



Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr



Efficient Resource Management System Based on 4Vs of Big Data Streams [☆]

Navroop Kaur ^{*}, Sandeep K. Sood

ARTICLE INFO

Article history:

Received 4 July 2016

Received in revised form 31 December 2016

Accepted 20 February 2017

Available online xxxx

Keywords:

Big data streams

Cloud computing

Self-organizing maps

Characteristics of Data (CoD)

ABSTRACT

Big data streams are generated continuously at unprecedented speed by thousands of data sources. The analysis of such streams need cloud resources. Due to growth of big data over cloud, allocating appropriate cloud resources has emerged as a major research problem. The current methodologies allocate cloud resources based upon data characteristics. But due to random nature of data generation, the characteristics of data in big data streams are unknown. This poses difficulty in selecting and allocating appropriate resources to big data stream. Solving this problem, an efficient resource management system is proposed in this paper. The proposed system initially estimates the data characteristics of big data stream in terms of volume, velocity, variety and variability. The estimated values are expressed in terms of a vector called Characteristics of Data (CoD). On the other hand, clusters of cloud resources are created dynamically with the help of Self-Organizing Maps (SOM). SOM uses CoD to create and allocate cluster to big data stream. Moreover, the topological ordering of clusters formed by SOM is used to reduce waiting time. The proposed system is tested experimentally. The experimental results show that the proposed system not only efficiently predicts data characteristics but also effectively enhanced the performance of cloud resources.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Rapid development and adoption of smart objects in every sphere has amplified the prevalence of Internet of Things (IoT). The growing number of IoT devices has led to a drastic increase in data volume and data velocity. On the other hand, the heterogeneity of IoT devices enhances data variety. Consequently, IoT data is characterized by volume, velocity and variety.

According to Gartner IT Glossary [1], big data is defined as:

“Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.”

With respect to the above definition, it can be stated that the data harvested by IoT devices has entered big data era.

Furthermore, in a smart environment, some of the IoT devices are periodic while the others are event triggered [2]. The periodic devices generate constant amount of data at regular intervals,

thereby generating *big data streams*. On the other hand, event triggered devices are activated when certain event is encountered. The erratic nature of trigger event varies data flow rate. The variation in data flow rate is termed as variability. Therefore, in addition to volume, velocity and variety, IoT data satisfies another dimension of big data called variability.

The IoT devices can generate data stochastically. For example, one event triggered device may, in turn, trigger other IoT devices. In such a case, it is difficult to determine how many devices will be activated and how much data will be generated. Such stochastic nature leads to the generation of big data streams with unknown characteristics. Here, data characteristics imply volume, velocity, variety, and variability of data.

Apart from the IoT devices, big data streams are generated by other applications too, such as social media, click-streams, business transactions, GPS systems, and sensor networks. Intuitively, these applications generate huge volumes of data at high velocity. Moreover, data from these sources consists of images, video, audio and text which contribute to data variety. The trending topics on social media and daily/seasonal loads enhance variability. Therefore, big data streams from these sources are characterized by 4Vs: Volume, Velocity, Variety and Variability. It can be noted here that big data streams from most of the applications are generated randomly and therefore they too have unknown characteristics.

The incessant and unprecedented speed of big data streams escalates the problem of its real time analysis. Conventionally, cloud

[☆] This article belongs to Online Forecasting.

^{*} Corresponding author.

E-mail addresses: navonline98@gmail.com (N. Kaur), san1198@gmail.com (S.K. Sood).

computing is used to tackle this issue. But with the growth of big data over cloud [3], selecting appropriate cloud resources for such real time analysis has emerged as major research problem. The current practices [4–6] allocates cloud nodes based upon user-defined memory size, GPU power and processing power. For such an allocation, the user must be acquainted with characteristics of data (or the 4Vs of data). For example, the user selects higher memory size for higher volume data; higher GPU power for video streams and higher processing power for higher velocity and higher variability. Alternatively, user selects cloud nodes based upon the 4Vs of data stream. Such resource selection is limited by the expertise of user. Moreover, even if the user is expert, the knowledge of data characteristics is necessary.

As stated earlier, the 4Vs of incoming big data streams are unknown to the user due to random data generation by IoT devices and other applications. Therefore, user is unable to determine appropriate cloud resources for real-time data analysis.

In order to solve this problem, an efficient resource management system is proposed in this paper. The proposed system initially estimates the characteristics of data which are expected to arrive in next time interval. In order to estimate data characteristics, all the 4Vs are taken on-board. Later, the 4Vs are used to dynamically create and allocate clusters of free cloud resources with the help of Self-Organizing Maps (SOM). The clusters formed by SOM are created in topological ordered fashion such that more is the relationship among clusters, closer is their ordering. The proposed method exploits such topological ordering to reduce waiting time.

The rest of the paper is organized as follows. Section 2 investigates the related work. Section 3 presents the detailed description of proposed system. Section 4 provides experimental setup, results and discussion. Lastly, Section 5 concludes the paper.

2. Related work

The computing resources offered by traditional computing paradigms are unable to handle large volumes of complex data [7]. As a result, cloud computing emerged as a powerful technology which offers on-demand and unlimited resources. Pumma, Achalakul and Li [8] posited that suitable amount of cloud resources should be determined prior to the start of execution. Consequently, literature finds vast amounts of work on resource prediction and resource provisioning algorithms [9–11]. In addition, there are many articles which are focused on specific cloud applications [6, 12–20]. Nevertheless, the work on big data application scheduling is still sparse.

The growth of big data over cloud [3] ushers the issue of selecting appropriate cloud resources. In 2015, Vasile et al. [21] emphasized that dynamic resource provisioning is a challenging issue in big data application scheduling. The authors in Ref. [22] proposed various research directions for real time, distributed, dynamic, adaptive and multi-objective scheduling of big data applications. Sfrent and Pop [23] accentuated that efficient resource scheduling algorithm plays an essential role for big data applications. They found that best resource scheduling algorithm can be discovered under certain conditions.

In 2014, Sandhu and Sood [5] proposed a QoS based framework to determine and allocate optimal resources to big data applications over distributed cloud. In the proposed framework, the functional and QoS requirements are provided by the user. The functional requirements include processing power, GPU power, RAM and size of input data. On the other hand, the QoS requirements include response time, quality of output data and visualization of results. Based on functional and QoS requirements, the required cloud cluster is determined using Naïve Bayes algorithm. Later, the proposed framework uses SOM to allocate cloud resources to big

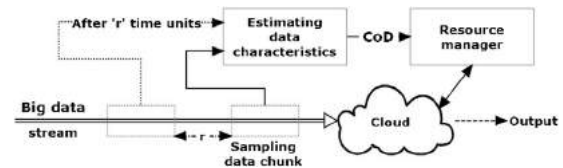


Fig. 1. Overview of the proposed system.

data application. This framework is efficient only if the functional requirements are known to the user.

Streaming big data applications escalated the need and problem of real time data processing. Various tools such as Scribe [24], Nova [25], Incoop [26], Apache Simple Scalable Streaming System (S4) [27], Apache Storm [28], and Apache Flume [29] emerged as a solution to streaming big data problem. These tools focused on processing big data streams but ignored the resource scheduling strategy. To bridge this gap, Sun et al. [30] proposed a graph based method. The proposed method schedules resources held by these tools such that energy-efficiency is enhanced without effecting the response time.

Furthermore, the authors in [31] proposed a method to processes data streams such that the profit of cloud provider is maximized. In 2016, Rahman and Graham [32] developed a priority based method for multimedia data processing. But the methods proposed in [31] and [32] are static hybrid algorithms.

The authors in [33] argued that it is necessary to predict the volume of streaming big data for efficient node allocation. They efficiently predicted the size of big data using Markov chain and assigned nodes for data processing accordingly. But the model does not consider the velocity, variety and variability of big data for resource allocation. In 2014, Castiglione et al. [34] emphasized that variability at cloud data center effects the cloud resource allocation. In 2016, Peng et al. [35] presented a strategy to allocate cloud resources to the incoming request based upon its application type (data variety). In 2015, Baughman et al. [36] illustrated that predicting the velocity of incoming big data request is crucial for efficient provisioning of cloud resources. From these studies, it is concluded the 4Vs of streaming big data are important parameters for cloud resource allocation.

To the best of our knowledge, none of the aforementioned studies considered the 4Vs of the streaming data for efficient cloud resource allocation. Therefore, the prime contribution of this paper is to exploit various dimensions of big data streams for efficient resource allocation.

3. Proposed system

The proposed system is aimed to allocate appropriate resources to big data stream based upon its 4Vs. To achieve the required goal, the proposed system works in two steps. The first step initially extracts small chunks of data from incoming stream as shown in Fig. 1. The chunk size is a small multiple of default block size of underlying storage system (such as HDFS which has default block size of 64 MB). Limiting chunk size to the size of few blocks reduces the overhead to read and analyze data. The selected data chunk is analyzed to estimate the volume and velocity of various varieties of data in the stream. While estimating these values, the variability of data is considered. In this way, all the 4Vs of big data stream are covered by the proposed system. The estimated values are expressed in terms of a vector called Characteristics of Data (CoD). The second step uses CoD to dynamically create and allocate clusters of free cloud resources using SOM.

After 'r' time units, another chunk is selected using adaptive sampling technique [37] as shown in Fig. 1. Here, 'r' depends upon the sampling rate. The initial sampling rate is selected according

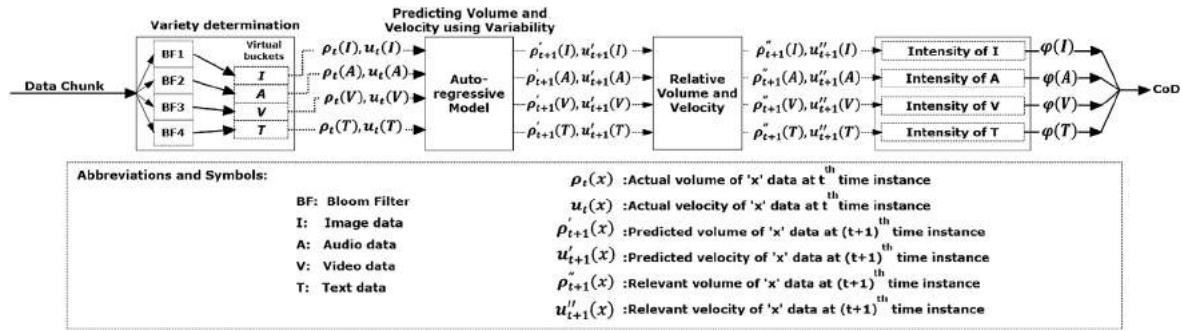


Fig. 2. Estimating data characteristics.

to Nyquist–Shannon sampling theorem (i.e. sampling rate is twice the frequency of data arrival). This mechanism allows the system to capture variability of a newly arrived stream at an early stage and thereby allocate appropriate cluster. Later, adaptive sampling technique adjusts the sampling rate autonomously. For adjusting sampling rate, migration overhead [38] is considered. The value of 'r' is increased if migration overhead is high and decreased if execution time is high. The sampled data chunk is analyzed again and resources are allocated using the above mentioned steps. The detailed description of the two steps is given in the subsections that follow.

3.1. Step 1: estimating data characteristics

For estimating data characteristics, the data chunk is analyzed in four stages as shown in Fig. 2. In the first stage, data variety is determined. The second stage estimates the volume and velocity of data using data variability. The third stage is meant to overcome the lack standard values of volume and variety which define big data. In other words, given a value of data volume and velocity, it cannot be determined whether it is big data or not. In order to tackle this issue, the third step compares the values with other request arriving at cloud and calculates relative volume and velocity. The final stage is meant for efficient representation of relative values. The detailed explanation of these stages is given below.

3.1.1. Variety determination

Broadly, big data streams consist of text data, image data, audio data, and video data. This classification of data variety is geared by the available types of big data analytics. Big data analytics refers to the techniques used for unlocking potential intelligence from data. There are mainly four types of big data analytics [39], namely, text analytics, image analytics, audio analytics, and video content analytics. Text analytics extract information from textual data such as emails, news, social network feeds, log files, business transactions and many more. Image analytics is important in wide range of big data applications. Specifically, medical and geospatial images need dedicated and rigorous image analytic algorithms to detect analogies and anomalies. Audio analytics is used to extract information from unstructured audio. For example, analysis of millions of recorded calls at call centers, diagnosis and treatment of speech related medical conditions [40], analysis of infant cries [41], and other social media audios. Video content analytics is meant to extract useful information from video streams. Video data is basically a sequence of images, arriving at a certain frequency, usually accompanied by audio and sometimes by text subtitles. Therefore, it can be said that video analytics is a more rigorous and complex form of audio, image and text analytics. With the prevalence of closed-circuit television (CCTV) and video sharing websites, video analytics is gaining momentum.

The variety determination stage uses bloom filter [42]. Bloom filter allows the stream elements that meet a criterion to pass

Algorithm 1 Bloom filter.

Input: Array $BF[n]$, Set S , Hash functions h_1, h_2, \dots, h_k
 Step 1: Initialize, $BF[1], \dots, BF[n] = 0$
 Step 2: for each $key y_i \in S$
 Step 2.1: Calculate $h_1(y_i), \dots, h_k(y_i)$
 Step 2.2: Set $BF[h_j(y_i)] = 1, 1 \leq j \leq k$
 Step 3: for each stream element 'e' arriving at the filter
 Step 3.1: Calculate $h_1(e), \dots, h_k(e)$
 Step 3.2: if $BF[h_j(e)] = 1$ for all j
 Step 3.2.1: Allow 'e' to pass
 Step 4: Exit

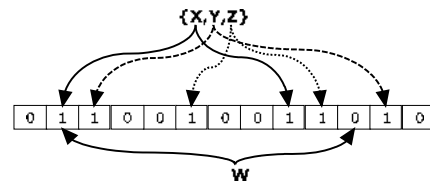


Fig. 3. Working of bloom filter with $n = 13, m = 3$ and $k = 2$.

through, rejecting the other ones. It consists of (i) an array of n bits, all initialized to 0's; (ii) a collection of k hash functions h_1, h_2, \dots, h_k ; (iii) a set S of m key values. Each hash function maps key values to n buckets, corresponding to n bits of array. The purpose of bloom filter is to allow stream elements which are in S while rejecting the other ones.

If p is the acceptable false positive rate then the values of n and k are calculated using Eqs. (1)–(2).

$$n = \text{ceil} \left(\frac{-m * \ln(p)}{[\ln(2)]^2} \right) \tag{1}$$

$$k = \text{round} \left(\ln(2) * \frac{n}{m} \right) \tag{2}$$

The working of bloom filter is given by Algorithm 1. An example of its working is shown in Fig. 3. Here, there are three key values X, Y, Z . Initially these values are hashed using the two hash functions and the corresponding bits are set to 1. When a stream element W is encountered, the two hash functions are calculated again. It can be observed that W results in a hash value which is '0'. Therefore, W is not allowed to pass through.

In the proposed system, four bloom filters are used as shown in Fig. 2. The first filter allows only images to pass and blocks the data of other varieties. Similarly, the second, third and fourth filter respectively allows audio, video and text data to pass. The implementation of all bloom filters is same except for the data type.

Here, set S of a particular bloom filter consists of all the possible data formats. For example, set S in the first filter comprises of all the image formats such as jpeg, png, etc. The set S in the second filter consists of all the audio formats such as mp3, wav, aiff, etc. Similarly, the set S of the other two filters are constructed. Hence, the value m is equal to the total number of formats in S .

The values of n and k are calculated using Eq. (1) and Eq. (2) respectively. The hash functions are selected such that every format hashes to one unique element of the array. The data which is allowed to pass is stored in its respective virtual bucket as shown in Fig. 2. Therefore, there are four buckets corresponding to the four filters (or four data types). These four buckets are initially empty. Data is added to them as it passes through the filter.

The absolute value of data volume in the bucket and the velocity at which it is added is calculated. Let $\rho_t(I)$ and $u_t(I)$ be absolute volume and velocity of image data at t th time instance. These values are initiated to zero. Every time image data is added to its respective bucket, the volume and velocity values are updated using Eq. (3) and Eq. (4). Here, δ denotes the volume of data passed through the filter at one time instance.

$$\rho_t(I) = \rho_t(I) + \delta \tag{3}$$

$$u_t(I) = u_t(I) + 1 \tag{4}$$

Similarly, the volume $\rho_t(A)$ of audio data, velocity $u_t(A)$ of audio data, volume $\rho_t(V)$ of video data, velocity $u_t(V)$ of video data, volume $\rho_t(T)$ of text data and velocity $u_t(T)$ of text data is calculated from their respective buckets. Using these values, the volume and velocity of data which is expected to arrive at $(t + 1)$ th instance is predicted at second stage of estimating data characteristics.

3.1.2. Volume & velocity prediction using variability

Data flow can be inconsistent with periodic peaks, or when something is trending on social media. Such variability leads to generation of data in high volume and at high velocity during a particular time period. Therefore, variability plays a significant role while estimating volume and velocity of data which is expected to arrive in next time interval.

In order to predict volume and velocity by considering variability as an important factor, auto-regressive model [43] is used as shown in Fig. 2. Auto-regressive model is remarkably flexible in handling a wide range of time-varying processes. It forecasts the value of variable using a linear combination of predictors. It operates under the premise that past values have significant effect on future trends. Consequently, considering variability during prediction is an inherent property of auto-regressive model.

Let $\rho'_{t+1}(I)$ and $u'_{t+1}(I)$ denote the predicted values of volume and velocity of image data at $(t + 1)$ th time instance. Using auto-regressive model, these values are given by Eq. (5) and Eq. (6) respectively.

$$\rho'_{t+1}(I) = \alpha_1\rho_t(I) + \alpha_2\rho_{t-1}(I) + \dots + \alpha_q\rho_{t-q+1}(I) \tag{5}$$

$$u'_{t+1}(I) = \beta_1u_t(I) + \beta_2u_{t-1}(I) + \dots + \beta_qu_{t-q+1}(I) \tag{6}$$

where

$$\alpha_i = \frac{\text{cov}(\rho_t(I), \rho_{t-i}(I))}{\text{var}(\rho_t(I))}$$

and

$$\beta_i = \frac{\text{cov}(u_t(I), u_{t-i}(I))}{\text{var}(u_t(I))}$$

Here $\text{cov}(\)$ and $\text{var}(\)$ stands for co-variance and variance respectively. Similarly, the predicted volume and velocity of audio, video and text data is calculated using auto-regressive model. The equations for their prediction are same as Eq. (5) and Eq. (6) with the only exception of data type, so they are omitted here.

3.1.3. Calculating relative volume and velocity

As stated earlier, there are no standard values which define big data. Therefore, the predicted values of volume and velocity in second stage are compared with other requests arriving on cloud data center. Such comparison allows the system to identify how intense is the volume and velocity of incoming big data stream. The values obtained after comparison are called relative volume and velocity. The calculation for relative volume and velocity of image data is discussed below. The calculation for audio, video and text data are similar with the only change of data type.

The relative volume and velocity of image data is calculated using Eq. (7) and Eq. (8). Here, $\max(\rho_t(I))$ and $\max(u_t(I))$ are respectively the maximum volume and velocity of image data among all the streams arriving at the system during time span 't'.

$$\rho''_{t+1}(I) = \text{round}\left(\frac{\rho'_{t+1}(I)}{\max(\rho_t(I))}, 1\right) \tag{7}$$

$$u''_{t+1}(I) = \text{round}\left(\frac{u'_{t+1}(I)}{\max(u_t(I))}, 1\right) \tag{8}$$

Here, the function $\text{round}(\text{num}, \text{num_digit})$ rounds the number "num" to "num_digit" number of decimal places. Therefore, in Eqs. (7)–(8), the relative volume and velocity is rounded to one decimal place.

Furthermore, the big data stream whose volume is equal to $\max(\rho_t(I))$, will have its relative volume equal to one. This implies that the maximum value for relative volume is one. Intuitively, the lowest value is zero. Therefore, Eq. (7) will result in the value in the range [0, 1]. Moreover, since the value is rounded to one decimal place, so {0, 0.1, 0.2, ..., 0.9, 1} is set of possible values of relative volume. Similar is the case for relative velocity.

3.1.4. Efficient representation of predicted 4Vs

After predicting the 4Vs, it is necessary to represent the values in a form which can be efficiently used to allocate appropriate cloud resources. To achieve this goal, initially, the intensity of image data $\varphi(I)$ is calculated by using Eq. (9).

$$\varphi(I) = \rho''_{t+1}(I) * u''_{t+1}(I) \tag{9}$$

The intensities of audio, video and text data are calculated correspondingly. It can be noted that the intensity lies in the range of 0 and 1, both values inclusive. This is due to the fact that both $\rho''_{t+1}(\)$ and $u''_{t+1}(\)$ lie in the range [0, 1].

After obtaining intensities of image, audio, video and text data, they are simply represented in the form Characteristics of Data (CoD) vector. CoD is defined as

Definition 1 (Characteristics of Data (CoD)). If $A = \{0, 0.1, 0.2, \dots, 0.9, 1\}$ and $\varphi(I), \varphi(A), \varphi(V)$ and $\varphi(T)$ denote the intensities of image, audio, video and text data respectively, of big data stream, then Characteristics of Data is defined by a vector (Q_1, Q_2, Q_3, Q_4) , where $Q_i \in A$ and $Q_1 = \varphi(I), Q_2 = \varphi(A), Q_3 = \varphi(V)$, and $Q_4 = \varphi(T)$. In other words, $\text{CoD} = (\varphi(I), \varphi(A), \varphi(V), \varphi(T))$.

Therefore, the first step takes a chunk of stream, estimates its 4Vs, and represents it in the form of CoD vectors. These vectors are used as input to the second step of proposed system.

3.2. Resource management

The second step works in two stages. First stage forms dynamic clusters of cloud resources while the second stage allocates an appropriate cluster to big data stream. The detailed explanation of these two stages can be given in the subsections that follow.

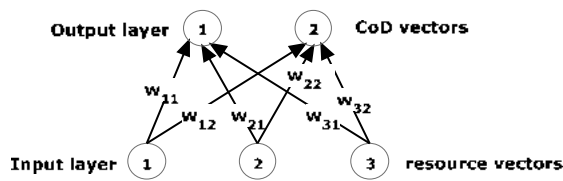


Fig. 4. Input and output layer of SOM.

3.2.1. Dynamic cluster formation

Dynamic clusters are formed with the help of SOM. SOM is one of the unsupervised neural network learning techniques for cluster analysis. It consists of an input layer and an output layer as shown in Fig. 4. The input layer consists of input vectors while the output layer consists of output vectors. Note that, for simplicity, only few vectors are shown in Fig. 4. Weights are assigned to edges moving from input to output layer.

The output vectors in SOM are represented by neurons. These neurons are arranged on a flat grid in topological ordered fashion. Fig. 5(a) shows output neurons as squares. It can be observed from the figure that the squares with different shades of red are lying towards left of the grid. Similarly, green colored neurons are lying at top right corner of the grid. Similar is the case with blue colored neurons. This color coding represents their topological ordering. In other words, the neurons with similar color coding are more closely related to one another. Furthermore, the input vectors before clustering is done by SOM are shown by grey colored circles in Fig. 5(a). They are clustered such that the topological ordering is preserved as shown in Fig. 5(b). It can be observed from Fig. 5(b) that the input vectors move closer to one of the output vector. The closer input vectors are added to a single cluster (as shown by color coding in Fig. 5(b)). In this way, the number of resources in a cluster is autonomously decided by SOM. It can also be observed that the clusters, hence formed, preserve the topological ordering of output vectors.

In the proposed system, CoD vectors are output vectors while the vectors corresponding to cloud resources are input vectors (as shown in Fig. 4). The input vectors are clustered by SOM using Procedure 1. The procedure starts with the initialization of weights and learning factor η . The learning factor defines the speed with which neural network learns. It is initialized to a value slightly lower than 1 which decreases monotonically with the passage of time. This implies that initially SOM learns quickly and later the speed of learning is decreased. After initialization, SOM randomly chooses one of the resource vectors (S_i) and finds its distance from each CoD vector. The CoD vector (C_j) with minimum distance is the winning vector. Therefore, resource S_i is allocated to stream with CoD C_j . This process is repeated for every resource. All the resources whose winning vector is C_j are said to be in one cluster.

Furthermore, the clusters formed using SOM are identified by using a parameter called Characteristics of Cluster (CoC). CoC can be formally defined as

Definition 2 (Characteristics of Cluster (CoC)). If $A = \{0, 0.1, 0.2, \dots, 0.9, 1\}$ and $\text{CoD} = (Q_1, Q_2, Q_3, Q_4)$ is the winning output vector for the resources in cluster, then Characteristics of Cluster is defined by a vector (R_1, R_2, R_3, R_4) , where $R_i \in A$ and $R_1 = Q_1, R_2 = Q_2, R_3 = Q_3, R_4 = Q_4$.

In other words, if C_j is the winning CoD vector for the resources in cluster 'i', then CoC of cluster 'i' is equal to C_j .

3.2.2. Dynamic cluster allocation

Once clusters are formed, their allocation is a simple process. Initially, a cluster with $\text{CoC} = \text{CoD}$ formed by big data stream is selected for allocation. If the selected cluster has enough resources to

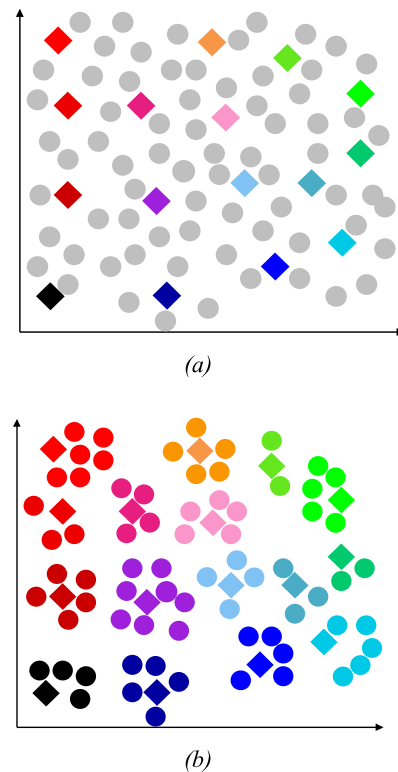


Fig. 5. Dynamic cluster formation using SOM. (a) Topological ordered output vectors; (b) Clustering of input vectors such that topological ordering is preserved.

process the stream, then it is allocated. Otherwise, a nearest topological ordered cluster having enough resources is searched and allocated. Such a scheme allows the streams to avoid waiting for the respective cluster to get free. Therefore, the waiting time is reduced. Here, it can be noted that whole of the big data stream converge at the same allocated cluster.

Procedure 1 Dynamic clustering by SOM.

1. Initialize weights from inputs to outputs to a small random value.
2. Assign value slightly less than 1 to the learning factor η .
3. Repeat steps 3 to 9 while computation bounds are not exceeded.
4. For each input vector S_i , repeat steps 6 to 8.
5. For each output neuron j , calculate square of Euclidean distance of S as

$$D(j) = \sum_{k=1}^q (S_{ik} - W_{jk})^2$$

6. Select J such that $D(J)$ is minimum.
7. Set $\text{CoC}(S_i) = \text{CoD}(C_J)$
8. Update weights of all topological neighbors of J such that

$$W_{jk}(t+1) = (1 - \eta(t))W_{jk}(t) + \eta(t)S_k$$

9. Decrement η monotonically
10. Output the virtual clusters with their respective CoC.

3.3. Workflow of proposed method

The detailed workflow of proposed system is shown by horizontal swim lane diagram in Fig. 6. As shown in diagram, the system executes its two steps in close co-ordination in following three conditions.

- Arrival of a new stream: If a new stream arrives in the system, its CoD is calculated. A cluster is selected such that $\text{CoC} = \text{CoD}$. If the selected cluster is not free, a nearest free topo-

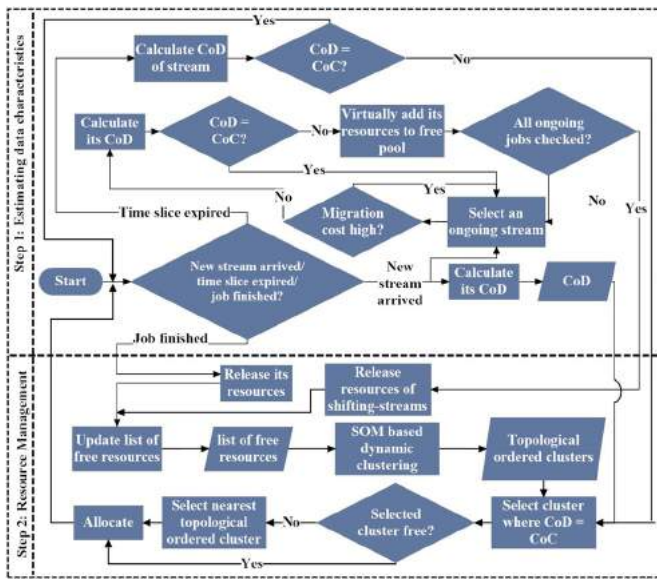


Fig. 6. Swim lane diagram of proposed system.

logical ordered cluster is allocated. In addition, CoD of all the other streams with low migration cost [38] is calculated again. The streams whose CoD = CoC continue to run on the same cluster. On the other hand, resources of streams whose CoD ≠ CoC (called shifting-streams) are virtually added to free pool list. Note that, here, resources are not actually released by shifting-streams until CoD of all the streams is calculated. This reduces waiting time. Once CoD of every stream in the system is calculated, list of free resources is updated clusters are formed with the help of SOM. The newly created clusters are allocated to shifting-streams as described in Section 3.2.2.

- On Expiry of time period ‘r’: As illustrated in Fig. 1, the CoD is calculated after ‘r’ time units. The stream is migrated to appropriate cluster if migration cost is low.
- Completion of any ongoing job: There are other jobs running on a cloud data center in addition to the streams. On completion of any such ongoing job, its resources are released and added to the free pool. These resources are then used during next cluster formation process.

Therefore, the proposed system allocates appropriate resources to big data stream based on its 4Vs. It migrates the stream to a new cluster whenever required for reducing execution time.

4. Experimental analysis

This section evaluates the proposed system experimentally. The two steps of the proposed system are evaluated separately in Section 4.1 and 4.2 respectively.

4.1. Experimental evaluation of Step 1 of proposed system

In order to generate a big data stream, four datasets are initially taken. The first dataset [44] is an image dataset consisting of 68,040 images. The second dataset [45] is an audio set which is a collection of 1,059 music tracks. The third dataset [46] is a surveillance video dataset consisting 29 hours of video. The fourth dataset [47] is a textual dataset consisting of 8000000 vocabulary words. Using these four datasets, a database is created such that:

- It consists of 14% image data, 23% of audio data, 24% of video data and 39% of text data.

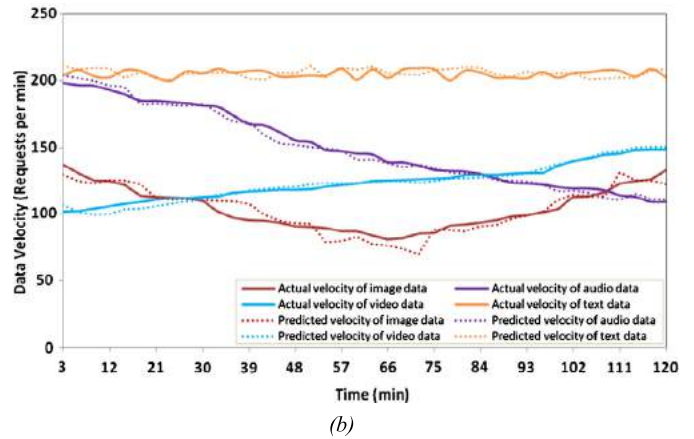
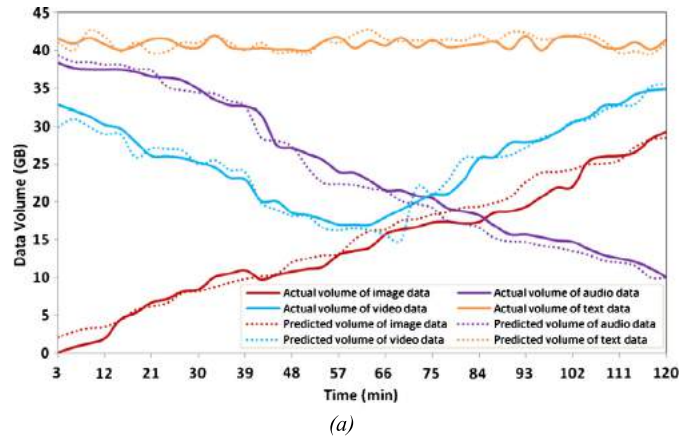


Fig. 7. Prediction accuracy of proposed system. (a) Predicting data volume; (b) Predicting data velocity.

- The image data is added in the database such that its volume is low initially and it increases towards the end of database (as shown by red colored line in Fig. 7(a)).
- The volume of audio data in the database is high initially and it decreases towards the end of database (as shown by purple colored line in Fig. 7(a)).
- The volume of video data in the database first decreases and then increases (as shown by blue colored line in Fig. 7(a)).
- The volume of text data remains almost constant throughout the database (as shown by orange colored line in Fig. 7(a)).

The created database is carefully fed to the system in form of a stream such that the velocities of image, audio, video and text data follow the following pattern:

- Velocity of image data first decreases and then increases with time (as shown by red colored line in Fig. 7(b)) such that it forms 18% of the overall velocity of the stream.
- Velocity of audio data decreases with time (as shown by purple colored line in Fig. 7(b)) such that it forms 26% of the overall velocity of the stream.
- Velocity of video data increases with time (as shown by blue colored line in Fig. 7(b)) such that it forms 21% of the overall velocity of the stream.
- Velocity of text data remains almost constant with time (as shown by orange colored line in Fig. 7(b)) such that it forms 35% of the overall velocity of the stream.

It can be noted here that the stream exhibits stochasticity in way that the volume and velocity of various data types varies with time. Therefore, a stream is generated such that actual volume and

Table 1
Mean absolute prediction error.

	Volume		Velocity	
	Volume %	MAPE	Velocity %	MAPE
Image data	14%	0.214	18%	0.235
Audio data	23%	0.183	26%	0.164
Video data	24%	0.176	21%	0.179
Text data	39%	0.115	35%	0.111

velocity at particular time instance is known to us. These actual values are compared with values predicted by the system. The prediction results are shown in Fig. 7(a)–(b). Here, the first step of proposed system, which predicts the volume and velocity of image, audio, video, and text data, is implemented in Matlab. The experiment uses block size of 128 MB, and a constant sampling rate of 0.33 samples/min (or one sample after every 3 minutes).

4.1.1. Discussion of results

It can be observed from Fig. 7(a)–(b) that the difference between actual and predicted values is low. In addition, Mean Absolute Prediction Error (MAPE) is calculated in each case which is summarized in Table 1.

It can be observed from Table 1 that MAPE decreases with the increase in data volume. For example, as explained in Section 4.1.1, the stream consists of 14% of image data, 23% of audio data, 24% of video data and 39% of text data. This implies that the volume of image data is lowest. On the other hand, the MAPE for predicting volume of image, audio, video and text data is 0.214, 0.183, 0.176, and 0.115 respectively. This implies that MAPE is highest for image data. Hence, MAPE is lower for higher data volume and increases with the decrease in data volume. Similar trend is observed in the case of data velocity. This is in accordance with the Law of Large Numbers which states that error decreases with the increase in data size.

It can be concluded from the above discussion that the proposed system efficiently predicts volume and velocity of image, audio, video, and text data. Moreover, since the CoD vector depends upon the predicted volume and velocity, therefore, the accuracy of CoD is implied.

4.2. Experimental evaluation of Step 2 of proposed system

This section compares the proposed system with a similar QoS based resource management system proposed by Sandhu and Sood [5]. For experimental analysis, 10 big data streams are generated. The procedure of stream generation is same as that explained in section 4.1. The percentage of volume and velocity is varied to generate these 10 different big data streams as summarized in Table 2. (Note that, in Table 2, I, A, V and T stands for image, audio, video and text data respectively). One stream is fed to the system after every ten minutes.

Virtual Machines (VMs) are selected from Amazon Elastic Compute Cloud (EC2) instances using QoS based and proposed system. VMs take the generated streams as input and run Alon–Matias–Szegedy (AMS) algorithm [48] on each stream. AMS is used to determine the frequencies of distinct elements in a stream. In an experiment of two hours, the resource utilization, resource availability of cloud resource is measured as shown in Fig. 8(a) and 8(b) respectively. In addition, the overall execution latency and response time of all the streams is aggregated and shown in Fig. 8(c) and 8(d) respectively.

4.2.1. Discussion of results

As stated in Section 2, in QoS based method proposed in [5], the processing power, GPU power, RAM and size of input data

Table 2
Percentage of volume and velocity in generated streams.

	Volume %				Velocity %			
	I	A	V	T	I	A	V	T
Stream 1	14	23	24	39	18	26	21	35
Stream 2	17	22	25	36	34	25	20	21
Stream 3	26	18	32	24	30	21	23	26
Stream 4	21	17	29	33	19	33	21	27
Stream 5	15	34	23	28	21	23	30	26
Stream 6	25	25	25	25	25	25	25	25
Stream 7	32	42	26	0	24	40	36	0
Stream 8	36	33	0	31	27	35	0	38
Stream 9	24	0	30	46	39	0	29	32
Stream 10	0	29	34	37	0	41	28	31

are provided by user. These user requirements are used to allocate resources to the incoming request. As stated in Section 1, user requirements depend upon data characteristics which are usually unknown to the user in case of big data streams. Therefore, in QoS based method, user may not be able to determine appropriate resources for big data stream. Moreover, the stream is run on same resources for whole time period. On the other hand, the proposed method predicts the 4Vs of the stream and determines the appropriate resources accordingly. The allocation is changed with the changing characteristics of data in a stream. It is due this fact that resource utilization is higher in case of proposed system as compared to QoS based method as shown in Fig. 8(a).

Furthermore, both QoS based and proposed system uses SOM for cluster allocation. In both the cases, a nearest topological ordered cluster is immediately allocated, if the selected cluster is not available due to which waiting time is reduced. Hence, there is not much difference in the resource availability as shown in Fig. 8(b).

Fig. 8(c) shows the comparison of execution latency between the QoS based and proposed system. In case of QoS based method, the execution latency remains almost same with only a slight increase towards the end of experiment. This is due to the fact that as more streams are added to the system, the probability of non-availability of required cluster increases (it may be already occupied by some other stream). Therefore, time is spent in finding the nearest topological ordered cluster. On the other hand, the proposed system shows higher execution latency in the beginning. This is because CoD for the stream is calculated before it can be allocated an appropriate cluster. Once appropriate cluster is allocated, the execution latency is decreases because suitable resources can process the stream at faster pace. Moreover, it is observed from Fig. 8(c) that there are periodic peaks after every 10 minutes since a new stream is added in the system after 10 minutes. CoD of new stream and all the other streams is calculated again which increases the execution latency. In spite of this, the overall execution latency of the proposed system is less than the QoS based method.

Fig. 8(d) shows the comparison of response time between the QoS based and proposed system. It can be observed that the response time for proposed system remains almost same throughout the experiment. This is because whenever characteristics of data in the stream changes, it is migrated to a more suitable cluster in the proposed system. This leads to a constant response time. On the other hand, response time in QoS based method increases with time since the stream is run on same resources for whole time period irrespective of the data characteristics.

5. Conclusion

In this paper, an efficient resource allocation system for big data streams is proposed. The proposed system allocates resources to stream based upon its data characteristics. The allocation is

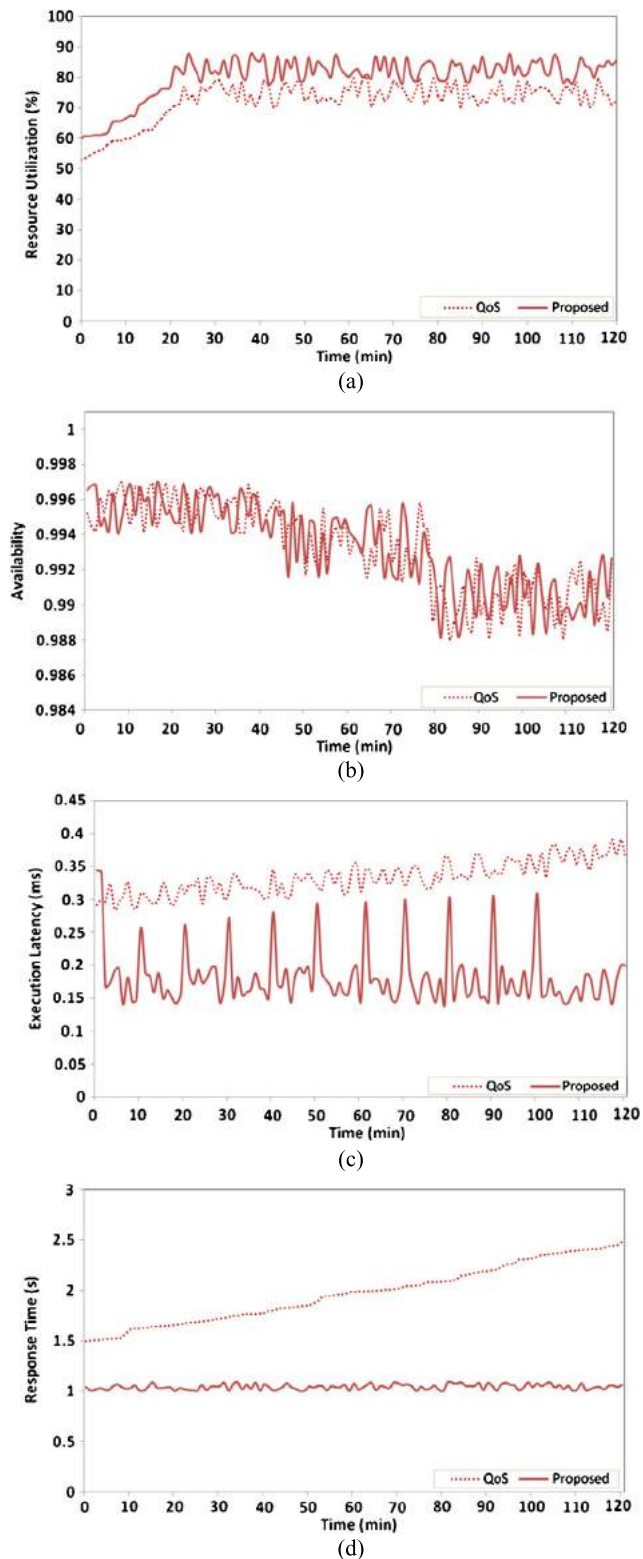


Fig. 8. Performance comparison of QoS based system and proposed system. (a) Comparison of resource utilization; (b) Comparison of resource availability; (c) Comparison of execution latency; (d) Comparison of response time.

changed whenever the data characteristics change. This mechanism results in a constant response time for a stream. Moreover, the waiting time for stream is reduced by topological ordering of clusters formed by SOM. The experimental results show that the proposed system shows a performance edge over other similar technique.

References

- [1] Gartner Inc., What is big data?, in: Gartner IT Glossary, 2013 [Online]. Available <http://www.gartner.com/it-glossary/big-data> (accessed 12 December 2016).
- [2] N. Kaur, S.K. Sood, An energy-efficient architecture for the Internet of Things (IoT), *IEEE Syst. J.* (2015) 1–10.
- [3] I.A.T. Hashem, I. Yaqoob, N. Badrul Anuar, S. Mokhtar, A. Gani, S. Ullah Khan, The rise of 'Big Data' on cloud computing: review and open research issues, *Inf. Syst.* 47 (2015) 98–115.
- [4] B.G. Batista, C. Henrique, G. Ferreira, D. Costa, M. Segura, D. Machado, L. Filho, M.L. Maciel, A QoS-driven approach for cloud computing addressing attributes of performance and security, *Future Gener. Comput. Syst.* 68 (2017) 260–274.
- [5] R. Sandhu, S.K. Sood, Scheduling of big data applications on distributed cloud based on QoS parameters, *Clust. Comput.* 18 (2) (2014) 817–828.
- [6] Z. Zheng, X. Wu, Y. Zhang, M.R. Lyu, J. Wang, QoS Ranking prediction for cloud services, *IEEE Trans. Parallel Distrib. Syst.* 24 (6) (2013) 1213–1222.
- [7] N. Ammu, M. Irfanuddin, Big data challenges, *Int. J. Adv. Trends Comput. Sci. Eng.* 2 (1) (2013) 613–615.
- [8] S. Pumma, T. Achalakul, L. Xiaorong, Automatic VM allocation for scientific application, in: *Proceedings of the International Conference on Parallel and Distributed Systems, ICPADS, 2012*, pp. 828–833.
- [9] S. Singh, I. Chana, Cloud resource provisioning: survey, status and future research directions, *Knowl. Inf. Syst.* 49 (3) (2016) 1005–1069.
- [10] S. Singh, I. Chana, Q-aware: quality of service based cloud resource provisioning, *Comput. Electr. Eng.* 47 (2015) 138–160.
- [11] J. Zhang, H. Huang, X. Wang, Resource provision algorithms in cloud computing: a survey, *J. Netw. Comput. Appl.* 64 (2016) 23–42.
- [12] J. Rao, Y. Wei, J. Gong, C.Z. Xu, QoS guarantees and service differentiation for dynamic cloud applications, *IEEE Trans. Netw. Serv. Manag.* 10 (1) (2013) 43–55.
- [13] W.-J. Wang, Y.-S. Chang, W.-T. Lo, Y.-K. Lee, Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments, *J. Supercomput.* 66 (2) (2013) 783–811.
- [14] Z. Zhu, S. Li, X. Chen, Design QoS-aware multi-path provisioning strategies for efficient cloud-assisted SVC video streaming to heterogeneous clients, *IEEE Trans. Multimed.* 15 (4) (2013) 758–768.
- [15] W.-H. Hsu, C.-H. Lo, QoS/QoE mapping and adjustment model in the cloud-based multimedia infrastructure, *IEEE Syst. J.* 8 (1) (2014) 247–255.
- [16] J.M. Chang, QoS-aware data replication for data-intensive applications in cloud computing systems, *IEEE Trans. Cloud Comput.* 1 (1) (2013) 101–115.
- [17] S. Misra, S. Das, M. Khatua, M.S. Obaidat, QoS-guaranteed bandwidth shifting and redistribution in mobile cloud environment, *IEEE Trans. Cloud Comput.* 2 (2) (2014) 181–193.
- [18] K.T. Chen, Y.C. Chang, H.J. Hsu, D.Y. Chen, C.Y. Huang, C.H. Hsu, On the quality of service of cloud gaming systems, *IEEE Trans. Multimed.* 16 (2) (2014) 480–495.
- [19] S.K. Sood, Function points-based resource prediction in cloud computing, *Concurr. Comput. Pract. Exp.* 28 (2016) 2781–2794.
- [20] S.K. Sood, R. Sandhu, Matrix based proactive resource provisioning in mobile cloud environment, *Simul. Model. Pract. Theory* 50 (2015) 83–95.
- [21] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, J. Kołodziej, J. Kołodziej, Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing, *Future Gener. Comput. Syst.* 51 (2015) 61–71.
- [22] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H.S.-H. Chung, Y. Li, Cloud computing resource scheduling and a survey of its evolutionary approaches, *ACM Comput. Surv.* 47 (4) (2015) 1–33.
- [23] A. Sfrant, F. Pop, Asymptotic scheduling for many task computing in Big Data platforms, *Inf. Sci. (Ny)* 319 (2015) 71–91.
- [24] Agile data integration platforms – cloud-based (iPaaS) and on-premise software [Scribe software, [Online]. Available <http://www.scribesoft.com/> (accessed 9 December 2016).
- [25] C. Olston, S. Seth, C. Tian, T. ZiCornell, X. Wang, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V.B.N. Rao, V. Sankarasubramanian, Nova, in: *Proceedings of the 2011 International Conference on Management of Data, SIGMOD '11, 2011*, pp. 1081–1082.
- [26] P. Bhatotia, A. Wieder, R. Rodrigues, U. a. Acar, R. Pasquin, Incoop: MapReduce for incremental computations, in: *Proceedings of the 2nd ACM Symp. Cloud Comput., SOCC '11, 2011*, pp. 1–14.
- [27] L. Neumeyer, B. Robbins, A. Nair, A. Kesari, S4: distributed stream computing platform, in: *2010 IEEE International Conference on Data Mining Workshops, 2010*, pp. 170–177.
- [28] Apache storm, [Online]. Available <http://storm.apache.org/> (accessed 9 December 2016).
- [29] Welcome to apache flume – apache flume, [Online]. Available <http://flume.apache.org/index.html> (accessed 9 December 2016).
- [30] D. Sun, G. Zhang, S. Yang, W. Zheng, S.U. Khan, K. Li, Re-stream: real-time and energy-efficient resource scheduling in big data stream computing environments, *Inf. Sci. (Ny)* 319 (2015) 95–112.
- [31] R. Tolosana-Calasan, J.Á. Bañares, C. Pham, O.F. Rana, Resource management for bursty streams on multi-tenancy cloud environments, *Future Gener. Comput. Syst.* 55 (2016) 444–459.

- [32] M. Rahman, P. Graham, Responsive and efficient provisioning for multimedia applications, *Comput. Electr. Eng.* 53 (2016) 458–468.
- [33] Q. Zhang, Z. Chen, L.T. Yang, A nodes scheduling model based on Markov chain prediction for big streaming data analysis, *Int. J. Commun. Syst.* 28 (9) (2015) 1610–1619.
- [34] A. Castiglione, R. Pizzolante, A. De Santis, B. Carpentieri, A. Castiglione, F. Palmieri, Cloud-based adaptive compression and secure management services for 3D healthcare data, *Future Gener. Comput. Syst.* 43–44 (2015) 120–134.
- [35] J. Peng, X. Zhi, X. Xie, Application type based resource allocation strategy in cloud environment, *Microprocess. Microsyst.* (2016), <http://dx.doi.org/10.1016/j.micpro.2016.09.014>.
- [36] A.K. Baughman, R.J. Bogdany, C. McAvoy, R. Locke, B. O'Connell, C. Upton, Predictive cloud computing with big data: professional golf and tennis forecasting [application notes], *IEEE Comput. Intell. Mag.* 10 (3) (2015) 62–76.
- [37] A. Jain, E.Y. Chang, Adaptive sampling for sensor networks, in: *Proceedings of the 1st International Workshop on Data Management for Sensor Networks in Conjunction with VLDB 2004, DMSN '04, 2004*, pp. 10–14.
- [38] W. Voorsluys, J. Broberg, S. Venugopal, R. Buyya, Cost of virtual machine live migration in clouds: a performance evaluation, in: *Lecture Notes in Computer Science*, in: LNCS, vol. 5931, 2009, pp. 254–265.
- [39] A. Gandomi, M. Haider, Beyond the hype: big data concepts, methods, and analytics, *Int. J. Inf. Manag.* 35 (2015) 137–144.
- [40] J. Hirschberg, A. Hjalmarsson, N. Elhadad, 'You're as sick as you sound': using computational approaches for modeling speaker state to gauge illness and recovery, in: *Advances in Speech Recognition*, Springer US, Boston, MA, 2010, pp. 305–322.
- [41] H.A. Patil, 'Cry Baby': using spectrographic analysis to assess neonatal health status from an infant's Cry, in: *Advances in Speech Recognition*, Springer US, Boston, MA, 2010, pp. 323–348.
- [42] A. Rajaraman, J.D. Ullman, Bloom filter, in: *Mining of Massive Datasets*, first edit, Cambridge University Press, 2014, pp. 116–118.
- [43] G. Deodatis, M. Shinozuka, Auto-regressive model for nonstationary stochastic processes, *J. Eng. Mech.* 114 (11) (1988) 1995–2012.
- [44] UCI machine learning repository: corel image features data set, [Online]. Available <https://archive.ics.uci.edu/ml/datasets/Corel+Image+Features> (accessed 13 December 2016).
- [45] UCI machine learning repository: geographical original of music data set, [Online]. Available <https://archive.ics.uci.edu/ml/datasets/Geographical+Original+of+Music> (accessed 13 December 2016).
- [46] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J.T. Lee, S. Mukherjee, J.K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, M. Desai, A large-scale benchmark dataset for event recognition in surveillance video, in: *CVPR 2011, 2011*, pp. 3153–3160.
- [47] UCI machine learning repository: bag of words data set, [Online]. Available <https://archive.ics.uci.edu/ml/datasets/Bag+of+Words> (accessed 13 December 2016).
- [48] A. Rajaraman, J.D. Ullman, Estimating moments, in: *Mining of Massive Datasets*, first edit, Cambridge University Press, 2014, pp. 122–127.