INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2015; **28**:1610–1619
Published online 2 April 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.2779

A nodes scheduling model based on Markov chain prediction for big streaming data analysis

Qingchen Zhang[1,*,†], Zhikui Chen[1] and Laurence T. Yang[2]

[1]*School of Software Technology, Dalian University of Technology, Liaoning, China*
[2]*Department of Computer Science, St. Francis Xavier University, Antigonish, Canada*

SUMMARY

Streaming data analysis is an important part of big data processing. However, streaming data is difficult to be analyzed and processed in real time because of the rapid data arriving speed and huge size of data set in stream model. The paper proposes a nodes scheduling model based on Markov chain prediction for analyzing big streaming data in real time by following three steps: (i) construct data state transition graph using Markov chain to predict the varying trend of big streaming data; (ii) choose appropriate cloud computing nodes to process big streaming data depending on the predicted result of the data state transition graph; and (iii) assign big streaming data to these computing nodes using the load balancing theory, which ensures that all subtasks are accomplished synchronously. Experiments demonstrate that the proposed scheduling algorithm can fast process big streaming data effectively. Copyright © 2014 John Wiley & Sons, Ltd.

Received 19 June 2013; Revised 6 January 2014; Accepted 19 February 2014

KEY WORDS: big streaming data; cloud computing; Markov chain

1. INTRODUCTION

Recently, we have witnessed efforts to open up the Internet of Things (IoT), which has been used in many application areas, such as smart home, intelligent transportation, and smart grid [1–5]. With the fast development of IoT, sampled data from IoT is now rapidly expanding at unprecedented speed [6]. Besides, the amount of data collected from electronic commerce and scientific computation can exceed hundreds of terabytes every day. For example, Facebook's log exceeds 25 TB of data every day. We have entered the era of big data where data are being fast generated from many different sources such as sensors, mobile devices, and electronic commerce [7–9]. Such big data are difficult to be analyzed in real time with traditional data management methods and infrastructures [10–16].

Currently, there are two computational paradigms for big data analysis, namely, stream data processing and batch data processing. In the batch processing paradigm, sampled data are analyzed and processed after they are organized and stored. In the past few years, batch processing paradigm has been focused on by most of researchers. With the development of IoT, personal computing, and electronic commerce, streaming data analysis has gradually become a hot topic in science and commercial areas. As an important part of big data, streaming data is difficult to be analyzed in real time because of the rapid data arriving speed and huge size of data set in stream model. How to support the real-time processing for big streaming data is a challenging issue, which has sparked great interest of scientists and engineers coming from enterprises, universities, and institutions.

Cloud computing has emerged as a significant technology to deal with big data in time by leveraging vast amounts of computing resources available on demand with low resource usage cost [17–19].

*Correspondence to: Qingchen Zhang, School of Software Technology, Dalian University of Technology, Liaoning, China.
†E-mail: 623759909@qq.com

When deploying a cloud application in a cloud platform, the application user needs to select some cloud nodes to analyze and process the data. How to make optimal deployment of cloud applications is a challenging issue. Most of current methods usually rank the available cloud nodes based on their QoS values and choose the best performing ones. A drawback of the ranking methods is that these methods cannot reflect the communication relations among cloud nodes, especially for the communication-intensive cloud applications, which have a considerable communication delay that can be comparable or even higher than the time required to compute.

To address the drawback, the paper proposes a nodes scheduling model based on Markov chain prediction, which consists of two parts [20]. In the first part, a data state transition graph is constructed using Markov chain to track the historical state, which is used to predict the varying trend of big streaming data. Specially, the state transition graph is used to predict the size of the data. In the other part, a nodes scheduling algorithm combined with the data state transition graph is presented to process big streaming data in real time by following three steps: (i) partition cloud nodes into clusters depending on the communication cost; (ii) explore an appropriate cluster depending on the predicted result; and (iii) dispatch big data analysis to these computing nodes using the load balancing theory in a way to enable synchronized completion at best effort performance.

Finally, a series of experiments is designed to evaluate the performance of the presented algorithm, which demonstrates that the proposed approach outperforms other scheduling methods for big streaming data analysis.

The rest of this paper is structured as follows. Section 2 resumes some related research on nodes scheduling algorithms in cloud computing. The model for predicting the size of data based on Markov chain is presented in Section 3. Section 4 describes the nodes scheduling algorithm based on Markov chain prediction. The conducted experiments with the evaluation results are reported in Section 5. Finally, the conclusion and discussion of this paper are summarized in Section 6.

2. RELATED WORK

Nodes scheduling is an important issue in cloud computing. Many nodes scheduling algorithms have been proposed in the past few years. The most representative approach may be the nodes scheduling based on QoS ranking, which have been employed in RIDGE and GridEigenTrust. Other typical nodes scheduling algorithms include random-based nodes scheduling schemes, which choose cloud nodes randomly to analyze and process big data, and matching approaches, which are used to compare the users' requirements and the QoS values of cloud nodes [21].

Most of these methods, which take only the nodes' QoS performance into consideration, do not consider the communication cost among cloud nodes, which has important effects on the communication-intensive cloud application, because they have a considerable communication delay that can be comparable or even higher than the time required to compute.

To tackle this issue, the paper proposes a nodes scheduling algorithm based on Markov chain prediction to provide optimal deployment for the communication-intensive cloud application.

3. MODEL FOR PREDICTING THE SIZE OF DATA BASED ON MARKOV CHAIN

The global state information of streaming data plays an important role on processing big streaming data effectively. In order to record the data state information, the paper uses Markov chain to construct a data state transition graph, which keeps the varying trend of big streaming data. In the data state transition graph, the data size is mapped to the state space, whose varying trend is viewed as a continuous process of the state transition. As time goes on, the states of the data are kept in the state transition graph. The amount of sampled data in the next moment can be predicted by the statistical regularity of the state transition.

3.1. Model Definition

Definition 1. (d_t chain) Given that X_t represents the amount of the data in the time window whose size is $|W|$ and t represents the critical moment of every two time windows, the mapping function is defined as Eq. (1).

DOI: 10.1002/dac

$$d_t = f(X_t) = \begin{cases} 0 & X_t < 2 \\ \lfloor \log_2(X_t) \rfloor & X_t \geq 2 \end{cases} \tag{1}$$

For each time window, d_t is a random variable. $\{d_t, t \in T\}$ can be approximately regarded as a Markov chain, defined as a d_t chain.

Theorem: If D is the state space of d_t chain, then D is a finite set and $D \subset N$.

Proof: According to the definition, X_t represents the amount of the data in the time window $|W|$, so there must be a variable $\varepsilon \in R$, making $0 \leq X_t \leq \varepsilon$ for every moment t.

Given $g(X_t) = \log_2(X_t), 2 \leq X_t \leq \varepsilon$, the derivative function of $g(X_t)$ is as follows:

$$g'(X_t) = \frac{1}{X_t \ln 2} > 0 \quad 2 \leq X_t \leq \varepsilon \tag{2}$$

According to Eq. (2), $g(X_t)$ is a monotonically increasing function, so $g(X_t)$ is bounded and Eq. (3) can be obtained.

$$\min\{g(X_t)\} = 1, \ \max\{g(X_t)\} = \log_2 \varepsilon \tag{3}$$

According to Eq. (3), $f(X_t)$ is also bounded, and for every $\varepsilon \geq 2$, the formula (4) can be obtained.

$$\min\{f(X_t)\} = 0, \ \max\{f(X_t)\} = \lfloor \log_2 \varepsilon \rfloor \tag{4}$$

Given that $f(X_t)$ is a rounded down function, we define that $N = \lfloor \log_2 \varepsilon \rfloor$ when $\varepsilon \geq 2$.

In summary, the state space (*SSpace*) of the d_t chain is $D = \{0,1,2,.....,N\}$, so $D \subset N$ and $|D| = N + 1$ ($|D|$ is the size of *SSpace*).

Definition 2. State transitionIn the moment t_i, the state transition of the variable d_t is defined as $T_i = \{T_{ij} | T_{ij}(d_i, d_j, p_{ij}), i, j \in S\}$, which represents all possible transitions of d_t in the moment t_i, where d_i is the current value of d_t chain in the moment t_i, d_j is the value of d_t chain in the next moment t_j, and p_{ij} are the probabilities of d_t from d_i to d_j.

If t_i and t_j are adjacent in the time, p_{ij} is the transition probability of one step, and T_i is the state transition of one step of d_t, which can be represented as Eq. (5).

$$T_i = \begin{bmatrix} T_{00} & T_{01} & \cdots & T_{0N} \\ T_{10} & T_{11} & \cdots & T_{1N} \\ \cdots & \cdots & \cdots & \cdots \\ T_{N0} & T_{N2} & \cdots & T_{NN} \end{bmatrix} \tag{5}$$

If the state of d_t is d_i in the moment t_0 and the state is d_j in the moment $t_0 + m$, which represents the state of d_t after m step state transition, and the probability is represented as p_{ij}^m. The statistic value of d_j can be obtained according to Eq. (6).

$$\overline{d_j} = \sum_{j=1}^{|D|} d_j \times p_{ij}^m \tag{6}$$

Definition 3. (STSpace) The set, which consists of all the possible state transition of the d_i chain, is called the state transition space with the abbreviation of *STSpace*. *STSpace* can be outlined as a graph, called *STG*, shown as Figure 1.

In the state transition graph, a vertex denotes a state, and an edge represents a state transition. If the maximum value of d_t is N, the size of the state space is not more than $N + 1$, and the size of *STSpace* is not more than $(N + 1)^2$.

DOI: 10.1002/dac

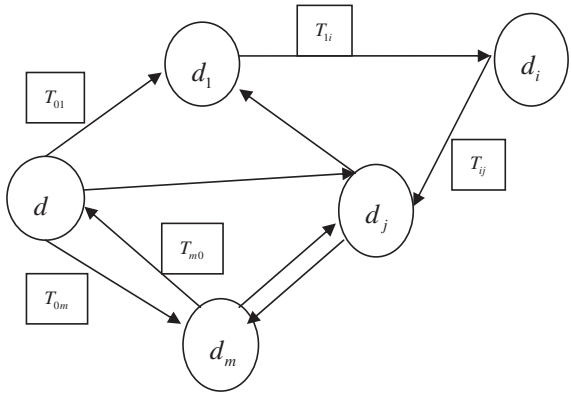

Figure 1. The state transition graph (*STG*).

3.2. Prediction algorithm for the size of data based on Markov chain

In the prediction model for the size of data based on Markov chain, there are two issues required to address. One is to calculate the state transition probability of one step, p_{ij}, and the other is to predict the size of data in the next moment. To tackle these two issues, the paper first maintains the *STG*.

The state transition of the data is kept in the *STG*, and the state transition probability of one step is calculated depending on *STG*.

If the state of the data is d_i in the moment t_i, assume that the state of the data is X_j in the next moment $t_i + 1$. For maintaining *STG*, the paper defines four operations for *STG* in the following text.

(1) Query the state d_j in the *STG*.
(2) Insert a new state into the *STG*. If the state d_j is not in the *STG*, a new state d_j is created and then inserted into the *STG*. Then, the state transition T_{ij} is inserted into the *STSpace*.
(3) Update *STG*. If a new state d_j is inserted in the *STG*, update the counter of d_j and T_{ij}.
(4) Delete a state from *STG*. If a state does not appear in a very long period, remove this state and the relevant state transitions from *STG*.

The updating algorithm is as Algorithm 1.

Algorithm1. Algorithm of updating *STG*

Input: The state d_j
Output: The state transition graph *STG*

```
1   if(find(dj)){
2         dj.count++;
3         Tij.count++;
4   }//end of line 1
5   else{
6         create a new state dj;
7         insert dj into SSpace;
8         insert Tij into SSpace;
9         set dj.count=1;
10        set Tij.count=1;
11         }//end of line 5
12  return STG;
```

The time complexity of querying a state in the state space is $O(N)$, and the complexity of inserting a state into the *STG* is $O(1)$. Therefore, the time complexity of the updating algorithm is $O(N)$.

DOI: 10.1002/dac

p_{ij} can be calculated according to Eq. (7).

$$p_{ij} = \frac{T_{ij}.count}{S_i.count} \tag{7}$$

Where $T_{ij}.count$ represents the times of the state transition T_{ij} from the state d_i to the state d_j, and $S_i.count$ denotes the times of the state transition at the moment t_i.

The algorithm for predicting the size of the data is outlined as Algorithm 2.

Algorithm2. Algorithm for predicting the size of the data

```
     Input: The state transition graph STSpace
     Output: The prediction value d
1    pij= Φ; d=0;
2    for(each state di ∈SSpace){
3        for(each state di ∈SSpace){
4            if(Tij.count>0){
5                calculate pij based on formula (7);
6            }//end of line 4
7        }//end of line 3
8    }//end of line 2
9    calculate the prediction d based on formula (6);
10   return d;
```

In every moment, the time complexity of computing p_{ij} is $O(N^2)$, and the time complexity of predicting $\underline{d}$ is $O(1)$, so the time complexity of the predicting algorithm is $O(N^2)$.

4. NODES SCHEDULING ALGORITHM BASED ON MARKOV CHAIN PREDICTION FOR BIG DATA

After data are collected, cloud users must explore many optimal cloud nodes from the cloud platform to deploy their cloud applications. In order to optimize a parallel data analysis and process, this paper addresses (i) node selection, that is, 'how many' and 'which' computing nodes in cloud should be used and (ii) data partition and synchronized completion, that is, how to optimally apportion big data across parallelized computation environments to ensure synchronization, where synchronization refers to completing all workload portions at the same time even when resources and inter-networks are heterogeneous and situated in multiple Internet-separated clouds [21].

To address these problems, the paper first clusters the cloud nodes into groups to make the communication cost minimally and then select one cluster for big streaming data analysis depending on the predicted result.

Assume that there are n cloud nodes in a cloud platform, the response time between different nodes can be represented as an n by n matrix (8), where q_{ij} represents the response time between node i and node j. Apparently, this matrix is symmetric.

$$Q = \begin{pmatrix} 0 & q_{12} & \cdots & q_{1n} \\ q_{21} & 0 & \cdots & q_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ q_{n1} & q_{n2} & \cdots & 0 \end{pmatrix} \tag{8}$$

q_i represents the vector of response time from node i to other nodes, that is, $q_i = (q_{i1}, q_{i2}, \ldots, q_{i,i-1}, q_{i,i+1}, \ldots, q_{i,n})$.

DOI: 10.1002/dac

In this paper, the response time is used to represent the distance between two nodes. If a node has lower response times to other nodes, it means that the node has shorter distances to the other nodes.

Equation (9) is used to calculate the distance D between a node and the center of the k-th clustering.

$$D = \lambda \times |cal_i - cal_{ck}| + (1 - \lambda) \times \frac{1}{d} \sum_{j:j \in C_k} p_{ij} \tag{9}$$

where cal_i is the computing capacity of the node i and cal_{ck} represents the computing capacity of the center of k-th clustering.

The clustering algorithm is outlined as shown in Algorithm 3, which includes the following three steps:

Step 1. Select randomly K nodes as the initial centers that are stored in F.
Step 2. Calculate the distance between every cloud node and each cluster and assign it to the closest cluster e.
Step 3. Calculate the new center of every cluster repeatedly and repeat steps 2 and 3 until the center of every cluster do not change.

Algorithm 3: The algorithms of cluster analysis

Input: Cloud nodes set I
Output: K clusters

```
1   F= Φ;
2   select centroids to F,|F|=K;
3   C=F;
4   repeat
5   for(each i ∈I){
6           e=arg min_{j∈C} dist(p_i,c_j);
7           e=arg min_{j∈C} dist(p_i,c_j);
8           e=e+{p_j};
9         }
10  for(each j ∈C){
11          c_j=avg(c_j);
12        }
13  until centroid of every cluster does not
    change;
```

After clustering, the nodes are partitioned into several clusters, one of which is explored for big streaming data analysis and process according to the predicted size of data $\underline{d}$. Specifically, the cluster with strong computing power is chosen for data with large size $\underline{d}$.

In order to make each node in the same cluster complete the tasks synchronously, the paper uses the load balancing theory to deploy the input data into those chosen computing nodes.

If the average completion time required by the node i for finishing a unit data is denoted as t_i, which includes two parts, namely, data transmission time and data processing time. And then the overall delay for executing a data size s_i (that is provided to node i for processing task) is $s_i\ t_i$. In order to ensure ideal parallelization for n nodes and a set of data, Eq. (10) is satisfied [11].

$$T = s_1 t_1 = s_2 t_2 = \ldots = s_n t_n \tag{10}$$

According the prediction algorithm of the data size, d' is the total size of the data, and the following formula can be obtained.

$$T = \underline{d} \Big/ \sum_{i=1}^{n} \frac{1}{t_i} \tag{11}$$

DOI: 10.1002/dac

From the aforementioned formulas, Eq. (12) must be satisfied.

$$s_i = \underline{d}/t_i \sum_{i=1}^{n} \frac{1}{t_i} \tag{12}$$

Depending on Eq. (12), the paper finds the optimal data partitioning solution.

5. EXPERIMENT

In this section, a series of experiments are carried out to evaluate the performance of the presented algorithm. There are 20 distributed nodes as cloud computing nodes in the cloud platform, each of which has a 2.8 GHz core, 1 GB memory, and 250 GB hard drive. The data in experiments are sampled from the digital home lab, including three sets of data: temperature, humidity, and carbon dioxide concentration, which the total size is up to 80 GB.

5.2. Performance analysis for the prediction algorithm for the data size

In order to evaluate the performance of the prediction algorithm, the paper introduces two performance evaluations, namely, the average relative error (AVE) and prediction success rate. AVE is defined as Eq. (13).

$$mean = \frac{1}{n} \sum_{j=1}^{n} \left| \frac{\underline{d_j} - d_j}{d_j} \right| \tag{13}$$

Wherein n is the number of prediction, $\underline{d_j}$ is the predicted value, and d_j is the real value. Prediction success rate is defined as follows.

Define 6 (SRatio) Given $\varepsilon > 0$, if $\underline{d_j}$, which is the predicting value of d_j, meets Eq. (14), it is predictive success.

$$\left| \frac{\underline{d_j} - d_j}{d_j} \right| < \varepsilon \tag{14}$$

Prediction success rate is defined as $Sratio = t/n$, wherein t is the number of prediction success and n is the total number of prediction, $\varepsilon = 5\%$ in this paper

Figure 2 describes the result of AVE of the prediction algorithm. The state transition space is updated frequently early in the forecast, so the prediction average error is high. As the number of prediction increases, state transition space stabilizes gradually, so the prediction AVE decreases. When the times of prediction are more than 700, the state transition space becomes stable, and the predicted AVE can be close to 0.

Next, the performance of the prediction algorithm is evaluated by comparing the proposed algorithm with multivariable regression algorithm [14] and improved expectation prediction maximization algorithm [15] in prediction accuracy. The result is described in Figure 3. Multivariable regression

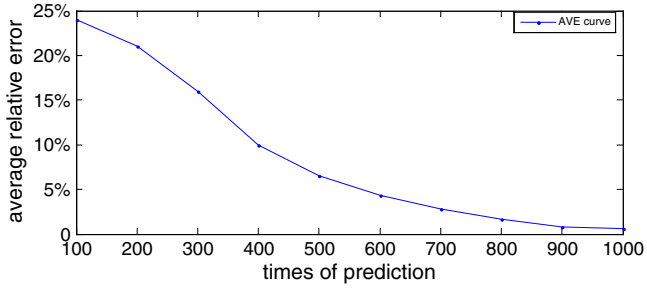

Figure 2. Prediction result.

DOI: 10.1002/dac

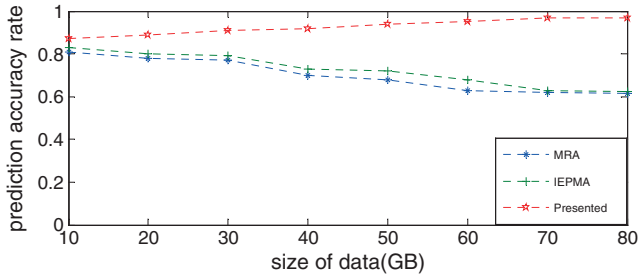

Figure 3. Prediction accuracy rate curve.

algorithm and improved exact pattern matching algorithm are both based on regression model, and the two methods are effective in prediction efficiency when the amount of data is small. As the data increases, the efficiency and accuracy of the two methods become lower gradually. In contrary, the presented algorithm obtains better performance, especially for big streaming data.

5.3. Impact of data transfer delay on overall execution time

In order to evaluate the performance of the presented scheduling algorithm, the performance characteristics of computing nodes in the context of data computation time and data transfer delay are analyzed first. Figure 4 describes the impact of the data transfer delay on the overall execution time.

In Figure 4, the first bar represents the execution time with one virtual machine, and the second bar represents the execution time with four virtual machines in parallel, all of which are in the same computer. From these two bars, the first bar is higher than the second bar, which explains the need of parallel execution of big streaming data analysis to improve the performance.

The third bar represents the execution time with four different machines in parallel, which is higher than the second bar because of a significant delay of big data to get transferred over the different computing nodes. Therefore, we have to deal with the data transfer delay carefully.

5.4. Performance comparison of the scheduling algorithm with other methods

To evaluate the performance of our presented method, the proposed scheduling algorithm is compared with the following scheduling algorithms.

The QoS-based scheduling algorithm. QoS-based scheduling algorithm ranks cloud nodes depending on the QoS of the nodes.

Random-based scheduling algorithm. Random-based scheduling algorithm uses random methods to select components.

Figure 5 describes the result of the performance comparison of all the algorithms.

In this experiment, cloud nodes are partitioned into three clusters, and the parameter is set to $\lambda = 0.5$. The result demonstrates that the execution time of all the algorithms increases as the data become big. The execution time of the random-based node scheduling algorithm and QoS-based scheme are lower when the size of data is smaller than 30 GB, because computation time occupies most of the overall execution time. However, when the size exceeds 40 GB, the transfer time

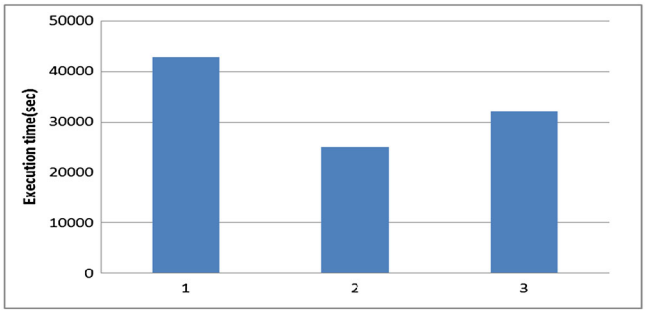

Figure 4. Performance characteristics of computing nodes.

DOI: 10.1002/dac

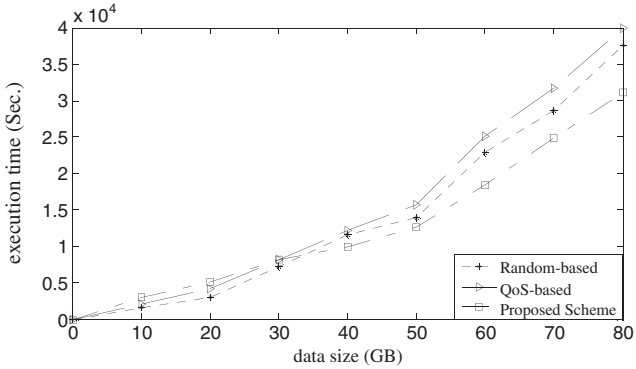

Figure 5. Performance comparison of the algorithms.

increases rapidly, and the proposed method obtains better performance than other schemes that cannot consider the communication relations between cloud nodes. In conclusion, in most cases, the proposed algorithm obtains the best performance.

6. CONCLUSION AND DISCUSSION

With the development of IoT, electronic commerce, and personal computing, how to support the real-time analysis and process for high speed streaming data is a significant issue. The paper has proposed a novel scheduling model based on Markov chain prediction. The model consists of two parts: (i) constructing a data state transition graph using Markov chain to predict the varying trend of big streaming data and (2) designing a node scheduling algorithm based on the data state transition to choose appropriate nodes from the cloud computing platform for big streaming data analysis. Experiments demonstrate the effectiveness of the proposed scheduling algorithm, especially for communication-intensive cloud applications.

In fact, data state can refer to many statistical characteristics of the data in a period, such as the amount of the data, the average value of the data, and the variance of the data. In the paper, we take the size as an example to demonstrate the performance of the proposed prediction model based on Markov chain. In the further research work, we will take many profiles into consideration. Additionally, in the practical applications, the response time between different nodes is affected by multiple factors, such as processing speed and data size. According to [14], the paper defines the response time between node i and node j as the communication time spent on sending a unit data from node i to node j. In the further research work, we will investigate other factors that affect the response time between different nodes to demonstrate the performance of the proposed model.

REFERENCES

1. Atzori L, Iera A, Morabito G. The Internet of Things: a survey. *Computer Networks* 2010; **54**(15):2787–2805.
2. Feki MA, Kawsar F, Boussard M, Trappeniers L. The Internet of Things: the next technological revolution. *Computer* 2013; **46**(2):24–25.
3. Zhang Z, Lu Z, Chen Q. Code division multiple access/pulse position modulation ultra-wideband radio frequency identification for internet of things: concept and analysis. *International Journal of Communication Systems* 2012; **25**(9):1103–1121.
4. Ning HS, Hu S. Technology classification, industry, and education for future Internet of Things. *International Journal of Communication Systems* 2012; **25**(9):1230–1241.
5. Liu Y, Chen Z, Xia F. An integrated scheme based on service classification in pervasive mobile services. *International Journal of Communication Systems* 2011; **25**(9):1178–1188.
6. Kortuem G, Kawsar F, Fitton D, Sundramoorthy V. Smart objects as building blocks for the Internet of Things. *IEEE Internet Computing* 2010; **14**(1):44–51.
7. Wang Y, Shi P, Li K, Chen Z. An energy efficient medium access control protocol for target tracking based on dynamic convey tree collaboration in wireless sensor networks. *International Journal of Communication Systems* 2012; **25**(9):1139–1159.
8. Yang K, Cheng X, Hu L, Zhang J. Mobile social networks: state-of-the-art and a new vision. *International Journal of Communication Systems* 2012; **25**(10):1245–1259.

DOI: 10.1002/dac

9. Zhang Q, Chen Z, Zhao L. Multi-node scheduling algorithm based on clustering analysis and data partitioning in emergency management cloud. *Lecture Notes in Computer Science* 2013; **7901**:165–174.
10. Qin X, Wang H, Du X. Big data analysis—competition and symbiosis of RDBMS and MapReduce. *Journal of Software* 2012; **23**(1):32–45.
11. Jung G, Gnanasambandam N, Mukherjee T. Synchronous parallel processing of big-data analytics services to optimize performance in federated clouds. *Proceedings of the 5th International Conference on Cloud Computing*, Honolulu, HI, USA, 2012; 811–818.
12. Cheng Y, Qin C, Rusu F. GLADE: big data analytics made easy. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, Scottsdale, Arizona, USA, 2012; 697–700.
13. Chiang RHL, Paulo G, Edward AS. Business intelligence and analytics education, and program development: a unique opportunity for the information systems discipline. *ACM Transactions on Management Information Systems* 2012; **3**(3):121–134.
14. Agrawal D, Das S, El Abbadi A. Big data and cloud computing: current state and future opportunities. *Proceedings of the 14th International Conference on Extending Database Technology*, Uppsala, Sweden, 2011; 530–533.
15. Meng X, Ci X. Big data management: concepts, techniques and challenges. *Journal of Computer Research and Development* 2013; **50**(1):146–169.
16. Birney E. The making of ENCODE: lessons for big-data projects. *Nature* 2012; **489**:49–51.
17. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stocia I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53**(4):50–58.
18. Iosup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T, Epema DHJ. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**(6):931–945.
19. Rimal BP, Choi EA. A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing. *International Journal of Communication Systems* 2012; **25**(6):796–819.
20. Fei C, Liu T, Lampropoulos GA, Anastassopoulos V. Markov chain CFAR detection for polarimetric data using data fusion. *IEEE transactions on Geoscience and Remote sensing* 2012; **50**(2):397–408.
21. Fan P, Wang J, Zheng Z, Lyu MR. Toward optimal deployment of communication-intensive cloud applications. *Proceedings of the 4th International Conference on Cloud Computing*, Washington, DC, USA, 2011; 460–467.

DOI: 10.1002/dac