Morocco, Poland, Canada, African countries i.e., South Africa, Mauritania, etc. and USA as well. In the mentioned figure, Product 10 covered the area of Ice, Land, and Sea from Canada, Product 7 contains the data from the Sea and Land area in between of Spain and Morocco, whereas, Product 9 and product 1 is from USA and Vietnam. Table 1 shows detailed information about the datasets used in our work, such as, the product name, image mode, mission, capturing date and time; the area covered, monitored country, size in MB, the absolute orbit of the satellite, the phase and cycle as well. Finally all the products of almost 1.7 GB size are combine to test the system on larger datasets.

EnviView, Beam, and Nest [36] are three popular tools that provide visualization and understanding of ESA earth observatory (EO) products. While understanding and performing basic analysis of products, we use EnviView 2.8.1, Beam VISAT-5.1, and Beam Nest 5.1. For our complex analysis, Hadoop provides an efficient solution through parallel programming and divide-and-conquer facilities [37]. Hadoop 2.3.0 with Map Reduce java programming is used for algorithm development using divide and conquer mechanism.

We developed and tested the proposed algorithm to extract the features of river using divide-and-conquer mechanism on corei5 3.20 GHz x4, UBUNTU 14.04 local machine with Hadoop single node environment.

### Table 1

Datasets details.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Mission/sensor</th>
<th>Capturing date</th>
<th>Area covered</th>
<th>Country</th>
<th>Absolute orbit/phase cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 2. ASA_APM_1PNPDE200819_093008_00000622008_00394_02452_0009 (33 MB)</td>
<td>ENVIASAR</td>
<td>8/19/2002 9:30:08</td>
<td>Land</td>
<td>Poland and Germany</td>
<td>2452/2.08</td>
</tr>
<tr>
<td>Product 4. ASA_WSM_1PNPDE20050331_075935_00000552036_00303_16121_0773 (55 MB)</td>
<td>ENVIASAR</td>
<td>31-MAR-2005 07:59:36.4091</td>
<td>Sea and Land</td>
<td>Capetown, South Africa</td>
<td>16,121/2.36</td>
</tr>
<tr>
<td>Wide Swath Mode Image</td>
<td>ENVIASAR</td>
<td>17-NOV-2002 12:58:52.00</td>
<td>Sea and Land</td>
<td>Spain</td>
<td>374/2.11</td>
</tr>
<tr>
<td>Wide Swath Mode Image</td>
<td>ENVIASAR</td>
<td>18-OCT-2002 12:58:52.00</td>
<td>Sea and Land</td>
<td>Poland</td>
<td>2452/2.2</td>
</tr>
<tr>
<td>Wide Swath Mode Image</td>
<td>ENVIASAR</td>
<td>19-Aug-2002 09:30:43</td>
<td>Sea and Land</td>
<td>Spain and Morocco</td>
<td>729/2.18</td>
</tr>
<tr>
<td>AP Mode Brows Image</td>
<td>ENVIASAR</td>
<td>23-July-2003 10:45:43</td>
<td>Land and Sea</td>
<td>Tunisia, Libya, Greece, Italy</td>
<td>23,408/2.50</td>
</tr>
<tr>
<td>AP Mode Brows Image</td>
<td>ENVIASAR</td>
<td>13-August-2003 18:13:39</td>
<td>Land and Sea</td>
<td>USA</td>
<td>7596/2.19</td>
</tr>
<tr>
<td>AP Mode Brows Image</td>
<td>ENVIASAR</td>
<td>26-April-2010 15:48:28</td>
<td>Land, Ice, Sea</td>
<td>Canada</td>
<td>42,636/2.88</td>
</tr>
</tbody>
</table>
node setup having 4 GB RAM and Gallium 0.4 on AMD OLAND graphics.

5.2. Analysis findings and discussion

The main focus of the analysis is on ENVISAT/ASAR EO products especially Product1 since ASAR Product1 has more and diverse nature of Rivers as well as diverse covered areas, such as, Sea, and small lakes, city, etc. Initially, the satellite image data is taken from Measurement Dataset (MDS) portion of the product. Keeping in view the continuous behavior of the Rivers, statistical analysis and pixel value distribution is made for exploring the properties, pattern and behavior of Rivers in the satellite image. We calculated the overall statistical measurements of the products, such as, the mean value of all the pixel values, the diversity and variation in the values to find out the nature and quality of the image data. These statistical measures are shown in Table 2, in which the number of lines shows the total rows in satellite image and number of sample/line shows how many pixels are there in each row. While exploring overall statistics of various products, we identified that the Product2 has more lines since the image quality might be lower due to its overall low mean and standard deviation values. It might also be the case that it covers some of the dark areas, such as, Forest.

In the next stage of analysis, we explore pixel view of different image blocks having different area covered by considering Amplitude_HH band of products. We observed that all the river’s pixels look similar and having a continuous nature as shown in continuous black pixels in Fig. 4(c) and (d). Land image blocks with no river have different pixel view than pixel view of the block with river as shown in Fig. 4(a) and (c). Since the pixel values of the block, with no river, are quite similar (low S.D), therefore, the difference between the mean value of the pixel values of the block and the mean value of the pixels of any sub block of that block is very low.

<table>
<thead>
<tr>
<th>Dataset/product</th>
<th>Product 1</th>
<th>Product 2</th>
<th>Product 3</th>
<th>Product 4</th>
<th>Product 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of output lines (records)</td>
<td>3915</td>
<td>5535</td>
<td>5440</td>
<td>4860</td>
<td>5940</td>
</tr>
<tr>
<td>No. of samples/line (no. of pixels/line)</td>
<td>1452</td>
<td>1478</td>
<td>842</td>
<td>5693</td>
<td>5646</td>
</tr>
<tr>
<td>Output mean (overall mean)</td>
<td>1867</td>
<td>396</td>
<td>1640</td>
<td>1057</td>
<td>1116.27</td>
</tr>
<tr>
<td>S.D. (overall S.D)</td>
<td>1136</td>
<td>221</td>
<td>1295</td>
<td>736</td>
<td>1023.1</td>
</tr>
</tbody>
</table>

Pixel value distribution of several image blocks, such as, Land block with no River, Sea block, and Land block with 1 river, and two blocks that have only one river are also inspected. The distribution among pixel values is quite low for all river pixels as shown in Fig. 5(a) and (b). Only few pixels in those blocks have the values above 600 and below 400, which results in minimum mean value. The distribution of image blocks having no River, either Land or Sea block, is entirely dissimilar from River block as shown in Fig. 5(c) and (d). Pixel values range from 1000 to more than 3000 in case of Land block with no River and from 2700 to 3500 for Sea block. Land blocks that have Rivers are also examined on pixel value distribution. The mean value and pixel distribution of the River portion in Land block are different from the other part of the block from pixel 61 to 121 as shown in Fig. 5(c). It is also apparent that the pixel values of the River portion have minimum difference among themselves as compare to the other portion of the image.

Considering the fact of greater mean difference between River pixel values and overall block pixel values, the analysis have been performed on mean difference between blocks and sub blocks by considering 10,000 pixels image blocks, which also have one or more River in the blocks. Table 3 clearly shows that the absolute difference between the mean value of river sub block and the

Please cite this article as: A. Ahmad, et al., An efficient divide-and-conquer approach for big data analytics in machine-to-machine communication, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.04.109
Mean value of complete block is very high as shown in the last column of the table. The difference for block 4 is bit lower as compare to other blocks due to river thickness and its existence in populated city area.

5.3. Proposed-divide-and-conquer algorithm

On the basis of analysis findings, we propose continuous natured feature extraction algorithm using traditional divide-and-conquer mechanisms, such as, merge sort. We consider River as a continuous natured feature presented in satellite EO products and provide detection algorithm for extracting rivers from Satellite products. We use classification decision tree REPTree [38] for small sub-block classification along with Euclidian Distance (ED) and other statistical measures to classify and extract Rivers from a satellite image. REPTree is a fast classification mechanism using tree structure, however, the technique cannot be applied directly to detect continuous natured feature extraction as River. We used REPTree for small sub-block classification with other ED and statistical measures such as mean, S.D, differences, etc., to find continuous features in EO products. In this section, we present the algorithm details including its parameters, functions, flow charts, and pseudo-code.

Algorithm parameters: following parameters and variable are used in the proposed algorithm.

- \( \partial_{b_{size}} \): Block size threshold. The minimum size of divided blocks that is going to be processed. The image Matrix is repeatedly divided into matrix blocks until it reached to \( \partial_{b_{size}} \), then analysis on each block is performed.
- \( \text{Set}_{rivers} \): Set of rivers detected. i.e R1, R2, R3, ... Rn and each Ri is also a set of ordered 3-tuples that contains the pixels information of the river
  \[ R_i = \{ (x, y, value) \mid (x, y) \in R_i \} \]
- B1, B2: blocks obtained by dividing matrix.
- Width_M: Width of the image matrix.
- Height_M: Height of the image matrix.
- \( \bar{X}_B, \text{S.D}_B \): mean and standard deviation of pixel values of block and calculated as:
  \[
  \bar{X}_B = \frac{\sum \text{Pixl values in block B}}{\text{no of pixels}}
  \]
  \[
  \text{S.D}_B = \sqrt{\frac{\sum \text{Pixel values} - \bar{X}_B^2}{\text{no of pixels}}}
  \]
- ED: Euclidian distance between two pixels.
  \[
  \text{ED}(P_1, P_2) = \sqrt{(P_1x - P_2x)^2 + (P_1y - P_2y)^2}
  \]
  Where p1: pixel1 position (P1x, P1y)
  p2: pixel2 position (P2x, P2y)
- \( \partial_{\text{Min.S.D}} \): Minimum S.D threshold set for the block that has a river.
- \( \partial_{\text{Mean.Diff}} \): Threshold set for absolute mean difference between \( \bar{X}_B \) and Mean of River pixel values.
- \( \text{NP.RDS} \): number of pixels in river set.
- \( \partial_{\text{NP.RDS}} \): Threshold set for detecting minimum number of pixel in a river set of single block.

Algorithm description and implementation: Proposed algorithm takes image data from MDS as input in matrix form. The Matrix is repeatedly divided into two equal blocks. These blocks are further divided into four more blocks, and so on so forth,
until the final block reach to minimum size threshold. Statistical measures are calculated for each subdivided block and then results are combined and conquer in the reverse order. The proposed divide-and-conquer algorithm works as similar to merge sort algorithm since initially the EO image matrix is divided from top to bottom and then processed and conquered from the bottom to top. Flowchart of the proposed algorithm “divide-and-conquer mechanism in Big Data architecture” is given in Fig. 6. The algorithm has two main functions, i.e., divide function and conquer function, and two other sub functions i.e., analyze function and River_in_block function. Divide function is the main function of the system, which is responsible for divisions of blocks recursively until the block size reached to its threshold then analyze function is called for each divided block. Analyze function perform some statistical analysis using various parameters and by using River_in_block function to detect all possible rivers i.e., Set of rivers, in each block and return the set to divide function. Finally, the results of each block are combined and compiled from down to top by conquer function.

Divide function takes a matrix block, initially an EO image data matrix. It then divides the incoming block into two equal parts, i.e., horizontally or vertically recursively until it reaches to the block size threshold. Then it calls analyze function to analyze each divided block and gets the set of rivers for each block. Later conquer methods is initiated to combine and process the results in reverse order by taking each two neighbors’ blocks results. Final

Fig. 5. Pixel value distribution for different image blocks. a) Pixel value Distribution of River, b) pixel of Distribution of River, c) pixel Value Distribution for land block with no river, d) pixel Value Distribution for Sea, e) pixel Value Distribution of both River and Land.
call of conquer method gives the final set of rivers detected in EO product. Flow chart of divide methods is shown in Fig. 7.

Analyze function, as depicted in Fig. 8 as a flow chart, takes a block from divide function and give the set of rivers detected in that block using River_in_block function. Initially it calculates statistical parameters associated with the incoming block, such as mean, S.D. We assume that the the block with River has higher pixel values (S,D) as it has both River and Land. Therefore, initially, if any block could not meet the minimum threshold, then it does not contain any River. Hence, empty set of rivers is returned for that particular block. If it meets the minimum threshold, then block is directly sent to the River_in_block function to get set of rivers. Afterward, for each detected river in the set, anomalies and false positives are reduced by using mean difference comparison. The difference between mean value of river pixels and mean value of the whole block pixels ($|\bar{X}_{RDC} - \bar{X}_{B}|$), and number of pixels for each detected river ($NP_{RDC}$) are mapped with different thresholds for reducing false positives in the set.

River_in_block function further divides the incoming block into 10x10 (can be more) sub blocks SB. It detects whether each sub block SB is a River by applying REPTree classification tree algorithm using mean, S.D of SB pixel values, and difference between them as parameters. Finally, it merges the River detected SBs, which are connected to each other by making a single river. The flow chart of River_in_block function is presented in Fig. 9.

Conquer method takes two neighbor blocks results i.e., set of river1, set of river 2, and process them, combine them and finally making a single set of river for a whole product. It performs tasks for each two blocks recursively. If any one of the set is empty then the other set is returned. Otherwise, it checks the Euclidean Distance (ED) between each river in set of river1 and set of river2. Smaller, ED means the rivers from two sets are from same river so should be merged. Finally, when conquer function finished execution by processing final complete image block, it returns the set of all Rivers in the product image. Flow chart of the conquer method is shown in Fig. 10.

**Pseudo code:** Algorithm pseudocode is given that uses recursion mechanism to achieve divide-and-conquer capability. First

---

**Table 3**

| Block | Block size (no. values) | No. of rivers in block | Mean value of rivers pixels Mean_R | Mean value of the overall block Mean_B | $|\text{Mean}_B-\text{Mean}_R|$ |
|-------|------------------------|-----------------------|-----------------------------------|--------------------------------------|------------------|
| 1     | 10,000                 | 2                     | 541                               | 2125                                 | 1584             |
| 2     | 10,000                 | 1                     | 451                               | 1615                                 | 1164             |
| 3     | 10,000                 | 1                     | 3850                              | 2458                                 | 1392             |
| 4     | 10,000                 | 1                     | 950                               | 1459                                 | 509              |
| 5     | 10,000                 | 2                     | 639                               | 1843                                 | 1204             |

---

Please cite this article as: A. Ahmad, et al., An efficient divide-and-conquer approach for big data analytics in machine-to-machine communication, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.04.109
The statement of divide is if-statement that checks the incoming block size. If the size of the incoming block is less than the threshold, it stops more block division and then start analysis on that block and returns results. Otherwise, it divides incoming block into two equal size sub blocks horizontally or vertically depending upon the width and height of the incoming image block. Afterward, the division function is called again for each of the two sub blocks that further perform division after checking the block size recursively. When the block size is reduced to the threshold, it stops further division and call analyze function to get the set of Rivers in the block, and then return set of rivers to the function calling statement. Analyze function detects the Rivers within a block by using Rivers_in_Block function and then compares mean difference between the River pixel value and the whole block value with mean difference threshold and also number of pixels with the threshold for each River in river set. Rivers_in_Block are used for detecting Rivers in blocks by further dividing blocks into fixed size sub blocks and using REPTree method. The conquer function is called from divide function for each of the two set of Rivers. The conquer function combines the results from two block. It combines two rivers into one, if their ED is short, meeting threshold and makes a single River set from those two incoming River sets. Finally, when all the results are compiled, the conquer function have only one set of rivers that contains all the River in the whole product.

1: Divide (image_matrix M)
2: {
3: if (size (M) <= \( \text{size} \) \(_B\)) {
4: Set_Rivers = Analyze (M);
5: Return Set_Rivers;
6: }
7: //end of if
8: if (Width_M < Height_M)
9: {// divide m into two parts vertically
10: B1 = M [0-Width_M/2][Height_M]; //first half of M
11: B2 = M [Width_M/2-Width_M][Height_M]; ///2nd half of M
12: }
13: else {
14: B1 = M [Width_M][0-Height_M/2]; //Uper half of M
15: B2 = M [Width_M/2-Height_M][Height_M]; // Lower half of M
16: }
17: // end of if else
18: //Recursion and division
19: Conquer (Set_Rivers1, Set_Rivers2); //combining blocks and results of blocks.
20: }//end of Divide

1: Analyze (Image_Matrix_Block B)
2: {
3: Calculate \( \bar{X}_B \), S.D. B;
4: if (S.D. B <= \( \text{s.D.} \) \(_B\)) // Block does not have any river.
5: Set_RiverDataClass_Set = \( \Phi \) return Set_RiverDataClass_Set;
6: // end of if
7: for each (RiverDataClass RDC: Set_RiverDataClass_Set)
8: if (\( |X_RDC - \bar{X}_B| < \text{mean diff} \) \&\& \( \text{NP}_RDC < \text{np_rdc} \)) Remove RDC from Set_RiverDataClass_Set;
9: }
10: //ReturnSet_RiverDataClass_Set; // Return set of rivers detected.
11: }

1: Rivers_in_Block (Matrix block B, Double \( \bar{X}_B \))
2: {
3: Define Set_R = \( \Phi \);
4: Devide B in to sub blocks S_B of size 10 x 10;
5: For each (S_B) Do
6: {
7: Calculate $\bar{X}_{SB}$, $SD_{SB}$, $|X_B-\bar{X}_{SB}|$;
8: If(REPTree($X_{SB}$, $SD_{SB}$, $|X_B-\bar{X}_{SB}|$) = river)
9: {
10: Set$_R$ = Set$_R$ U S_B;
11: }
12: }
13: $\forall$ SBi, SBj $\in$ Set$_R$ where (i$\neq j$), if (ED (SBi, SBj) < =2) the merger SBi, SBj
14: return Set$_R$
15: }

1: Conquer (Set of Rivers Set$_Rivers1$, Set of Rivers Set$_Rivers2$);
2: {
3: If (Set$_Rivers1$ = $\Phi$) then return Set$_Rivers2$;
4: If (Set$_Rivers2$ = $\Phi$) then return Set$_Rivers1$; //if either set is empty then no need to combine.
5: }
6: For each (River R1: Set$_Rivers1$)
7: For each (River R2: Set$_Rivers2$)
8: {
9: If(ED (R1, R2) < $\delta_{Ed\_Rivers}$) then R1 + R2;
10: //Combine R1 and R2, remove individual entries of R1 and R2
11: }
12: Return (Set$_Rivers1$ U Set$_Rivers2$); Combine RiverSet1 and Set$_Rivers2$
13: }

5.4. Results and evaluation

The proposed algorithm implementation is based on simple Java programming as well as Hadoop (Map Reduce). The algorithms are executed on ASAR and MERIS products for correctness and processing time measurements. It detects four rivers from Product 1, 2 rivers from Product2, 2 rivers from Product3, 3 rivers from Product4, and 2 rivers from Product 5. The detection mechanism could be improved depending upon the satellite image quality and its image taking height. The satellite image of Product1 and the sample river detected image are shown in Fig. 11.

The continuous feature extraction is implemented using simple Java iteration mechanism and as well as Hadoop divide and conquer mechanism. The implementation of the proposed algorithm using Map Reduce divide-and-conquer mechanism is more efficient than simple Java iteration implementation due to its divide and conquer nature as shown in Fig. 12. The bars in the graph show average processing time in seconds to process 1 MB data from various ESA products. The blue color bars shows the average time consumed in seconds to process 1 MB of ESA products by simple Java iteration, while red color bars represents average time taken in seconds to process 1 MB of ESA products by Map Reduce Java implementation on single node Hadoop. Results show that the Map Reduce implementation takes only half second average time to process 1 MB of image data for most of the products. We also observed that simple Java iteration takes more time to process ESA products than Map Reduce implementation. Moreover, ASA–APS products take longer time for its process than other products (i.e., for both simple Java iteration implementation and Map Reduce implementation). ASA–WSM products are processed more efficiently than other APS, APM, FRS, and RR mode products. The processing time varies from product to product due to variation in image bands, and different image modes depending on the product type. ASA–APS product size is larger than other product so

Please cite this article as: A. Ahmad, et al., An efficient divide-and-conquer approach for big data analytics in machine-to-machine communication, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.04.109
the processing time is exponentially increases due to the multiple nested iteration mechanism in the java iteration implementation and more division, conquering, and loop in Hadoop implementation.

We also implement the proposed divide and conquer continuous feature extraction algorithm using simple Java recursion and compare it with Hadoop Map Reduce implementation by taking product of various sizes. Initially, when the size of product is smaller, Java implementation produce results more efficiently then Hadoop implementation. However, when the size is crosses 100 MB, Hadoop out performs Java since Hadoop divide datasets into independent blocks of 64 MB. Therefore, upon exceeding the file size from 64 MB, the Hadoop system provide the capability of parallel processing that results in increasing the memory efficiency. On the other hand, Hadoop Map Reduce requires more input and output operation when file size is smaller. Therefore Java produces good results for smaller products. The average processing time with respect to the product size for both Java and Hadoop implementation is shown in Fig. 13.

6. Conclusion and future directions

In this paper, we proposed an architecture Big Data in M2M that uses a divide-and-conquer mechanism for analysis purposes. The proposed system architecture is capable arranging data block in a sequential manner by using machine ID. In order to achieve the computation efficiency, the data fusion domain is used to partition the data block. These data blocks can be equally distributed among various servers that follow the in divide and conquer mechanism. These units implement and design algorithms for each level of the architecture depending on the required analysis. The proposed system architecture is a generic model (application independent) that is used for any Big Data analysis. The advantage of the proposed system is to extract the features from Big Data depending upon the user requirements. Furthermore, the divide-and-conquer algorithm is proposed to detect continues natured features such as, river from Earth observatory satellite products. The algorithm implementation using Map reduce provides better results as compare to simple Java implementation. Map reduce implementation takes less than 0.5 s average processing time to process 1MB data of most of the ESA satellite products.

For future work, we are planning to extend the proposed architecture to make it compatible for efficient and real-time Big Data analysis for all applications, e.g., sensors, social networking, etc. We are also planning to use the proposed divide-and-conquer architecture and perform complex real time analysis on each earth observatory data for decision making such as: earth quake prediction, Tsunami prediction, fire detection, earth detection on land as well as in sea, etc.

Acknowledgments

This study was supported by the Brain Korea 21 Plus Project (SW Human Resource Development Program for Supporting Smart Life) funded by Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (21A20131600005). This work is also supported by the IT R&D Program of MSIP/IITP, [10041145], Self-Organized Software Platform (SoSp) for Welfare Devices.

References
