Multi-Agent based MapReduce Model for Efficient Utilization of System Resources

Heena kousar, B.R. Prasad Babu
Department of CSE, East Point College of Engineering &Technology
Bangalore, India

Article Info

Article history:

Received Nov 21, 2017
Revised Jan 8, 2018
Accepted Jan 27, 2018

Keywords:

Agent
Cloud computing
Forcasting
Hadoop
MapReduce
Parallel computing
Stream computing

ABSTRACT

Recently with increased adoption of big data, Internet of Things and sensor technology by various organization for provisioning smart intelligent services for various application uses. Data processing on real-time social media and sensor data is been a key area of research in recent times and these data are massive and continuous. Smart application using sensor and social media data can be classified into three class: 1) online processing of streaming data; 2) online processing of historical data; and 3) hybrid processing of both. The existing model are designed considering stream or batch processing. For provisioning real-time processing MapReduce framework using Hadoop framework is considered by state-of-art technique for data inflow forecasting. However, the Hadoop based forecasting model are not efficient in fully utilizing system resource. Agent based MapReduce forecasting model is adopted by state-of-art technique to utilize system efficiently. However, they incurs high computation overhead, thus increase cost of computing cost. To overcome this work present an agent based Data Inflow Forecasting (DIF) model for both stream and non-stream (historical) data by using Multivariate Gaussian Mixture (MGM) model. This work present an Agent based MapReduce (AMR) framework to process data in real-time and utilize system resource efficiently. To provide scalability for processing social media and sensor data DIF-AMR model adopts cloud computing architecture. Experiment are conducted to evaluate performance of DIF-AMR of over existing model shows significant performance improvement in terms of computation time.

Corresponding Author:

Heena kousar,
Department of CSE, East Point College of Engineering &Technology,
Bangalore, India.
Email: hkheenakousar73@gmail.com

1. INTRODUCTION

With increasing adoption of Internet of Things (IoT) to provision real-time services for various smart application uses. Such as smart transportation system, disaster management system etc. where sensor are deployed across the globe to provision real-time services to users. Data processing based on real-time sensor data is been a key research area of various academia, industry and government organization in recent times. A data driven paradigm of smart infotainment applications, aiming at mining the essential value of sensor data and promoting smart services, is arisen in smart intelligent transport domain [1]. However, sensor data and its real-time application services have some new characteristics comparing to traditional sensor data and applications:

1) Massiveness: With the growth of smart infotainment application services, large amount of data is generated by different sensor periodically. For example, location and traffic management services generates sensor data at very high rate, which lead to size of petabyte of data in a short term.

2) Streaming: Sensor data from different sources such as user wearable and its location and mobility information arrives into the server continuously from various sources as a streams at rapid rate. Efficient mechanism is needed to process/perform analysis on such stream data in real-time without any interruption.
3) Real-Time: In many application services such as pollution monitoring, live vehicle traffic monitoring computing, we need react quickly to such large sensor data once they are generated. Since these information becomes obsolete/outdated quickly. Performing analysis on such large data in near real-time requires an efficient mechanism to collect input stream, organize the data and perform computing and data analysis.
4) Ever-Expanding: As years passes new type and number of sensors available in market increases. A scalable real-time processing design is required to handle such ever increasing sensor data volume. That is, the future real-time prediction model should not only process large volume of sensor data and guarantee real-time quality by adding computing nodes, but also new types of sensor data easily.

Additionally, there are many application based on sensor data processing which can be classified into three classes based on how the sensor data are processed as shown in Table 1. Online processing of streaming data, offline processing of historical/statistical data and hybrid processing of both historical and streaming data. To cope with different applications in smart infotainment domain, the processing system need to deal with both massively large historical data and streaming data. The historical sensor data is considered to be more valuable and hence need to be mixed with stream data in many cases for performing statistical computation, pattern discovery and data inflow prediction.

Table 1. Sensor Data Application Types

Type	Application	Characteristic
Online processing of real-time streaming sensor data	Fake license plate detection, black listed vehicle/users	Single row queries form sensor data stream
	Real-time monitoring of traffic and travel time prediction	Multiple rows aggregation from windowed data on stream
Offline processing of historical/statistical sensor data	Mining vehicle and user data	Aggregation on batch data
Hybrid processing of both historical and stream sensor data	Over speeding vehicle discovery, Cloned license plate detection	Single streaming rows comparison from statistical sensor data

In order cater performance requirement of smart infotainment system applications, how to provide scalable and real-time processing support for massively large and continuous sensor data is becoming on essential issue in smart infotainment system. Novel methodologies are required to support abovementioned applications and to overcome underlying challenges.

Challenges in designing sensor data processing system: To cope with above mentioned features and applications of sensor data, there exist challenges in designing a sensor data processing system that can process massively large sensor data in near real-time and handle different type of sensor data application. The challenges are summarized:.

1. Firstly, management of both historical and stream sensor data, especially considering the massive volume of historical data and continuously arriving stream data. The sensor data is composed of different dimensions and needs to be transferred to different processing task of smart infotainment system. Some data may be shared across different applications. As a result lead to complexity in managing sensor data.
2. Secondly, building of an integrated architecture to support online processing of stream data, offline processing of historical data, and hybrid processing of both stream and historical data. Since stream data is continuous and historical data is massive, they need to be segmented and fused for processing in distributed architecture in a coordinated manner. For example, to perform computation of clone plate vehicle identification, millions of historical records and stream data in speed of 10,000 records per seconds needs to be executed every second.
3. Thirdly, there is increasing innovative smart application that depend on processing of sensor data. Some of these may be short-term applications. As a result, effective and simplified data inflow pattern of application and sensor data processing system are needed.

The significant growth of BigData and cloud technologies has incurred significant challenges in network architecture in datacenter to process sensor data in real-time. In [2] adopted traffic engineering technique to address the bandwidth requirement of datacenter network. In [3] presented a data inflow prediction model by adopting rate control mechanism based on predicted traffic in network. Recently researcher have considered tight integration of application and network layer for optimizing network and routing layer for predicting application traffic [4]. All these model requires prior understanding of application traffic in data center networks (i.e. the ability to forecast data inflow before packet arrives in network). However, it is difficult to predict data inflow of application accurate.

All existing methodology [5] focused on predicting data inflow based on network level parameters using heuristic algorithm. For example, [5] computes data inflow using flow counter measurement on switches and [6] used socket buffer occupancy at each nodes to compute data inflow for each nodes. However, these methodologies have number of drawbacks. Firstly, most of them cannot predict the data inflow demand before data enter the network. Secondly, performance requirement identified on network path cannot accurately reflect the actual demand of application due to congestion control at end nodes and presence of noise of background flow. Thirdly, they fail to identify priority information and fine-grained data inflow dependencies imposed by applications. As a result, network-layer based approaches are shown to be performing poorly in predicting the real-time application demand [7], thus affecting system performance.

Agent based forecasting model aid in achieving fine-grained performance for performing real-time analysis on social media and sensor data adopting MapReduce framework using cloud architecture [8], [9] and [10]. Hadoop framework [11] is a widely used MapReduce framework that adopts cloud platform. MapReduce framework is composed of two stage. In the initial stage, input data to be processed is fragmented into segments. Each segment is associated with a map computing worker that provides ⟨Key | Value⟩ pairs as outputs. The outputs obtained are sorted on the basis of the Key values associated. The sorted values are the input to reduce computing workers, that is, ⟨Key | Sorted List (Value)⟩. Reduce computing workers keeps the outcome in Hadoop distributed file system (HDFS). The map and reduce computing workers are generally virtual machines (VMs) in public cloud environments. A simple MapReduce model deployed on the VM based computing environment is shown in Figure 1.

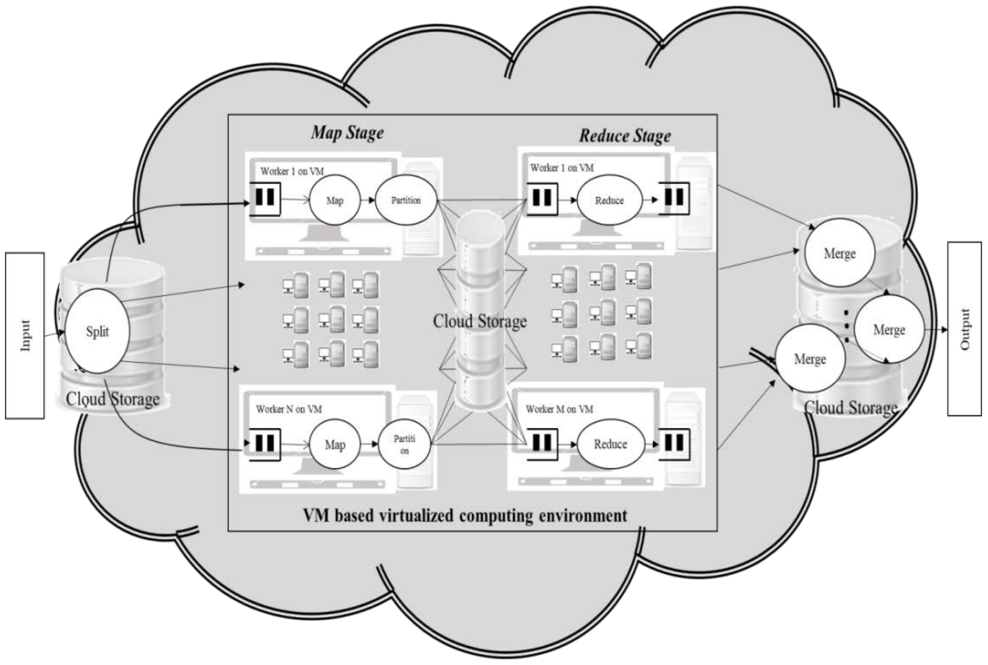

Figure 1. Architecture of MapReduce framework deployed on public cloud platform

In [8] presented agent based forecasting model to predict air pollution level across different region. To improve accuracy of prediction their model adopted artificial intelligence and to forecast shortest path Dijkstra algorithm. The perform analysis on real-time Hadoop MapReduce framework is adopted. However, they considered performance evaluation only for offline data. In [9] presented a hybrid model to process both stream and non-steam sensor data by adopting MapReduce framework. Their model achieves scalable performance over tradition open source streaming model Storm [12]. However, their model incurs computing overhead. As a result incur high cost of processing. To minimize cost [10] presented an agent based forecasting model on top of Hadoop. The model incurs slight overhead for application but at the reduction of computing time. Thus, reducing cost of computing. However, result presented in [13] shows adopting

Hadoop MapReduce for processing incurs computing overhead. Since, it does not fully utilize the resource available.

To cope with research challenges in designing efficient real-time social media and sensor data processing and Data Inflow Forecasting (DIF) model, this work present an agent based data inflow forecasting model for both stream and non-stream (historical) data by using Multivariate Gaussian Mixture (MGM) model. To perform analysis on massively large social media and sensor data, this work adopt parallel computing framework. This work present an Agent based MapReduce (AMR) framework to process data in real-time and utilize system resource efficiently. To provide scalability for processing social media and sensor data AMR model adopts cloud computing architecture.

The Contribution of research work is as follows:

a) A novel Agent based model MapReduce framework is presented for data inflow forecasting.
b) The proposed DIF-AMR support processing of stream, non-stream data and combination of both in real-time.
c) The DIF-AMR provides scalable processing by adopting cloud computing architecture.
d) DIF-AMR utilize system resource efficiently and experiment outcome shows significant reduction in computing time.

The rest of the paper is organized as follows. In section II the proposed agent based forecasting model is presented. In penultimate section experimental study is carried out. The conclusion and future work is described in last section.

2. PROPOSED AGENT BASED FORECASTING MODEL

This section present an efficient stream and non-stream data inflow forecasting model by adopting agent based MapReduce framework. The process of performing stream and non-stream data inflow forecasting model on agent based MapReduce framework consist of following steps:

a) Data inflow predictor selection

Data inflow predictor can be defined by Multivariate Gaussian Mixture (MGM) model, which is composed of numerous weighted Gaussian likelihood density model. The GM model with $\mathcal{K}$ variables can be stated as follows:

$$\mathbb{L}(\mathcal{D}|\mathcal{G}) = \sum_{q=1}^{\mathcal{K}} \beta_{q} \ell_{q}(\mathcal{D}|\gamma_{q}). \tag{1}$$

where $\mathcal{D}$ is the collection of data streams, ℓ_{q} is the likelihood density for the q^{th} variable, γ_{q} is the qualifier specifier for the q^{th} variable, satisfying $\beta_{q} > 0$, $\sum_{q=1}^{\mathcal{K}} \beta_{q} = 1$, $\mathcal{G} = (\beta_1, \dots, \beta_{\mathcal{K}}, \gamma_1, \dots, \gamma_{\mathcal{K}})$ is the qualifier specifier set for GM model.

b) Qualifier specifier approximation

Expectation maximization algorithm is one of the extensively used methodology for qualifier specifier approximation. Expectation maximization algorithm aid in minimizing the complexity of maximum likelihood estimation. Expectation maximization algorithm is composed of two phases such as Expectation phase and Maximization phase. In Expectation phase the responding degree is computed using present qualifier specifier and in Maximization phase, qualifier specifier are updated based on responding degree. For one K-variable GM model, in which collection of data stream is $\mathcal{D} = \{d_1, \dots, d_{\mathcal{J}}\}$ and the qualifier specifier are $\mathcal{G}_{p} = (\varphi_{p}, \delta_p)$,$p = 1, \dots, \mathcal{K}$, one iteration of the expectation maximization algorithm is expressed as follows:

Expectation phase:

$$\mathbb{L}(p|d_{p}, \mathcal{G}) = \frac{l(d_{q}|\mathcal{G}_{p}^{O}) * \beta_{p}^{O}}{\sum_{t}^{\mathcal{K}} l(d_t|\mathcal{G}_{p}^{O}) * \beta_{p}^{O}} \tag{2}$$

Maximization phase:

$$\beta_{p}^{\mathcal{N}} = \frac{1}{\mathcal{J}} \sum_{q=1}^{\mathcal{J}} \mathbb{L}(p|d_{q}, \mathcal{G}) \tag{3}$$

$$\varphi_{p}^{O} = \frac{\sum_{q=1}^{\mathcal{J}} d_{q} * \mathbb{L}(p|d_{q},\mathcal{G})}{\sum_{q=1}^{\mathcal{J}} \mathbb{L}(p|d_{q},\mathcal{G})} \quad (4)$$

$$\delta_{p}^{O} = \frac{\sum_{q=1}^{\mathcal{J}} \mathbb{L}(p|d_{q},\mathcal{G}) * (d_{q} - \varphi_{p}^{O}) * (d_{q} - \varphi_{p}^{O})^{\mathcal{B}}}{\sum_{q=1}^{\mathcal{J}} \mathbb{L}(p|d_{q},\mathcal{G})}. \quad (5)$$

The expectation maximization will stop if convergence condition is satisfied or predefined iteration is completed. The local data inflow for all node is obtained post completion/termination of expectation maximization algorithm.

c) Global inflow prediction model

In order to predict the global inflow of data stream, the local inflow of data stream obtained by expectation maximization algorithm should be pooled. For that firstly, the similarity among different local models is obtained. Secondly, the local models are classified. And lastly, the local models are pooled based on weight of each class to form global models. Let $\mathcal{H}_1$, $\mathcal{H}_2$ represent two local models, and the distance among them is computed as follows:

$$\mathcal{A}(\mathcal{H}_1,\mathcal{H}_2) = \frac{1}{\beta(\mathcal{H}_1,\mathcal{H}_2)}. \quad (6)$$

Where,

$$\mathcal{B}(\mathcal{H}_1,\mathcal{H}_2) = \sum_{q}^{s} \int_{-\infty}^{+\infty} \mathcal{J}_{\varphi_1^q \Sigma_1^q}(\vec{d}) * \mathcal{J}_{\varphi_1^q \Sigma_2^q}(\vec{d}) s\vec{d} \quad (7)$$

The similarity among two local models is considered to be high if two local models are close to each other. Then classification problem of local models can be stated as a minimum spanning tree problem, and can be addressed using [8]. Let the weight of local models be expressed as:

$$\beta_{p,\mathcal{K}} = \{\beta_1, \dots, \beta_j\}, \quad (8)$$

Covariance matrix of local models is expressed as:

$$\delta_{p,\mathcal{K}} = \delta_1, \dots, \delta_j, \quad (9)$$

And average value of local models is expressed as:

$$\varphi_{p,\mathcal{K}} = \{\varphi_1, \dots, \varphi_j\}. \quad (10)$$

Then to identify which set the model $\mathcal{H}_q$ belongs to is expressed by following function

$$\mathcal{T} = (\mathcal{H}_q, r) = \begin{cases} 1, & \mathcal{H}_q \in \mathcal{H}_{\mathcal{R}} \\ 0, & \mathcal{H}_q \notin \mathcal{H}_{\mathcal{R}}. \end{cases} \quad (11)$$

Let consider a set of data stream for local model $\mathcal{H}_q$ be $\alpha(\mathcal{H}_q)$. If data stream size of local models are not equal, the weight of global models can be computed as:

$$\beta_r = \frac{\sum_{q=1}^{j} \beta \cdot \alpha(\mathcal{H}_q) \cdot \mathcal{T}(\mathcal{H}_q, r)}{\sum_{q=1}^{j} \beta \cdot \alpha(\mathcal{H}_q)}, \quad (12)$$

The covariance matrix of global models is computed as:

$$\varphi_r = \frac{\sum_{q=1}^{j} \beta_q \cdot \alpha(\mathcal{H}_q) \cdot \delta_q \cdot \mathcal{T}(\mathcal{H}_q, r)}{\sum_{q=1}^{j} \beta_q \cdot \alpha(\mathcal{H}_q) \cdot \mathcal{T}(\mathcal{H}_q, r)}, \quad (13)$$

The average value of global models is computed as:

$$\varphi_r = \frac{\sum_{q=1}^{j} \beta_q \cdot \alpha(\mathcal{H}_q) \cdot \varphi_r \cdot \mathcal{T}(\mathcal{H}_q, r)}{\sum_{q=1}^{j} \beta_q \cdot \alpha(\mathcal{H}_q) \cdot \mathcal{T}(\mathcal{H}_q, r)} \tag{14}$$

d) Agent based MapReduce model

The proposed agent based MapReduce model is composed of following agent as shown in Figure 2.

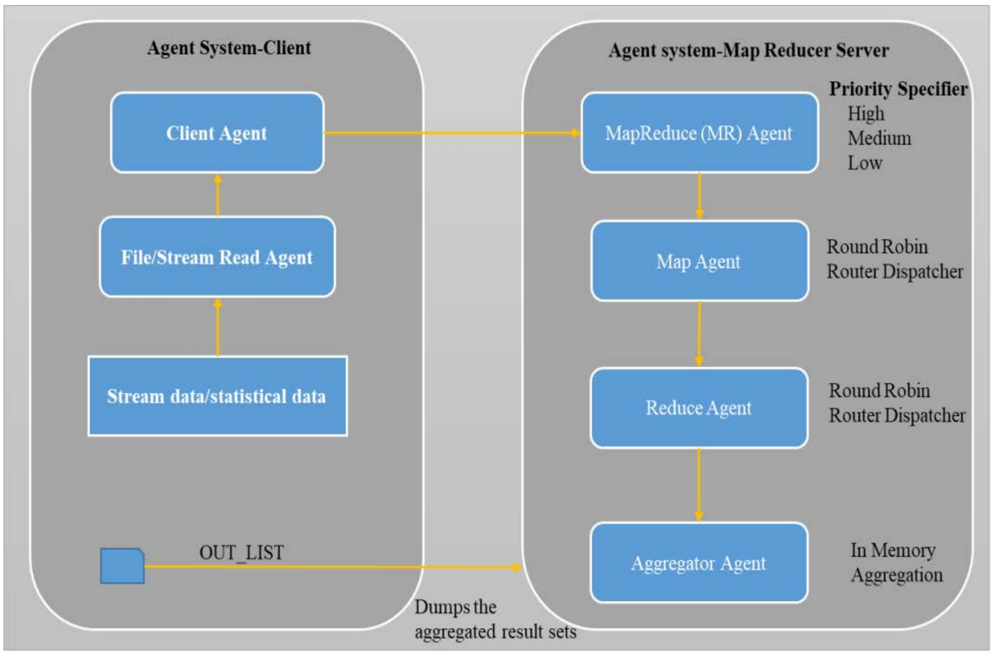

Figure 2. Architecture of proposed Agent based MapReduce (AMR) Model

File/Stream Read Agent: here the client system (File/Stream Read Agent) read a text file/stream data and sends each line of text as message to the Client Agent.

Client Agent: the Client Agent has the reference to the Remote Agent (MapReduce Agent) and the message is passed on the Remote Agent.

Server/MapReduce Agent: the Server/MapReduce Agent gets the message from the Client Agent. The Agent uses Priority Specifier to select the priority of the message and filters the queues accordingly. The

Priority Specifier is used to segregate the message among the MapReduce requests and retrieving the list of outcomes (OUT_LIST) message from the Aggregator Agent.

MapReduce Agent: sends across the message to the Map Agent for mapping the words. This Map Agent uses Round Robin Router dispatcher. After mapping the words, the messages is sent across to the Reduce Agent for reducing the words. Similar to Map Agent the Reduce Agent also uses Round Robin Router dispatcher. The reduced results are sent to the Aggregator Agent that does an in-memory aggregation of the result.

e) Agent based MapReduce implementation of data inflow forecasting model

This section present the detail implementation of data inflow forecasting on agent based MapReduce framework. Firstly, the data stream should be routed into one directed graph, in which arcs represent number of lines and the vertices represent number of files. Each arc is represented using two things, the total volume and present load of the streams/files, which are derived from different computational application studies. Firstly, we input the data that holds an arc of the graphs and its content. Then the Map operation produces two contents for each arc data, one keyed under each of the vertices that form the arc. The value each record possess augmenting flow $\mathbb{A}$ and path. Each $\mathbb{A}$ is equal to total volume minus present load. Post completion of Map operation, a sequence of containers are created, each container possess records for each arc adjacent to its corresponding/neighbouring vertex. In Reduces stage, records with same key generated in Map stage are cumulated by performing shuffle and sort operation to search the path and compute $\mathbb{A}$. Post completion of reduce operation, all the 2-distance paths among paths/node pair are generated. Again the output of previous reduce operation and the raw arc is combined to form input and perform iterative computation. Similarly, the Map operation produces sequences of records and records with same key are send to a specific reduce worker. Post completion of Map phase, all 3-distance paths and 𝔸 are computed. Similarly, the process is repeated. The iteration process is carried out until output of two reduce functions are same. This means 𝔸 and paths among each node pair are enclosed in all the output produced by each round decrease. All we need to do is to sort these outcomes by 𝔸 and node pair.

In next section experiment are conducted to evaluate the performance DIF-AMR over exiting model considering different experiments.

3. RESULT AND ANALYSIS

This section describes the experimental analysis of proposed Data Inflow Forecasting AMR (DIF-AMR) performance achieved over exiting model [10]. The DIF-AMR is implemented using C#, Dot Net framework 4.5 and deployed on Microsoft Azure Cloud computing platform.

a) Word frequency detection computation

The DIF-AMR framework is deployed on D3 VM instance composed of 4 virtual computing cores, 14 GB RAM and 200 GB local SSD hard drive space. The DIF-AMR framework is deployed on Microsoft azure cloud computing platform composed of one master nodes and four virtual computing nodes to perform map and reduce operation which runs on Windows Server 2012 R2 operating system. HDInsight cluster [14], [15] is considered for exiting model on Microsoft azure cloud computing platform. Identical computing platform and configuration is considered for both proposed and existing model. The word frequency statistic application is developed using C# programing language and Dot.net framework 4.5. And for existing Hadoop based model it is designed using Java programing language. The Wikipedia dataset [16] is considered for experiment analysis. The Wikipedia dataset is huge in size (i.e. >100 GB) and is split into 512 MB each and stored in Azure cloud container. For experimental analysis this work consider 8GB of data. The word frequency statistics applications were executed on the DIF-AMR and existing Hadoop deployments and the results obtained are noted.

The Map operation computation time at each computing node is noted and result obtained are shown in Figure 3. From result obtained it can be seen that DIF-AMR achieves faster execution time than exiting model. Reduce operation computation time at each computing node is noted and result obtained are shown in Figure 4. From result obtained it can be seen that DIF-AMR achieves faster execution time than exiting model. From result obtained it can be seen that FIF-AMR exhibits faster computation time than existing Hadoop based model [10]. The average computation time of Map and Reduce operation computation time is noted and is shown in Figure 5. The computation time of Map operation is reduced by 8.16% by DIF-AMR over existing model. The computation time of Reduce operation is reduced by 85.23% by DIF-AMR over existing model. The total computation time for performing work frequency statistics computation on Azure cluster is shown in Figure 6. The computation time of map phase considering DIF-AMR is 100.29 seconds and for existing model is 198.0 seconds. The computing time reduction of 49.34% and 43.36% is achieved by proposed DIF-AMR over existing model for Map and Reduce operation respectively. The total computation time of map phase considering DIF-AMR is 107.05 seconds and for existing model is 210.0 seconds. The total computing time reduction of 49.023% is achieved by proposed DIF-AMR over existing model.

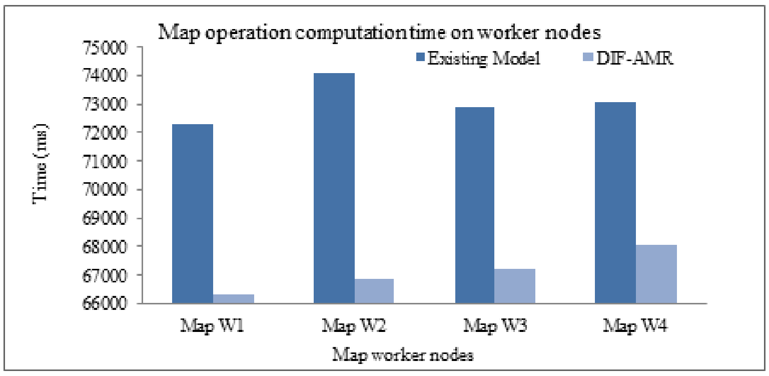

Figure 3. Map operation computation time on worker nodes

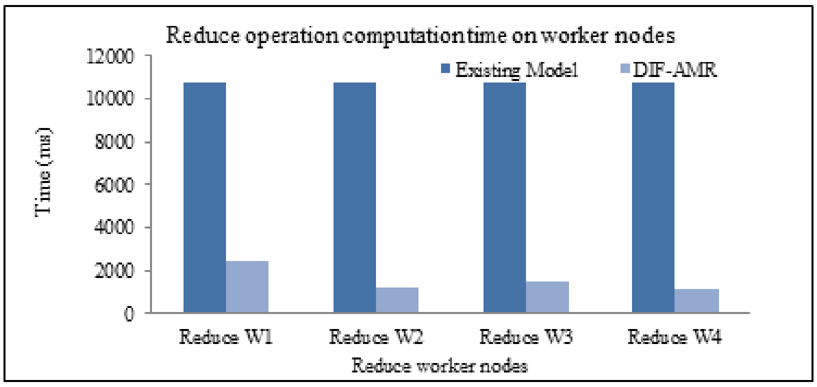

Figure 4. Reduce operation computation time on worker nodes

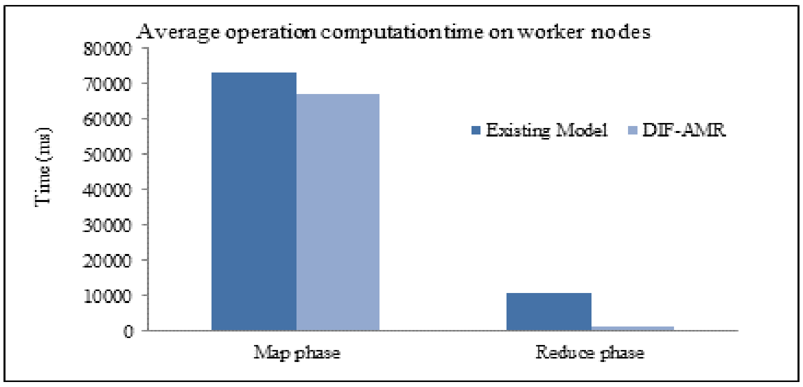

Figure 5. Average computation time on worker nodes on Map and Reduce phase

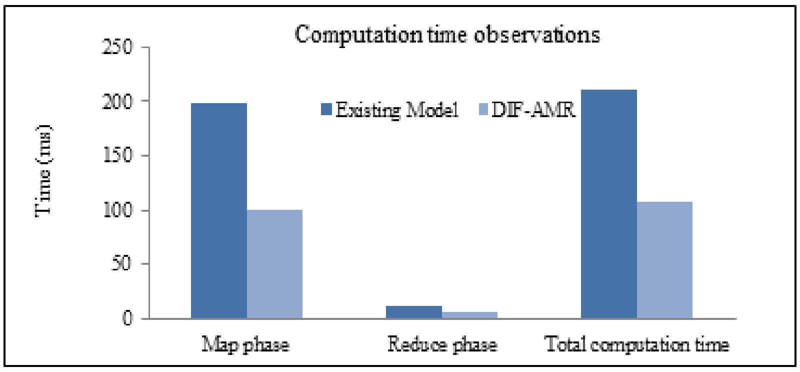

Figure 6. Computation time performance evaluation considering DIF-AMR and existing model

b) Hot-Word detection computation

The DIF-AMR framework is deployed on D3 VM instance composed of 1 virtual computing cores, 14 GB RAM and 200 GB local SSD hard drive space. The DIF-AMR framework is deployed on Microsoft azure cloud computing platform composed of one master nodes and four virtual computing nodes to perform map and reduce operation which runs on Windows Server 2012 R2 operating system. HDInsight cluster [14], [15] is considered for exiting model on Microsoft azure cloud computing platform. Identical computing platform and configuration is considered for both proposed and existing model [17]. The hot-word detection algorithm [18] is developed using C# programing language and Dot.net framework 4.5. And for existing Hadoop based model it is designed using Java programing language. The “Movietweetings” dataset [19] is considered for experiment analysis and stored in Azure cloud container. Tweets consisting of 25000, 50000, 75000 and 100000 movies is considered and is represented as 25K, 50K, 75K and 100K. The hot-word detection algorithm were executed on the DIF-AMR and existing Hadoop deployments and the results obtained are noted. The total computation time of DIF-AMR and existing model is noted and is shown in Figure 7. Experiment analyses shows as number of tweets increases the computation time of both DIF-AMR and existing model increases. The computation performance improvement of DIF-AMR over existing model for 25K is about 10.6%, for 50K is about 16.69%, for 75K is about 58.41% and for 100K is about 57.98% is achieved. An average computation performance improvement of 46.94% is achieved DIF-AMR over existing model. Based on the results obtained in the hot word detection algorithm it can be concluded that the DIF-AMR model exhibits lower computation time when compared to the existing model.

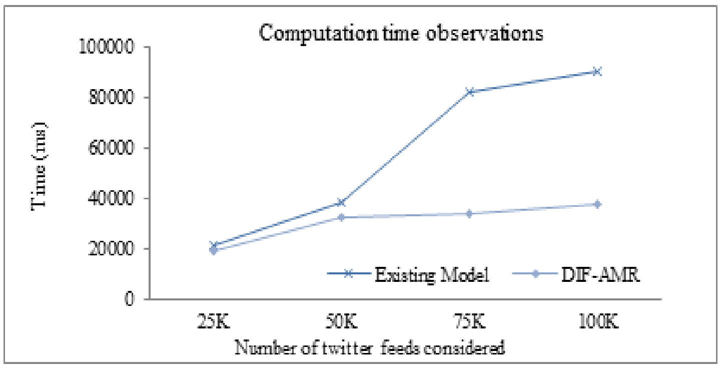

Figure 7. Computation time performance evaluation considering DIF-AMR and existing model

HDInsight cluster [14], [15] is considered for exiting model on Microsoft azure cloud computing platform. Identical computing platform and configuration is considered for both proposed and existing model. The hot-word detection algorithm [18] is developed using C# programing language and Dot.net framework 4.5. And for existing Hadoop based model it is designed using Java programing language. The "Movietweetings" dataset [17] is considered for experiment analysis and stored in Azure cloud container. Tweets consisting of 25000, 50000, 75000 and 100000 movies is considered and is represented as 25K, 50K, 75K and 100K. The hot-word detection algorithm were executed on the DIF-AMR and existing Hadoop deployments and the results obtained are noted. The total computation cost of DIF-AMR and existing model is noted and is shown in Table 2. Experiment analyses shows as number of tweets increases the computation cost of both DIF-AMR and existing model increases. The computation cost improvement of DIF-AMR over existing model for 25K is about 32.67%, for 50K is about 37.34%, for 75K is about 63.05% and for 100K is about 62.23% is achieved. An average computation cost improvement of 48.82% is achieved DIF-AMR over existing model. Based on the results obtained in the hot word detection algorithm it can be concluded that the DIF-AMR model exhibits lower computation cost when compared to the existing model.

Table 2. Cost computation using DIF-AMR technique

Number of Twitter Feeds	Cost computation ($)		Improvement (%)
	Existing	DIF-AMR	
25K	0.4291	0.2889	32.67
50K	0.8460	0.5301	37.34
75K	1.5275	0.5644	63.05
100K	1.6711	0.6311	62.23

In this section the execution of the imprecise applications namely word frequency statistics and hot word detection is presented. The results presented here prove that the DIF-AMR model reduces the computation time observed due to the novel agent based parallel computation method incorporated. An average reduction of 49.02% for word frequency statistics and 46.94% for the hot word detection is reported considering the DIF-AMR model when compared to the existing model [10]. The cumulative analysis over state-of-art technique in Table 2 shows the efficiency of DIF-AMR over state-of-art technique in terms of robustness and scalability. Since, DIF-AMR support parallel processing of stream and non-stream data by adopting MapReduce framework. Adoption of agent based computing aid in fully utilizing system resources. Support accurate processing of all type of data shows robustness of DIF-AMR. Adoption cloud platform aid in proving scalability of processing of large amount of data of various types on large computing clusters. All these feature attributed to the performance improvement of DIF-AMR over existing model, shown in Table 3.

Table 3. Comparison with state of art technique

	[8]	[9]	[10]	DIF-AMR
MapReduce platform considered	Hadoop	Storm	Hadoop	Custom
Agent based	Yes	No	Yes	Yes
Cloud adopted	Yes	Yes	Yes	Yes
Non-stream data processing support	Yes	Yes	Yes	Yes
Stream data processing support	No	Yes	No	Yes
Hybrid processing support	No	Yes	No	Yes
Forecasting accuracy	Yes	No	Yes	Yes

4. CONCLUSION

The main contribution of this work is presenting an agent based scalable stream and non-stream processing of social media and sensor data using MapReduce framework. Efficient data inflow forecasting model adopting MapReduce and cloud platform is presented. Multivariate Gaussian Mixture (MGM) model is designed to improve the accuracy of forecasting for both stream and non-stream data. DIF-AMR overcomes the limitation of state-of-art technique in terms of computation overhead which are experimental proven. Experiment are conducted to evaluate the performance of proposed DIF-AMR over state of art-technique in terms of computation time considering different experiment such as word frequency statistic prediction and hot word detection. Experiment outcomes shows DIF-AMR improves computation time by 49.02% over existing model for word frequency statistics and 46.94% for hot word detection. The experiment outcomes shows significant performance improvement of DIF-AMR over exiting model in terms of computation time considering execution on cloud platform for relatively large stream and non-stream data. Thus DIF-AMR is scalable irrespective of data size and computing cluster size.

The future work would consider performance evaluation considering large dataset and also consider security provisioning for computing data on cloud platforms. This work would further consider optimization of MapReduce scheduler for further reduction of computation time.

REFERENCES

[1] L. Tang et al., ``A framework of traveling companion discovery on trajectory data streams," *ACM Trans. Intell. Syst. Technol.,* vol. 5, no. 1, Art. ID 3, Dec. 2013.
[2] T. Benson, A. Anand, A. Akella, and M. Zhang, *"Micro TE: Fine grained traffic engineering for data centers,"* in Proc. CoNEXT, Art. no. 8, 2011.
[3] K. Chen et al., *"OSA: An optical switching architecture for data center networks with unprecedented flexibility,"* in Proc. NSDI, p. 18, 2012.
[4] A. D. Ferguson, A. Guha, J. Place, R. Fonseca, and S. Krishnamurthi, *"Participatory networking,"* in Proc. Hot-ICE, p. 2, 2012.
[5] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, *"Hedera: Dynamic flow scheduling for data center networks,"* in Proc. NSDI, p. 19, 2010.
[6] A. Curtis, W. Kim, and P. Yalagandula, *"Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection,"* in Proc. IEEE INFOCOM, pp. 1629–1637, 2011.
[7] H. H. Bazzaz et al., *"Switching the optical divide: Fundamental challenges for hybrid electrical/optical datacenter networks,"* in Proc. SOCC, Art. no. 30, 2011.
[8] A. El Fazziki, D. Benslimane, A. Sadiq, J. Ouarzazi and M. Sadgal, "An Agent Based Traffic Regulation System for the Roadside Air Quality Control," in *IEEE Access,* vol. 5, pp. 13192-13201, 2017.
[9] Z. Zhao, W. Ding, J. Wang and Y. Han, "A Hybrid Processing System for Large-Scale Traffic Sensor Data," in *IEEE Access*, vol. 3, pp. 2341-2351, 2015.
[10] Wei Dai, Peng Hu, "Research on Personalized Behaviors Recommendation System Based on Cloud Computing", *TELKOMNIKA Indonesian Journal of Electrical Engineering* Vol.12, No.2, February 2014, pp. 1480 ~ 1486.
[11] Hadoop, http://hadoop.apache.org, accessed Oct. 22, 2017.
[12] Hadoop. [Online]. Available: http://hadoop.apache.org/, accessed Oct. 21, 2017.
[13] Haiwen Han, Weiping Zheng, "A Privacy Data-oriented Hierarchical MapReduce Programming Model", *TELKOMNIKA Indonesian Journal of Electrical Engineering,* Vol. 11, No. 8, August 2013, pp. 4587~4593 e-ISSN: 2087-278X.
[14] Hdinsight (hadoop on azure)," https://www.hadooponazure.com/, accessed Oct. 20, 2017.

[15] Xiaojia Wang, “Electricity Consumption Forecasting in the Age of Big Data”, *TELKOMNIKA Indonesian Journal of Electrical Engineering*, Vol. 11, No. 9, September 2013, pp. 5262~5266 ISSN: 2302-4046.

[16] Kajdanowicz, T.; Indyk, W.; Kazienko, P.; Kukul, J., *"Comparison of the Efficiency of MapReduce and Bulk Synchronous Parallel Approaches to Large Network Processing,"* Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on, pp.218,225, 10-10 Dec. 2012.

[17] Changjian Wang; Yuxing Peng; Mingxing Tang; Dongsheng Li; Shanshan Li; Pengfei You, *"MapCheckReduce: An Improved MapReduce Computing Model for Imprecise Applications,"* Big Data (BigData Congress), 2014 IEEE International Congress on , vol., no., pp.366,373, June 27 2014-July 2 2014.

[18] S. Dooms, T. De Pessemier, and L. Martens, *“Movietweetings: a movie rating dataset collected from twitter,”* in Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys, vol. 13, 2013.

[19] G. Zhai, L. Tian, Y. Zhou, Q. Sun and J. Shi, *"A computing resource adjustment mechanism for communication protocol processing in centralized radio access networks,"* in China Communications, vol. 13, no. 12, pp. 79-89, December 2016.