Received: 29 September 2016 | Revised: 9 March 2017 | Accepted: 17 March 2017

DOI: 10.1002/nem.1978

SPECIAL ISSUE PAPER

WILEY

Dynamic resource allocation for big data streams based on data characteristics (5Vs)

Navroop Kaur | Sandeep K. Sood

Guru Nanak Dev University Regional Campus Gurdaspur, Gurdaspur, Punjab, India

Correspondence
Navroop Kaur, Guru Nanak Dev University Regional Campus Gurdaspur, Gurdaspur, Punjab, India.
Email: navonline98@gmail.com

Summary

Various Internet-based applications such as social media, business transactions, mobile applications, cyber-physical systems, and Internet of Things have led to the generation of big data streams in every field. The growing need to extract knowledge from big data streams has pioneered the challenge of selecting appropriate cloud resources. The current techniques allocate resources based on data characteristics. But because of the stochastic 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. Working towards this direction, this paper proposes a system that predicts the data characteristics in terms of volume, velocity, variety, variability, and veracity. The predicted values are expressed in a quadruple called Characteristics of Big data (CoBa). Thereafter, the proposed system uses self-organizing maps to dynamically create clusters of cloud resources. One of these clusters is allocated to the big data stream based on its CoBa. The proposed system is dynamic in the sense that it changes the cloud cluster allocated to big data stream if its CoBa changes. Experimental results show that the proposed system has a performance edge over other streaming data processing tools such as Storm, Flume, and S4.

1 | INTRODUCTION

A collection of huge volumes of diverse types of structured and unstructured data that cannot be handled by state-of-the-art data processing platforms is termed as big data. The size of big data may span over multiple terabytes, petabytes, or even zettabytes. In a survey conducted by IBM,[1] more than half of 1144 respondents believed that datasets of size over 1 TB qualify as big data. However, defining specific thresholds for big data volume is rather impractical since it is a relative term.[2] The data volume that defines big data today may not meet the threshold in future owing to enhanced storing capacities allowing data in higher volumes to be captured easily. In addition, an unstructured dataset of certain size may be considered as big data while a structured dataset of same size may not qualify as big data. This implies that the definition of big data has gone beyond the realm of data volume to some other defining characteristics such as variety and velocity.[3] Variety refers to heterogeneity of data types. The text, image, audio, and video data captured by sensors, actuators, business transactions, social networks, and smart environments contribute to data variety. Most of this data are in unstructured format.[4] Velocity refers to unprecedented speed of data generation and data analysis. There is an interesting interplay between volume and velocity. For example, data collected and transmitted by a spacecraft after every 6 seconds is considered as big data because of its huge volume.[5] On the other hand, small amount of data generated by thousands of tweets per second is also considered as big data because of its high velocity. Due to such considerations, there are no specific thresholds for data velocity.

Furthermore, in addition to volume, variety, and velocity (the 3 Vs), there are other dimensions of big data, namely, veracity, variability, value, and visualization. Veracity, a term

coined by IBM, implies the untrustworthiness inherent in some sources of data. Variability, introduced by SAS, refers to variation in data flow rate. Value, introduced by Oracle, refers to the process of extracting knowledge from big data. Visualization is the representation of results obtained from big data in the form of charts and graphs.

Big data has emerged as a widely recognized trend. The main sources of big data include social media, business transactions, mobile applications, cyber-physical systems, and Internet of Things (IoT).[6-8] These applications generate continuous streams of big data across the globe. Here, it can be noted that there are 2 main situations where data are considered as streaming data, firstly, when it flows too fast to store in its entirety, and, secondly, when the application mandates immediate response to data. For example, social media, business transactions, and mobile applications generate data at a very high velocity. Facebook handles nearly 1 million photographs per second; twitter produces 6000 tweets per second on an average,[9] and Wal-Mart handles more than 1 million transactions per second.[4] On the other hand, immediate response is required on rich spatiotemporal information acquired by advanced sensors of IoT network and cyber-physical systems. With respect to this, data generated by all the above mentioned major sources of big data qualify as big data streams.

The distillation of knowledge from big data streams is essential as well as a difficult task. It is beyond the capabilities of traditional systems and requires cloud resources.[2] With the growth of big data on cloud,[7] selecting optimal number of cloud resources for each incoming request emerged as a key challenge. Moreover, the requirement of real-time analysis of big data streams further enhances the necessity of selecting appropriate resources. Working towards this direction, it is posited that there 2 approaches for selecting appropriate resources. The first approach allows users to select the resources. For such an allocation, user must be acquainted with the characteristics of data. For example, 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. Consequently, such resource selection is limited by the expertise of user. Moreover, even if the user possesses enough expertise, knowledge of data characteristics is necessary. Nevertheless, due to stochastic nature of data generation by various sources, characteristics of incoming big data stream are generally unknown. This fact makes the manual selection of cloud resources relatively unsuitable for big data streams.

On the other hand, the second approach (the one used in proposed system) allows system to automatically select suitable cloud resources by predicting data characteristics. Here, the major question that arises is as follows: Which data characteristics should be predicted by the system? In 2015, Zhang, Chen, and Yang[10] argued that data volume plays a vital role in efficient selection of cloud resources for big data streams. On the other hand, Baughman et al[11] illustrated that predicting velocity of incoming big data request is crucial for efficient provisioning of cloud resources. Furthermore, Castiglione et al[12] emphasized that variation in data flow rate (variability) affects cloud resource allocation. In 2016, Peng et al[13] accentuated the importance of data variety for selecting appropriate cloud resources. In addition, in 2017, Rehman et al[14] found that volume, velocity, variety, variability, and veracity (5Vs) are essential parameters for efficiently mining data streams. From these studies, it is concluded these 5Vs of big data streams are driving parameters for cloud resource allocation. To the best of our knowledge, none of the existing approaches uses all the 5Vs for selecting optimal number of cloud resources. Hence, this paper proposes an automated system that predicts the 5Vs of big data stream and later uses the predicted values for dynamic resource allocation to big data streams.

For predicting the 5Vs, the system initially filters out the data from unreliable sources (veracity) and separates various data varieties with the help of Bloom filter.[15] Thereafter, it uses Kalman filter[16] to estimate the volume and velocity of each data variety arriving at the system in next interval. Variability of data is incorporated while estimating volume and velocity. Later, self-organizing map (SOM)[17] is used for dynamic resource allocation based on the predicted 5Vs. Here, it is worth mentioning that proposed method prefers Kalman filter over other prediction solutions since it can be easily tailored to provide unbiased estimations on a wide range of data streams even when the variance is high. On the other hand, SOM is used for clustering since it can easily and dynamically create clusters by creating reference vectors in topological ordered fashion such that more is the relationship between 2 reference vectors, closer is their topological ordering. The proposed method takes advantage of such topological ordering of SOM to reduce waiting time.

The scientific contribution of this paper is two-fold. Firstly, the proposed system efficiently predicts the aforementioned 5Vs of big data stream, which have not been proposed hitherto. Secondly, it presents a unique approach to determine and allocate appropriate resources for stream analysis based on data characteristics. Furthermore, the resource allocation technique proposed here is dynamic and adaptive.

This paper is organized into 5 sections. Section 2 gives the preliminary knowledge of concepts used in proposed system. Section 3 discusses the working of proposed system. Section 4 provides experimental setup, results, and discussion. Finally, Section 5 concludes the paper.

2 | PRELIMINARIES

This section first presents the work related to the field of big data and cloud resource management. Later, it gives the preliminary knowledge of the Bloom filter, Kalman filter, and SOM that are used in proposed system.

2.1 | Related work

Literature finds various attempts to define big data.[2] The definition of big data includes the explanation of its essential characteristics (the 7 Vs). Yaqoob et al[8] identified the origin of the term big data and its various sources. They found that the growth of data from applications such as social networks,[18] business transactions,[19] mobile applications,[20] cyber-physical systems,[21] and IoT[22] have sparked big data era. With such an upsurge, big data analytics is gaining momentum. Various authors summarized the state-of-the-art big data analytical technologies and techniques.[8,23,24] They highlighted the role of cloud computing in big data analytics. Supporting this viewpoint, Yang et al[6] stated that cloud computing provides fundamental support to big data analytics by leveraging various computation and storage resources on demand. Furthermore, Hashem et al[7] posited that big data applications are increasing on cloud. They presented the fact that selecting appropriate techniques and cloud resources for analyzing big data is a major research problem. Working towards this direction, Pumma, Achalakul, and Li[25] argued that suitable amount of cloud resources should be determined prior to the start of execution. Consequently, the field of predicting and provisioning cloud resources[26-28] gained momentum. Nevertheless, the work on big data application scheduling is still sparse.

Dynamic resource provisioning for big data applications is a challenging issue.[29] There are various research directions for real-time, distributed, dynamic, adaptive, and multiobjective scheduling of big data applications as identified by Zhan et al.[30] Sfrent and Pop[31] accentuated that big data applications are greatly affected by the resource scheduling algorithm. They studied various scheduling algorithms and found that best resource scheduling algorithm can be discovered under certain conditions. In 2014, Sandhu and Sood[32] proposed a quality of service–based framework to schedule big data applications over distributed cloud.

With the ubiquity and prevalence of Internet-based applications, big data started flowing in continuous streams. This led to the emergence of stream-processing paradigm. The challenge of deriving insights from big data streams is recognized as a key challenge. Various authors proposed various algorithms and techniques for mining big data streams.[13,14,33-38] The proposed algorithms illustrated various techniques to collect, integrate, analyze, and visualize big data streams in real time. Nevertheless, less attention has been paid to resource scheduling for big data streams. Sun et al[39] proposed a graph-based method that manages the resources held by stream-processing tools in an energy-efficient way. Furthermore, Tolosana-Calasanz et al[40] proposed a method to processes data streams such that the profit of cloud provider is maximized. In 2016, Rahman and Graham[41] developed a priority-based method for multimedia data processing. But the methods proposed in Tolosana-Calasanz et al[40] and Rahman and Graham[41] are static hybrid algorithms. Zhang et al[10] efficiently predicted the big data volume using Markov chain and assigned nodes for data processing accordingly. But the model does not consider the other Vs of big data for resource allocation that are important parameters for every incoming data request.

2.2 | Bloom filter

Bloom filter[15] is probabilistic data structure that is used to filter out the elements that do not belong to a set. It consists of a bit-array of m elements, a set "S"of allowable "key" values and a collection of hash functions. MD5 hash algorithms are the most commonly used hash functions in Bloom filter. The working of Bloom filter involves 2 steps. The first step is meant to insert "key" values in the filter. This step starts by setting all the bits in the array to 0. A "key" value is hashed by all the hash functions. All the bit elements to which the "key" value hashes is set to 1. This process is repeated for all "key" values. Hence, bit-array, with some of the bits set to 1, is the resultant of step 1. The second step calculates hash values of the element arriving at filter. This element is allowed to pass only if all the bits to which its hashes are set to 1. Therefore, Bloom filter allows all the elements whose keys are in S, while rejecting most of the elements whose keys are not in S. The major strengths of Bloom filter are its space-efficiency, speed, and constant complexity for adding and filtering elements. These features make it suitable for streaming data. Therefore, it is used in the proposed system for filtering streams such that most non-member elements of the stream are deleted that guarantees veracity of data. Moreover, it never reports false negative. The element that it reports to be not of the set is "definitely not" in the set. On the other hand, it is subject to false positives. It can claim an element to be part of set when it is not. Nevertheless, this false positive rate does not impact its use in the proposed system because its space-efficiency and speed outweigh the false positive probability.

2.3 | Kalman filter

Kalman filter,[16] introduced by Rudolph E. Kalman in 1960, is an efficient recursive filter. It is a mathematical toolbox capable of dynamically predicting the future trends from

incoming streams of noisy sensor measurements. For predicting the future trends, it uses a set of mathematical equations called predictor-correction equations. The predictor equations are responsible for a priori estimate of the parameter(s) under consideration. These equations project forward the state of system and error covariance to obtain the required a priori estimate. The corrector equations are meant for providing feedback to the predictor equations. These equations incorporate new measurement of the parameter(s) under consideration into the a priori estimate to improve future estimates, thereby, minimizing error variance. The predictor-corrector equations work recursively to capture current state of the system and to optimally predict the future trends. Here, prediction is optimal in the sense that the estimated error covariance is minimized. This feature makes it suitable for a large class of prediction and forecasting problems.

There are various advantages of Kalman filter. Firstly, the mathematical equations of Kalman filter are not hard to compute that makes it easy to use. Secondly, the filter works efficiently even in the absence of optimal conditions. Thirdly, it can predict the values accurately even in the presence of noise. Fourthly, it not only provides an optimal estimation of received parameters but also predicts hidden parameters of the incoming stream from the available data. For example, let Kalman filter receives a stream of noisy observations about the position of an object. It can accurately predict the position as well as velocity of that object in future. Lastly, it can efficiently capture spatiotemporal correlations of data that makes it suitable for time-series analysis of a system. Therefore, the mathematical power of Kalman filter is used to predict the volume and velocity of big data streams by capturing spatiotemporal correlations (variability) of data.

2.4 | Self-organizing maps

Self-organizing map[17] is one of the unsupervised neutral network-learning techniques that can be used during exploratory phase of data mining. For exploring data properties, it projects higher dimensional data into lower dimensional (usually 2-D) grids. Such projection helps in efficient visualization of data and in cluster analysis. In the proposed system, SOM is used for clustering. During clustering, SOM transforms input data to vectors that forms input layer of neural network. On the other hand, it finds a set of centroids of each cluster. Each centroid is associated with a neuron. These neurons form the output layer of neural network. The input vectors are assigned to the cluster that provides best approximation centroid.

Unlike other clustering techniques, SOM imposes neighborhood relations on the centroids. This implies that the clusters that are closer are more related to one another than the clusters that are far away. Therefore, when an input vector is processed, not only the best approximation centroid but also the neighboring centroids are updated. The processing of vectors stops only when centroids do not change further. It is due to this feature of SOM that it is classified as unsupervised learning technique. After completion of processing, clusters are formed. Due to neighborhood relations of centroids, the output clusters exhibit topological ordering. The proposed system takes advantage of such topological ordering to minimize waiting time.

3 | PROPOSED SYSTEM

The proposed system aims to allocate appropriate resources to big data stream based on its volume, velocity, variety, veracity, and variability. The overall working of proposed system is shown in Figure 1, where a big data stream is analyzed on cloud to obtain required output. Here, it can be noted that big data stream consists of a variety of data elements. As stated earlier, big data stream is generated from various sources such as social media, business transactions, cyber-physical systems, and IoT. The data from these sources can be in the form of text (such as emails, news, social network feeds, numeric values, sensor measurements, HTML, XML, JSON, postscript, log files, etc), images, audio, and video. Various big data analytical algorithms are used for unlocking potential intelligence from these data. There are mainly 4 types of big data analytics,[2] namely, text analytics, image analytics, audio analytics, and video content analytics. On the basis of the type of analytics used, it can be said that big data stream consists of 4 types of data elements, namely, text data elements, image data elements, audio data elements, and video data elements as shown in Figure 1.

For analyzing stream consisting of various data elements on cloud, appropriate resources are selected by the proposed system. To accomplish this goal, the proposed system uses 2 modules, namely, workload estimator (WEst) and cloud resource manager (CRM). Workload estimator initially extracts small chunks of data from incoming stream, as shown in Figure 1, using an adaptive sampling technique.[42] Taking smaller chunks not only enables efficient forecasting but also reduces calculation. On the other hand, the adaptive sampling allows the system to adjust the sampling rate autonomously according to the characteristics of streaming data. The selected data chunk is analyzed by WEst to estimate the workload that is expected to arrive during the next time interval. The estimated workload, expressed in a quadruple called Characteristics of Big data (CoBa), is passed as input to CRM as shown in Figure 1. Cloud resource manager uses CoBa to dynamically create and allocate clusters of free cloud resources using SOM. This process is repeated after "t" time units, where "t" is the time slice whose value depends upon the sampling rate.

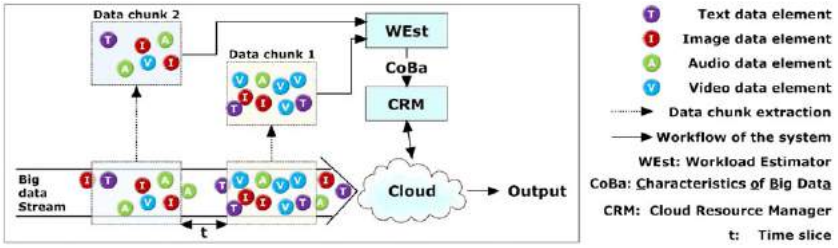

FIGURE 1 Overview of proposed method

The detailed working of the 2 modules is given in Sections 3.1 and 3.2, respectively. Section 3.3 presents a detailed flowchart of the proposed system and explains how WEst and CRM works in close coordination to achieve the required objectives.

3.1 | Workload estimator

For estimating the workload that will arrive in next time interval, WEst works in 3 steps as shown in Figure 2. The first step takes the extracted data chunk as input and works on variety and veracity of big data stream. It uses 4 Bloom filters corresponding to each of the text, image, audio, and video data elements. The bloom filter for text data allows only text elements from trustworthy sources to pass through it, thereby providing text stream as output. Similar is the case for the other filters. This implies that the incoming big data stream is converted into 4 streams each carrying only 1 type of data elements. These resultant streams are called elementary streams. The second step is a MapReduce-based framework that uses Kalman filter to estimate volume and velocity of each of the elementary stream. Here, it is worth mentioning that data flow can be inconsistent with periodic, seasonal, or daily peaks and troughs. For example, when something is trending on social media, data flows at higher velocity. Such a variation in data flow rate (ie, variability) plays an important role in predicting volume and velocity. Consequently, Kalman filter estimates volume and velocity by taking variability into account.

The first 2 steps, therefore, accomplish the task of predicting the 5Vs of big data stream. The third step uses the estimated values of volume and velocity to calculate a term called data flow level (described in Section 3.1.3). The flow rates of text, image, audio, and video data are represented in CoBa. Hence, WEst takes data chunk as input and gives CoBa as output. The detailed explanation of these 3 steps is given below.

3.1.1 | Step 1: Determining variety and veracity

The goal of this step is to separate elementary streams and filter out the data elements from untrustworthy sources. Such a separation helps to determine first 2 Vs, ie, variety and veracity of big data stream. As stated earlier, 4 Bloom filters are used to accomplish this task. The overall working of these 4 Bloom filters is shown in Figure 3. Here, the extracted data chunk consists of various data elements. This data chunk is fed to all the 4 bloom filters. The bloom filter for text data allows only text element to pass through. Similar is the case for other filters. Moreover, it can be noted that the bloom filter for image data received 3 image elements but allowed only 1 to pass through. This is because the others are from unreliable sources. The filtered data elements enter into their respective buckets. These buckets are open-ended that allows elements to pass through the other end, forming respective elementary streams.

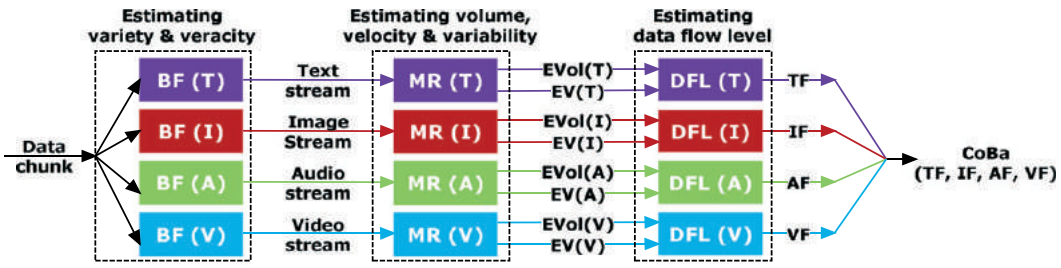

FIGURE 2 Workload prediction by workload estimator

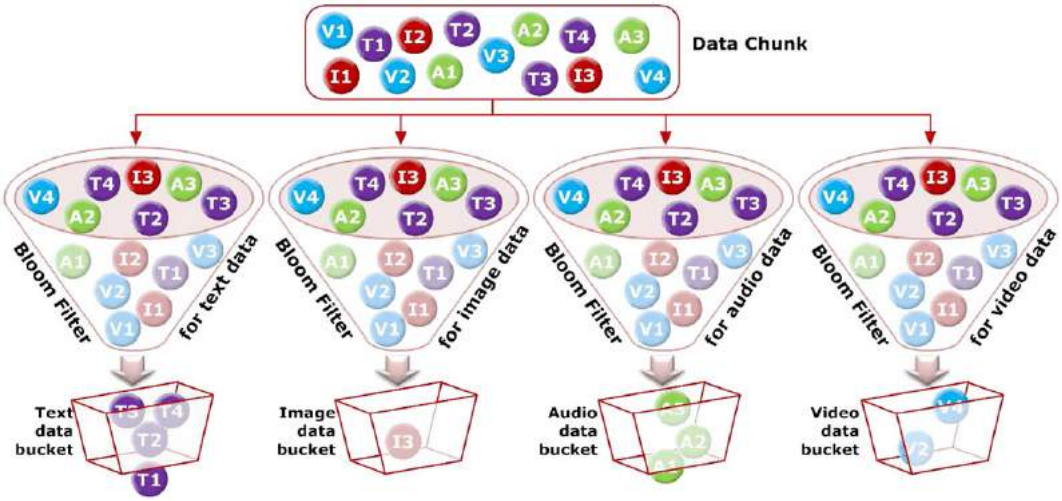

FIGURE 3 Predicting variety and veracity using Bloom filter

In the proposed system, each Bloom filter consists of

1. F: a set of all acceptable data formats (such as png, jpeg, etc for image data elements).
2. O: a set of all trustworthy sources.
3. $S = F \times O$: a set of key values.
4. $BF[m]$: a bit array of "m" elements such that $m = \left\lceil \frac{-|S|^* \ln(p)}{[\ln(2)]^2} \right\rceil$; where p is the acceptable false positive rate and $|S|$ denotes the cardinality of set S.
5. A collection of "H" hash functions $h_1, h_2, \ldots \ldots, h_H$ such that $H = \text{round}\left(\ln(2) * \frac{m}{|S|} \right)$.

Here, the sets F and O are application dependent and are defined as per requirements of the user. A set S of key values is constructed by taking Cartesian product of F and O. This implies that set S consists of a list of data formats from reliable sources. The data elements that belong to set S will be allowed to pass through the Bloom filter. Furthermore, "H" number of hash functions are randomly chosen. Moreover, all the bits in array BF[m] are initialized to 0. Thereafter, for every key $k_i \in S$, $h_1(k_i), \ldots \ldots, h_H(k_i)$ are calculated, and the corresponding bit, ie, $BF[h_j(k_i)]$ is set to 1. This process is called initialization of the filter. Once a Bloom filter is initialized, it is used to filter the data elements. For each stream element "e" arriving at the filter, hash values $h_1(e), \ldots \ldots, h_H(e)$ are calculated. If $BF[h_j(e)] = 1$ for all j, then the element is allowed to pass else it is blocked at the filter. In other words, filter allows the element "e" to pass only if it hashes to bit positions that are all set to one.

An example of initialization and working of Bloom filter is shown in Figure 4. Let there are 3 key values k_1, k_2, and k_3. The value of m and H are calculated by using $p = 0.1$. The resultant values are $m = 15$ and $H = 3$. Hence, an array of 15 bits is taken. The key values $k_1, k_2,$ and k_3 are hashed using

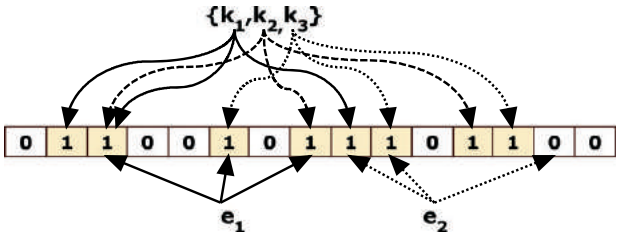

FIGURE 4 Example of working of Bloom filter

the 3 hash functions to respective positions in the array as shown by curved lines in Figure 4. Thereby, the Bloom filter is initialized such that 8 of 13 bits are set to 1. Later, when data element "e_1" arrives, the hash values are calculated using the 3 hash functions. It can be observed that all the bit positions of array to which "e_1" is hashed are set to one. Therefore, "e_1" is allowed to pass. On the other hand, "e_2" results in a hash value that is "0." So it is not allowed to pass through.

3.1.2 | Step 2: Estimating volume and velocity using variability

This step aims to estimate the volume and velocity of each elementary stream by taking variability on-board. It uses MapReduce-based framework to achieve the required objectives. Four such frameworks are used in the system, one each for the 4 elementary streams (shown in Figure 2). Working of all the 4 frameworks is same, except for the type of elementary stream input to it. Therefore, this section presents a generalized explanation of the framework that is applicable to all 4 of them.

The working of MapReduce-based framework is shown in Figure 5 where, initially, the elementary stream

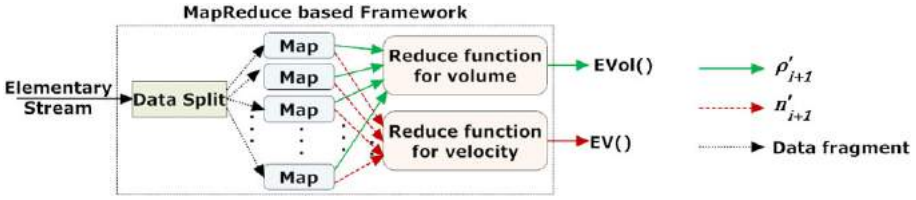

FIGURE 5 Overview of MapReduce-based framework

(from step 1) is randomly split across several map functions. The number of map functions varies according to the data arrival rate that allows the system to adjust data processing speed according to data arrival rate. Each map function estimates volume and velocity using Kalman filter. The estimated value of volume is sent to 1 reduce function while that of velocity to another reduce function. Each of the 2 reduce functions aggregate the received values to calculate overall volume and velocity. Since reduce function simply aggregates the values provided by map function, therefore, only 2 reduce functions can work effectively without creating bottleneck of the system. The estimated values of volume and velocity are denoted by *EVol*() and *EV*(), respectively. It can be noted here that *EVol*() and *EV*() are generalized notations applicable to all the 4 elementary streams. For example, estimated volume and velocity for text stream is represented by *EVol*(*T*) and *EV*(*T*). The details of map and reduce functions are as follows.

Map function: Map function executes the predictor-corrector equations of Kalman filter. These equations are implemented by every map function in all the 4 MapReduce based-frameworks on their respective elementary streams. The notations used in these equations are summarized in Table 1. Initially, predictor equations estimate volume, velocity, and error covariance of (i + 1)th data chunk before it actually arrives by using Equations 1 to 5.

$$\rho_{i+1}^{'} = \alpha_1\rho_i + \alpha_2\rho_{i-1} + \ldots\ldots\ldots + \alpha_q\rho_{i-q+1} \tag{1}$$

where

$$\alpha_j = \frac{covariance\ (\rho_i, \rho_{i-j})}{variance\ (\rho_j)}. \tag{2}$$

$$n_{i+1}^{'} = \beta_1 n_i + \beta_2 n_{i-1} + \ldots\ldots\ldots + \beta_q n_{i-q+1} \tag{3}$$

where

$$\beta_i = \frac{covariance\ (n_i, n_{i-j})}{variance\ (n_j)}. \tag{4}$$

$$\Omega_{i+1}^{'} = \Omega_i + Q \tag{5}$$

In Equation 1, the variability of data is captured by α, which denotes the corelation between 2 values. Let α is high for ρ_i and ρ_{i-x}. This implies that ρ_i is similar to ρ_{i-x}, then it is assumed that ρ_{i+1} will be similar to ρ_{i-x+1} because of periodic trends. Hence, for identifying periodic trends (or variability), the corelation of ρ_i with "q" preceding values is calculated. Here, "q" depends upon the time period for which variability is considered. For example, to capture daily peaks and troughs, variability of a single day is considered. In such a case, q will be equal to number of data chunks extracted by the system in 1 day. This implies that Equation 1 estimates the volume of (i + 1)th data chunk by taking variability into consideration. Similar is the case for velocity calculation in Equation 3. Furthermore, in Equation 5, Q is estimated with the help of recursive covariance estimation algorithm.[43]

Thereafter, corrector equations (given by Equations 6-9) modify these estimated values after the arrival of (i + 1)th data chunk. Equation 6 calculates a factor called Kalman gain using estimated error covariance, $\Omega_{i+1}^{'}$, measurement noise covariance, R, where R is given by recursive covariance estimation algorithm.[43] The corrected values obtained from the corrector equations are later used by predictor equations

TABLE 1 Notations used in predictor-corrector equations

Notation	Brief Meaning	Notation	Brief Meaning
$\rho_i^{'}$:	Estimated volume of ith data chunk	**Q:**	Process noise covariance
ρ_i:	Corrected volume of ith data chunk	**R:**	Measurement noise covariance
$n_i^{'}$:	Estimated velocity of ith data chunk	y_i:	ith measurement of volume
n_i:	Corrected velocity of ith data chunk	z_i:	ith measurement of velocity
$\Omega_i^{'}$:	Predicted error covariance of ith data chunk	K_i:	Kalman gain at ith prediction step
Ω_i:	Corrected error covariance of ith data chunk		

for estimating volume and velocity of (i + 2)th data chunk. This implies that predictor-corrector equations work in a continuous loop for every data chunk. The overall working of map function using predictor-corrector equations is summarized in Figure 6.

$$K_{i+1} = \frac{\Omega'_{i+1}}{\Omega'_{i+1} + R} \tag{6}$$

$$\rho_{i+1} = \rho'_i + K_i\left(y_i - \rho'_i\right) \tag{7}$$

$$n_{i+1} = n'_{i+1} + K_i\left(z_i - n'_i\right) \tag{8}$$

$$\Omega_{i+1} = (1 - K_{i+1})\Omega'_{i+1} \tag{9}$$

Reduce function: As stated earlier, there are 2 reduce functions, one each for volume and velocity. The reduce function for volume estimation receives the value of ρ'_{i+1} from all the map functions used in MapReduce-based framework as shown in Figure 5. It calculates the average, $Avg\left(\rho'_{i+1}\right)$, of all these values. Subsequently, $Avg\left(\rho'_{i+1}\right)$ is compared with mean, $\mu(\rho)$, and standard deviation, $\sigma(\rho)$, of volume of "N" big data streams, where N = total number of big data streams being processed on cloud during time period "t." This comparison results in *EVol*() such that

1. If $0 \le Avg\left(\rho'_{i+1}\right) < (\mu(\rho) - \sigma(\rho))$, then *EVol*()= "Low."
2. If $(\mu(\rho) - \sigma(\rho)) \le Avg\left(\rho'_{i+1}\right) \le (\mu(\rho) + \sigma(\rho))$, then *EVol*() = "Medium."
3. If $Avg\left(\rho'_{i+1}\right) > (\mu(\rho) + \sigma(\rho))$, then *EVol*() = "Low."

Similarly, the reduce function for velocity calculates *EV*(). The working of these 2 reduce functions is summarized in Figure 7.

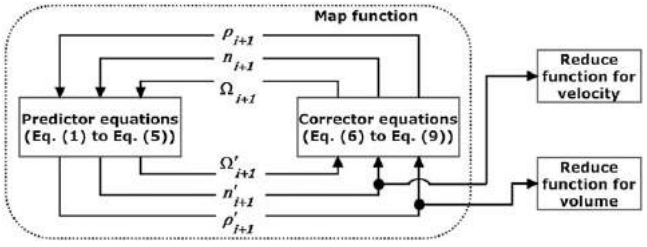

FIGURE 6 Working of predictor-corrector equations in each map function

3.1.3 | Step 3: Representation of estimated workload

The last step of WEst is aimed to represent *EVol*() and *EV*() of all the elementary streams in a form that can be efficiently used by CRM to allocate appropriate resources to big data stream. To achieve this goal, a term called data flow level is initially calculated for each elementary stream. Data flow level can be formally defined as.

> **Definition 1.** ***Data Flow Level—****If U and W are the 2 sets denoting volume and velocity, respectively, of an elementary stream "E" such that U = W = {Low, Medium, High} and X and Y are the 2 sets such that X = {(a,b) | (a,b) ∈ U × W} and Y = {c | c∈ℤ and 0 ≤ c ≤ 8}, then the data flow level of an "E" is given by a bijective function f: X → Y where f(Low, Low) = 0, f(Low, Medium) = 1, f(Low, High) = 2, f(Medium, Low) = 3, f(Medium, Medium) = 4, f(Medium, High) = 5, f(High, Low) = 6, f(High, Medium) = 7, f(High, High) = 8 is the one-to-one mapping from set X to set Y.*

Informally, data flow level of an elementary stream is given by Table 2. This allocation is based on experimental evidence (given in Section 4). It can be noted from the table that data flow level increases with the increasing volume of data in elementary stream. On the other hand, with the same volume, data level increases with the velocity of data in elementary stream. By using *EVol*(*T*), *EV*(*T*), and Table 2, the

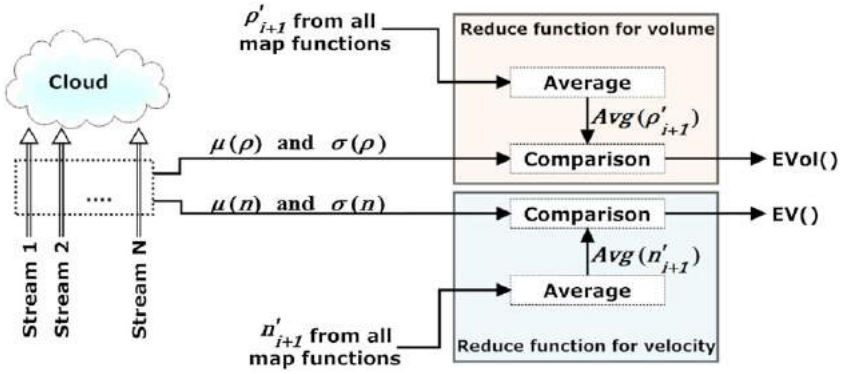

FIGURE 7 Working of reduce function

TABLE 2 Data flow level calculation table

EVol()	*EV*()	Data Flow Level	*EVol*()	*EV*()	Data Flow Level
Low	Low	0	Medium	High	5
Low	Medium	1	High	Low	6
Low	High	2	High	Medium	7
Medium	Low	3	High	High	8
Medium	Medium	4			

data flow level of text stream (denoted by *TF*) is calculated. Correspondingly, the data flow level of image stream (*IF*), audio stream (*AF*), and video stream (*VF*) are also calculated.

Recall that the 4 elementary streams correspond to 4 types of data in big data stream (data variety). Each elementary stream does not contain data from untrustworthy sources (data veracity). *EVol*() and *EV*() for each elementary stream are calculated by taking variation in data flow rate into consideration (volume, velocity, and veracity). This implies that *EVol*() and *EV*() represents the 5Vs of a big data stream. Furthermore, since TF, IF, AF, and VF are formed from *EVol*() and *EV*(), so they too represent the 5Vs of big data stream. In other words, TF, IF, AF, and VF, when taken together, represent the characteristics of big data in 5Vs. Hence, for efficient representation of 5Vs of big data stream, a quadruple called CoBa is introduced. CoBa is defined as

> **Definition 2.** ***Characteristics of Big Data (CoBa)—****If* $Y = \{c \mid c \in \mathbb{Z} \text{ and } 0 \le c \le 8\}$ *and* TF, IF, AF, and VF *denote, respectively, the data flow level of text, image, audio, and video streams, then an ordered quadruple* (C_1, C_2, C_3, C_4) *representing the volume, velocity, variety, veracity, and variability of a big data stream is called CoBa, where* $(C_1, C_2, C_3, C_4) \in (Y \times Y \times Y \times Y)$ *and* $C_1 = TF, C_2 = IF, C_3 = AF,$ *and* $C_4 = VF$. *In other words,* $CoBa = (TF, IF, AF, VF)$.

Let a big data stream forms CoBa = (8, 0, 0, 0). This implies that the data flow level is high for text stream and low for the other 3 elementary streams. In other words, this stream majorly consists of text data flowing in high volume and at high velocity. Such a stream is called text intensive big data stream in the proposed system. Similarly, names are given to few other streams as summarized in Table 3. This nomenclature and CoBa are used by CRM for efficient cloud resource allocation.

TABLE 3 Nomenclature of streams used in the proposed system

CoBa	Name given to big data stream
(8, 0, 0, 0)	Text intensive
(0, 8, 0, 0)	Image intensive
(0, 0, 8, 0)	Audio intensive
(0, 0, 0, 8)	Video intensive
(0, 0, 0, 0)	General purpose

Abbreviation: CoBA, characteristics of big data.

3.2 | Cloud resource manager

The data processing requirements are different for different big data streams. It is unreasonable to form static clusters of cloud resources for each requirement. Hence, there is a need to form dynamic clusters as per the current requirements. The task of dynamic cluster formation for a big data stream is accomplished by CRM with the help of SOM. Self-organizing map consists of an input layer and an output layer. The neurons in the output layer represent the clusters. Each input vector is mapped to one of the output neuron. All the input vectors that are mapped to same output neuron form a cluster. In the proposed system, CoBa forms the output neurons of SOM and cloud resources form input vectors. The cloud resources that map to same quadruple, CoBa, form a cluster. For example, let there are 3 big data streams on the cloud data center and let $B1, B2,$ and $B3$ be their respective CoBa quadruples. These 3 quadruples form the output neurons or the 3 output clusters. The free cloud resources, which form input vectors, are then mapped to these output neurons to form the respective clusters. Note that, in this example, only 3 clusters are formed. This will prevent a big data stream to move to another cluster if its CoBa changes. To remove this limitation, all the possible CoBa are initially arranged in a topological ordered fashion as shown by color coding in Figure 8. The system forms output neurons of few topological neighbors too. This will result in formation of topological ordered clusters of free cloud resources.

The process of mapping input vectors to output vectors, ie, cluster formation, is given by Algorithm 1. The algorithm 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 that decreases monotonically with the passage of time. This

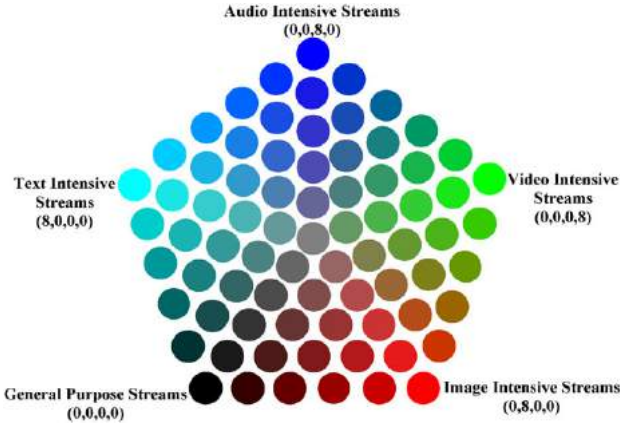

FIGURE 8 Topological ordering of characteristics of big data used in self-organizing maps

implies that initially, SOM learns quickly and later the speed of learning is decreased. After initialization, SOM randomly chooses one of the resource vectors (M_i) and finds its distance from each CoBa neuron. The CoBa neuron (B_J) with minimum distance is the winning vector. Therefore, resource M_i is allocated to stream with CoBa = B_J. This process is repeated for every resource. All the resources whose winning neuron is B_J are said to be in 1 cluster. The clusters hence formed are identified by a term called Characteristics of cluster (CoC). Characteristic of cluster can be formally defined as

Algorithm 1: Dynamic clustering by SOM

1. Initialize weights of edges 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 M_i, repeat steps 6 to 8.
5. For each output neuron j, calculate square of Euclidean distance of M_i as
$$D(j) = \sum_{k=1}^{q} \left(M_i k - w_{jk}\right)^2$$
6. Select J such that $D(J)$ is minimum.
7. Set $CoC(M_i) = CoBa(B_J)$
8. Update weights of all topological neighbors of J such that
$$w_{jk}(t,+,1) = (1,-,\eta,(t))w_{jk}(t) + \eta(t)M_k$$
9. Decrement η monotonically
10. Output the virtual clusters with their respective CoC.

Definition 3. ***Characteristics of Cluster (CoC)***—*If* $Y = \{c \mid c \in \mathbb{Z}$ *and* $0 \le c \le 8\}$ *and* $CoBa = (C_1, C_2, C_3, C_4)$ *is the winning output neuron for the resources in cluster, then Characteristics of Cluster (CoC) is the quadruple* (G_1, G_2, G_3, G_4), *where* $G_i \in Y$ *and* $G_1 = C_1, G_2 = C_2, G_3 = C_3,$ *and* $G_4 = C_4$. *In other words,* $CoC = (C_1, C_2, C_3, C_4)$.

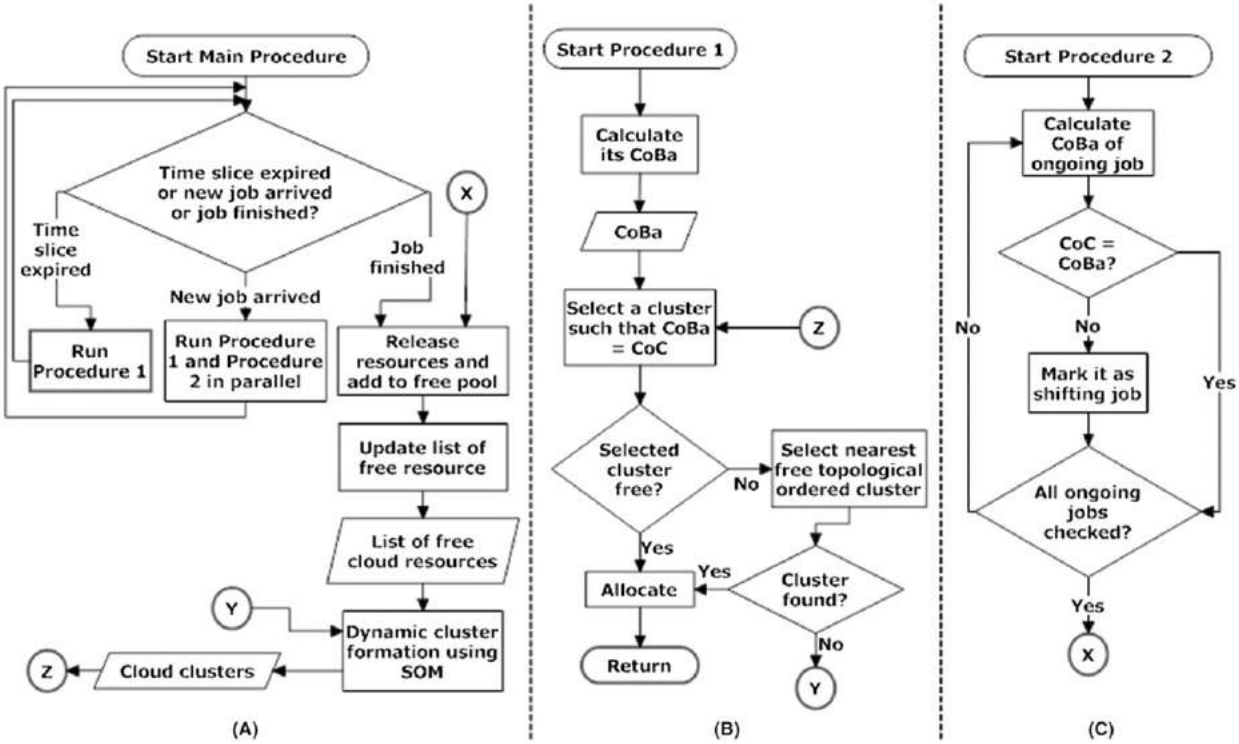

FIGURE 9 A, Flowchart explaining overall working of proposed system. B, Flowchart of procedure 1. C, Flowchart of procedure 2

Once clusters are formed, their allocation is a simple process as shown by Figure 9B. Initially, a cluster with CoC = CoBa (formed by big data stream) is selected for allocation. If the selected cluster has enough resources to 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.

3.3 | Systematic workflow of proposed system

The systematic workflow of the proposed system is shown by the flowcharts in Figure 9. Initially, the proposed system waits for the expiry of time slice "t" or arrival of a new job or completion of a job as shown in Figure 9A. For simplicity, a big data is referred as "job" from here on. In addition, recall that the value of "" depends upon the sampling rate of adaptive technique (as explained in Section 3). When the time slice "t" expires, the proposed system runs procedure 1. The flowchart for procedure 1 is shown in Figure 9B. Procedure 1 calculates CoBa of the stream and selects an appropriate cluster such that CoC = CoBa. If the selected cluster is free, it is allocated to the job, and control is returned to main procedure. It can be noted here that the selected cluster can be same as that on which the job is already running. In such a case, it continues to run on the same cluster. On the other hand, if the selected cluster is not free, then a nearest topological ordered cluster is searched and allocated. If no such cluster is found, the proposed system proceeds to form dynamic clusters of cloud resources by using SOM. Self-organizing map takes list of free resources as input and produces cloud clusters as output. The newly formed clusters are again taken as input by procedure 1 that allocates appropriate cluster to the job. After this allocation, control is returned to main procedure.

If the main procedure detects the arrival of a new, then procedures 1 and 2 are executed in parallel. Procedure 1 immediately allocates an appropriate cluster to the newly arrived job \to reduce waiting time. Procedure 2 checks all the other jobs in the system. Figure 9C shows flowchart for procedure 2 where CoBa of all the jobs in the system is calculated again. The jobs whose CoC is equal to its newly calculated CoBa continue to run on the same cluster. All the other jobs need to shift to a new cluster and are, therefore, marked as shifting jobs by the proposed system. When all the jobs are checked, the resources of shifting jobs are added to free pool list; clusters are formed by SOM and are allocated to shifting jobs using procedure 1. The control is again returned to main procedure. Lastly, on completion of a job, the resources held by it are released and added to free pool list. These resources are used during next cluster formation process. Hence, WEst (which calculates CoBa) and CRM (which forms and allocates resources using SOM) work in close coordination for efficiently managing resources of cloud.

4 | EXPERIMENTAL ANALYSIS

This section supports the proposed system with experimental evidences. In Section 4.1, the allocation of various data flow levels (summarized in Table 2) is supported experimentally. Section 4.2 analyzes the performance of proposed system by comparing it with other streaming data processing tools, which are Apache Storm 0.92, Apache Flume 1.5.2, and Apache S4 0.6.0. Storm is used by Twitter, Flume by Facebook, and LinkedIn and S4 by Yahoo for stream data processing. Section 4.2 is further divided into 3 subsections. Section 4.2.1 presents the method used to generate synthetic workload for the experiments. Section 4.2.1 presents the results of the experiment while Section 4.2.3 presents discussion of the results.

4.1 | Data flow level determination

In Section 3.1.3, it was stated that the allocation of data flow level, given in Table 2, is based on experimental evidence. So this section presents experimental results to support the stated fact. For experimentation, initially, the range of values of volume and velocity classified as low, medium, and high randomly defined and are given in Table 4. It can be noted here that velocity is specified in terms of number of data elements supplied to the system in 1 minute. For instance, if 20 image elements are supplied to the system per minute, then the velocity is said to be 20.

Thereafter, 4 databases are taken. The first database[44] is textual dataset consisting of 8 million vocabulary words. The second database[45] is an image dataset consisting of 68 040 images. The third database[46] is an audio dataset that is a collection of 1059 music tracks. The fourth database[47] is a surveillance video dataset consisting 29 hours of video. By using the first database, 3 workloads of 25, 45, and 65 GB of text data are generated. The data can be repeated to meet the volume requirement. Note that these 3 workloads lie in the range low, medium, and high, respectively, as per Table 4.

Initially, the workload of 25 GB is fed to Amazon Elastic Compute Cloud (EC2) compute optimized c4.large instances[48] at velocity of 50, and the execution time for

TABLE 4 Range of values of volume and velocity

Range	Volume, GB	Velocity, Data Elements per Min
Low	1-30	1-100
Medium	31-50	100-500
High	50-80	500-1000

counting distinct elements is noted. Later, the velocity is increased to 200 and 800, and the execution time is noted again. Here, note that the 3 velocities lie in low, medium, and high range as per Table 4. The same procedure is repeated for the other 2 workloads. In addition, the above experiment is repeated for image, audio, and video databases. The execution times under all the cases are shown by a bar chart in Figure 10.

4.1.1 | Discussion of results

It can be observed from Figure 10 that among all the volumes and velocities of text data (first 9 bars in the chart), the execution time is lowest when both volume and velocity of are low (shown by first bar in the chart). Therefore, data flow level in this case is assigned lowest value, ie, 0. Furthermore, the execution time increases when velocity of text data is increased to medium keeping volume low (as shown by second bar in the chart). This result leads to assigning value 1 to the data flow level. When the velocity of text data is increased to high keeping low volume, the execution time is further increased (as shown by third bar in the chart). Hence, in this case, value 2 is assigned to the data flow level.

Apart from this, the execution time for text data with medium volume and low velocity (shown by fourth bar in the chart) is higher than data with low volume and high velocity (shown by third bar in the chart). Therefore, the data level assigned is higher for the former case. On the similar grounds, all the data levels are determined. Furthermore, it can be observed from Figure 10 that similar trends are observed in case of image, audio, and video data. Hence, the data level determination given in Table 2 is justified.

4.2 | Performance evaluation of proposed system

4.2.1 | Workload generation method used

For efficiently evaluating the performance of proposed system, synthetic workload is generated. This workload generation uses all the 4 databases mentioned in Section 4.1. Here, the process is explained with the help of an example.

Let a workload is required such that it forms CoBa = (8, 8, 0, 0). This implies that in this workload, the volume and velocity of text and image data is high, while that of audio and video data is low (as per Table 2). For generating such a workload, IBM SPSS[49] is used. SPSS randomly selects data from textual database such that its volume lies in high range (say 60 GB). The range of values of volume used in the experiments is shown in Table 4. The data selection by SPSS is nonexclusive, ie, the data can be repeated to meet the volume requirement. The selected data is partitioned in 4 equal parts and added to 4 files such that each file consists of same volume (60/4 = 15 GB) of data. Similarly, data from image database is selected, partitioned into 4 parts and appended to the same 4 files. This process is repeated for the other 2 databases as well by taking low volume (say 16 GB) of data. This implies that each file consists of 15 GB text data, 15 GB image data, 4 GB audio data, and 4 GB video data. Hence, the workload consisting of 4 files created with known volume and variety. These 4 files are fed to the system in 4 minutes, ie, 1 file in 1 minute. Here, it can be noted that in a file, the volume of text data and image data is high. This implies that there are more text and image data elements in a file. Therefore, when the file is fed to the system, the velocity of text and image data will be automatically high. Similarly, the velocity of audio and video data will be low. Hence, CoBa formed by this workload is (8, 8, 0, 0). Similarly, the workloads forming different CoBa can be generated.

4.2.2 | Experimental setup and results

To test the performance of system, 15 workloads are generated with different CoBa as shown in Table 5. The generated workloads are fed to the system. Table 5 shows which workload is fed to the system at what time of the experiment. It can be observed that 1 workload is fed to the system after every

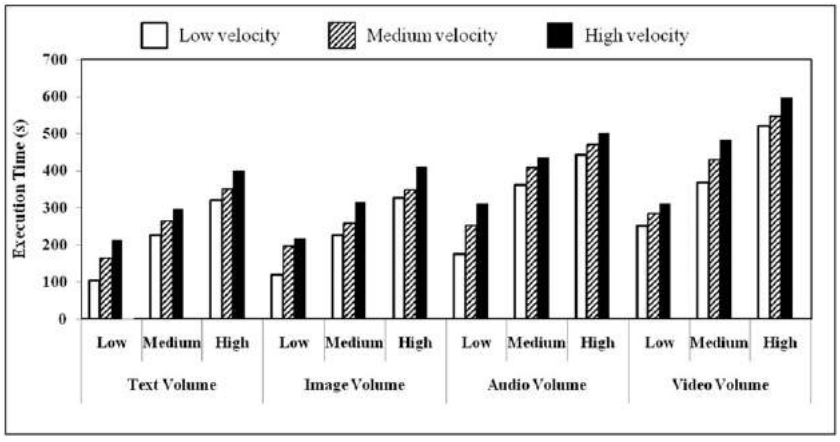

FIGURE 10 Experimental results justifying allocation of data flow level

TABLE 5 Various workloads generated and fed to the system

Workload No.	Fed to the System at Time, min	CoBa	Workload no.	Fed to the System at Time, min	CoBa
1	0	(8, 8, 0, 0)	9	32	(7, 8, 8, 8)
2	4	(0, 8, 0, 2)	10	36	(6, 6, 6, 0)
3	8	(0, 0, 8, 1)	11	40	(5, 3, 7, 4)
4	12	(8, 8, 8, 8)	12	44	(8, 4, 5, 7)
5	16	(2, 8, 4, 3)	13	48	(0, 7, 8, 2)
6	20	(8, 8, 8, 7)	14	52	(7, 0, 2, 8)
7	24	(1, 5, 2, 4)	15	56	(5, 7, 0, 6)
8	28	(4, 8, 1, 7)			

Abbreviation: CoBA, characteristics of big data.

4 minutes for a total experiment of 60 minutes. This implies that a big data stream is fed to the system such that its CoBa changes after every 4 minutes. In addition to the generated workloads, random data is supplied to the system to test veracity. Variability is automatically introduced due to change in CoBa. Hence, the big data stream characterized by 5Vs is supplied to the system for 60 minutes. In addition, this big data stream is also fed to Storm, Flume, and S4, which are streaming data processing tools. The proposed system and the 3 tools count the number of occurrences distinct elements of the stream. The proposed system selects virtual machines from Amazon EC2 instances from time to time based on data characteristics. The number and type of instances may vary with the variation of CoBa. On the other hand, Storm, Flume, and S4 use fixed number of instances for whole of the experiment. Each tool uses 10 Amazon EC2 compute optimized c4.large instances. The proposed system is implemented in a java-based application on Amazon EC2 compute optimized c4.large instances. The performance of the system is tested by comparing with other tools using 4 parameters, namely, execution latency, response time, resource utilization, and availability. The results are shown in Figure 11.

4.2.3 | Discussion of results

The result in Figure 11A shows that the execution latency of proposed system is initially higher. This is because time is taken by the system to compute CoBa of big data stream before resources are allocated to it. Once CoBa is calculated,

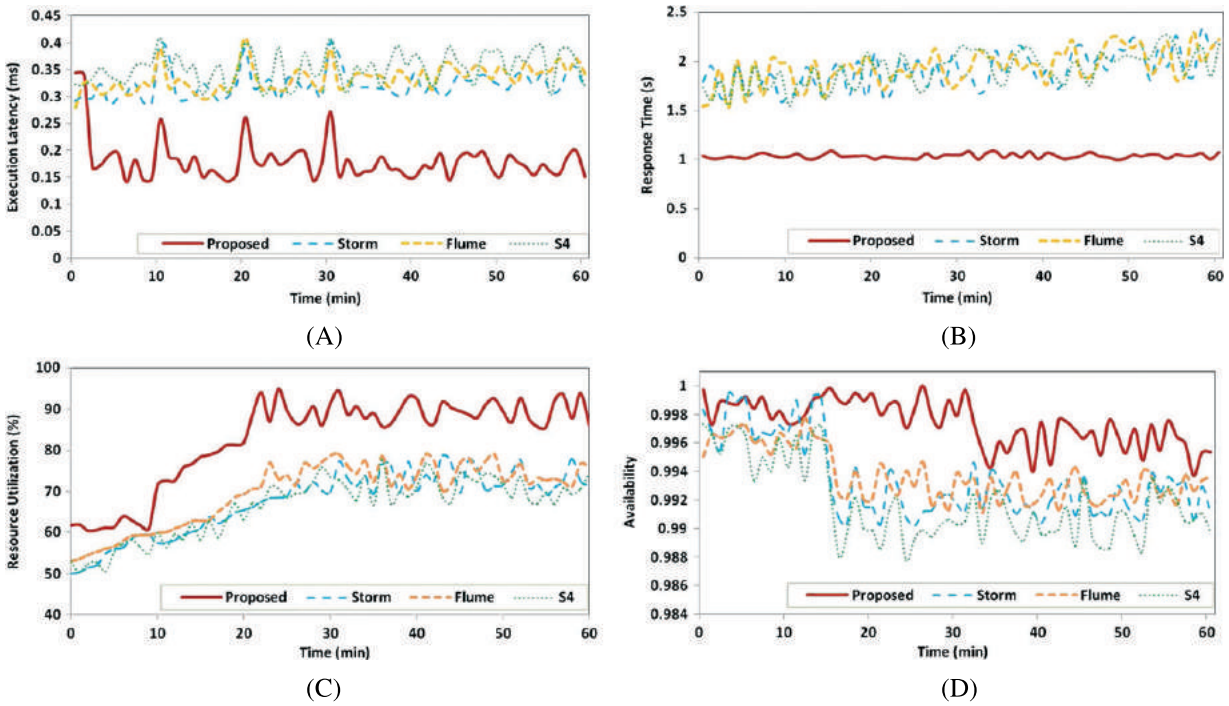

FIGURE 11 Comparison of performance of proposed system with other tools. A, Execution latency. B, Response time. C, Resource utilization. D, Availability

appropriate resources are allocated by the system that results in lower execution latency for proposed system as compared with other tools. Moreover, the execution latency experiences peaks at 3 points in the experiments. These 3 points are at 12th, 20th and 32nd minutes of the experiment. This is because the big data form CoBa equal to (8,8,8,8), (8,8,8,7), and (7,8,8,8) at these 3 points of time (refer Table 5). This implies volume and velocity of data in all the elementary streams are increased suddenly leading to higher execution latency for proposed system as well as for other tools.

Figure 11B shows the response time of the system and other tools. It is observed that the response time roughly increases with the increase in time in case of Storm, Flume, and S4. This is because the resources are not allocated according to data characteristics due to which they are unable to adapt according to changing conditions. On the other hand, the response time of proposed system remains almost constant for whole of the experiment since proposed system allocates appropriate resource to big data stream from time to time based on data characteristics.

Figure 11C shows the comparison results of proposed system with other tools in resource utilization. The results show that utilization of cloud resources is higher in case of proposed system. This is due the fact that whenever CoBa of a stream changes, it is moved to a more suitable cluster according to data characteristics. In other words, the consideration of 5Vs for resource allocation allowed better selection of resources for data processing that enhanced utilization of resources. Moreover, the resource utilization initially increases with time and then become constant. This is because SOM is learning technique. It learns from the current state of the system and improves its output as the time passes.

Figure 11D shows that resource availability is higher in case of proposed system as compared to other tools. The reason for higher resource availability is the topological ordered cluster formed by proposed system. The topological ordering allows big data stream to acquire other similar resources in case of nonavailability of required resources rather than waiting for busy resources. The results presented in this section suggest that the proposed system efficiently manages cloud resource in real time for big data streams.

5 | CONCLUSIONS

In this paper, a dynamic system is proposed that works in real time to predict volume, velocity, variety, variability, and veracity of data in big data streams. These 5Vs are used for dynamically allocating appropriate cloud resources to big data streams. Extensive experiments are performed to evaluate the performance of proposed system by comparing it with Storm, Flume, and S4. Virtual machines are selected from Amazon EC2 instances by the proposed system from time to time based on data characteristics. On the other hand, Amazon EC2 compute optimized c4.large instances are allocated for Storm, Flume, and S4. This allocation is constant throughout the experiment. The experimental results illustrate a performance edge of proposed system over other tools. The overall execution latency for processing data streams is 11.013 milliseconds for proposed system as compared with 19.648 milliseconds for Storm, 20.403 - milliseconds for Flume, and 21.567 milliseconds for S4. Furthermore, the average response time of proposed system is 1.036 seconds as compared with 1.908 seconds for Storm, 1.935 seconds for Flume, and 1.921 seconds for S4. This shows that the latency and response time is reduced considerably due to dynamic resource allocation in the proposed system based on data characteristics. Moreover, availability of cloud resources in proposed system is enhanced by 9.31%, 7.95%, and 10.02% as compared with Storm, Flume, and S4, respectively. In addition, resource utilization of proposed system is improved by 24.5%, 22.5%, and 34.4% over Storm, Flume, and S4, respectively. This is due to topological ordering of cloud resources. Therefore, predicting 5Vs and topological ordering of resources form 2 major strengths of proposed system.

REFERENCES

1. Schroeck M, Shockley R, Smart J, Romero-Morales D, Tufano P. Analytics: the real-world use of big data. In IBM Global Business Services; 2012.
2. Gandomi A, Haider M. Beyond the hype: big data concepts, methods, and analytics. *Int J Inf Manage*. 2015;35:137-144.
3. TechAmerica Foundation: Federal Big Data Commission. A practical guide to transforming the business of government; 2012:1–40. Retrieved from http://www.techamerica.org/Docs/fileManager.cfm?f=techamerica bigdatareport-final.pdf
4. Cukier K. The economist, data, data everywhere: a special report on managing information, 2010, February 25; 2010. Retrieved from http://www.economist.com/node/15557443
5. Lazar N. The big picture: big data computing. *Chance*. 2013;26 (2):28-32.
6. Yang C, Huang Q, Li Z, Liu K, Hu F. Big data and cloud computing: innovation opportunities and challenges. *Int J Digit Earth*. 2016;8947:1-41.
7. Hashem IAT, Yaqoob I, Badrul Anuar N, Mokhtar S, Gani A, Ullah Khan S. The rise of 'big data' on cloud computing: review and open research issues. *Inf Syst*. 2015;47:98-115.
8. Yaqoob I, Hashem IAT, Gani A, et al. Big data: from beginning to future. *Int J Inf Manage*. 2016;36(6):1231-1247.
9. Twitter usage statistics—Internet live stats. [Online]. Available: http://www.internetlivestats.com/twitter-statistics/. [Accessed: 20-Jan-2017].
10. Zhang Q, Chen Z, Yang LT. A nodes scheduling model based on Markov chain prediction for big streaming data analysis. *Int J Commun Syst*. 2015;28(9):1610-1619.

11. Baughman AK, Bogdany RJ, McAvoy C, Locke R, O'Connell B, Upton C. Predictive cloud computing with big data: professional golf and tennis forecasting [application notes]. *IEEE Comput Intell Mag*. 2015;10(3):62-76.
12. Castiglione A, Pizzolante R, De Santis A, Carpentieri B, Castiglione A, Palmieri F. Cloud-based adaptive compression and secure management services for 3D healthcare data. *Futur Gener Comput Syst*. 2015;43–44:120-134.
13. De Francisci Morales G, Bifet A, Khan L, Gama J, Fan W. IoT Big Data Stream Mining. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-KDD '16*. San Francisco, CA, USA: ACM; 2016:2119-2120.
14. Rehman MH ur, Liew CS, Wah TY, Khan MK. Towards next-generation heterogeneous mobile data stream mining applications: opportunities, challenges, and future research directions. *J Netw Comput Appl*. 2017;79:1-24.
15. Rajaraman A, Ullman JD. Bloom Filter. In: *Mining of Massive Datasets*. First edit ed. Cambridge University Press; 2014:116-118.
16. Kalman RE. A new approach to linear filtering and prediction problems. *J basic Eng*. 1960;82(1):35-45.
17. Kohonen T. *Self-Organization and Associative Memory.* vol. 8. Berlin, Heidelberg: Springer Berlin Heidelberg; 1989.
18. Sapountzi A, Psannis KE. Social networking data analysis tools and challenges. *Futur Gener Comput Syst*. 2016: https://doi.org/10.1016/j.future.2016.10.019.
19. Chen V, Chiang H, Sorey R. Business intelligence and analytics: from big data to big impact. *MIS Q*. 2012;36(4):1165-1188.
20. Stergiou C, Psannis KE. Recent advances delivered by mobile cloud computing and Internet of Things for big data applications: a survey. *Int J Netw Manag*. 2016. https://doi.org/10.1016/j.future.2016.10.019
21. Hahanov V, Miz V, Litvinova E, Mishchenko A, Shcherbin D. Big Data Driven Cyber Physical Systems. In: *The Experience of Designing and Application of CAD Systems in Microelectronics*. Polyana, Ukraine: IEEE; 2015:76-80.
22. Van Den Dam R. Internet of Things: The Foundational Infrastructure for a Smarter Planet. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. St. Petersburg, Russia; Vol. 8121 LNCS; 2013:1-12.
23. Chen M, Mao S, Liu Y. Big data: a survey. *Mob Networks Appl*. 2014;19(2):171-209.
24. Philip Chen CL, Zhang CY. Data-intensive applications, challenges, techniques and technologies: a survey on big data. *Inf Sci (Ny)*. 2014;275:314-347.
25. Pumma S, Achalakul T, Xiaorong L. Automatic VM Allocation for scientific application. In: *Proceedings of the International Conference on Parallel and Distributed Systems-ICPADS*. Singapore: IEEE; 2012:828-833.
26. Singh S, Chana I. Cloud resource provisioning: survey, status and future research directions. *Knowl Inf Syst*. 2016;49(3):1005-1069.
27. Zhang J, Huang H, Wang X. Resource provision algorithms in cloud computing: a survey. *J Netw Comput Appl*. 2016;64:23-42.
28. Memos VA, Psannis KE. A New Methodology Based on Cloud Computing for Efficient Virus Detection. In: *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering*. Springer International Publishing; 2015:37-47.
29. Vasile M-A, Pop F, Tutueanu R-I, Cristea V, Kołodziej J, Kolodziej J. Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. *Futur Gener Comput Syst*. 2015;51:61-71.
30. Zhan Z-H, Liu X-F, Gong Y-J, Zhang J, Chung HS-H, Li Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput Surv*. 2015;47(4):1-33.
31. Sfrent A, Pop F. Asymptotic scheduling for many task computing in big data platforms. *Inf Sci (Ny)*. 2015;319:71-91.
32. Sandhu R, Sood SK. Scheduling of big data applications on distributed cloud based on QoS parameters. *Cluster Comput*. 2014;18(2):817-828.
33. Kokkonis G, Psannis KE, Roumeliotis M, Schonfeld D. Real-time wireless multisensory smart surveillance with 3D-HEVC streams for Internet-of-Things (IoT). *J Supercomput*. 2016;1-19. https://doi.org/10.1007/s11227-016-1769-9
34. Plageras AP, Psannis KE, Ishibashi Y, Kim B-G. IoT-based surveillance system for ubiquitous healthcare. *IECON 2016-42nd Annu. Conf. IEEE Ind. Electron. Soc.*; 2016:24–27.
35. Trilles S, Belmonte Ò, Schade S, Huerta J. A domain-independent methodology to analyze IoT data streams in real-time. A proof of concept implementation for anomaly detection from environmental data. *Int J Digit Earth*. 2017;10(1):103-120.
36. Su X, Gilman E, Wetz P, Riekki J, Zuo Y, Leppänen T. Stream Reasoning for the Internet of Things. In: *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics-WIMS '16*. Nîmes, France: ACM; 2016:1-10.
37. Yasumoto K, Yamaguchi H, Shigeno H. Survey of real-time processing technologies of IoT data streams. *J Inf Process*. 2016;24(2):195-202.
38. Cortés R, Bonnaire X, Marin O, Sens P. Stream processing of healthcare sensor data: Studying user traces to identify challenges from a big data perspective. *Procedia Comput Sci*. 2015;52:1004-1009.
39. Sun D, Zhang G, Yang S, Zheng W, Khan SU, Li K. Re-stream: real-time and energy-efficient resource scheduling in big data stream computing environments. *Inf Sci (Ny)*. 2015;319:95-112.
40. Tolosana-Calasanz R, Bañares JÁ, Pham C, Rana OF. Resource management for bursty streams on multi-tenancy cloud environments. *Futur Gener Comput Syst*. 2016;55:444-459.
41. Rahman M, Graham P. Responsive and efficient provisioning for multimedia applications. *Comput Electr Eng*. 2016;53:458-468.
42. Jain A, Chang EY. Adaptive Sampling for Sensor Networks. In: *Proceeedings of the 1st international workshop on Data management for sensor networks in conjunction with VLDB 2004—DMSN '04*. Toronto, Canada: ACM; 2004:10–16.
43. Feng B, Fu M, Ma H, Xia Y, Wang B. Kalman filter with recursive covariance estimation-sequentially estimating process noise covariance. *IEEE Trans Ind Electron*. 2014;61(11):6253-6263.
44. UCI machine learning repository: bag of words data set. [Online]. Available: https://archive. ics.uci.edu/ml/datasets/Bag+of+Words. [Accessed: 20-Jan-2017].

45. UCI machine learning repository: corel image features data set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Corel+Image+Features. [Accessed: 20-Jan-2017].

46. UCI machine learning repository: geographical original of music data set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Geographical+Original+of+Music. [Accessed: 20-Jan-2017].

47. Oh S, Hoogs A, Perera A, Cuntoor N, Chen CC, Lee JT, Mukherjee S, Aggarwal JK, Lee H, Davis L, Swears E. A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video. In: *CVPR 2011*. Colorado, USA: IEEE; 2011:3153-3160.

48. EC2 instance types—Amazon web services (AWS). [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed: 22-Jan-2017].

49. IBM-SPSS software—India. IBM Corporation; 19-Jan-2017.

Navroop Kaur received her M.Tech degree with distinction in Computer Science and Engineering from Guru Nanak Dev University, Amritsar, and B.Tech degree with distinction from Beant College of Engineering and Technology, Gurdaspur. Presently, she is pursuing her doctorate degree in the Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar. She has publications in various SCI/SCIE journals. Her current working research areas include Internet of Things, big data, fog computing, and cloud computing.

Dr Sandeep K. Sood did his PhD in Computer Science and Engineering from IIT Roorkee, India. He is currently working as Head and Professor, Computer Science and Engineering, G.N.D.U. Regional Campus, Gurdaspur. He has 17 years of teaching and 10 years of research experience. He has more than 50 research publications. His research areas are network and information security, cloud computing, big data, fog computing, and Internet of Things.

How to cite this article: Kaur N, Sood SK. Dynamic resource allocation for big data streams based on data characteristics (5Vs). *Int J Network Mgmt*. 2017:e1978. https://doi.org/10.1002/nem.1978