Rajesh Kumar Shukla ·
Jitendra Agrawal · Sanjeev Sharma ·
Geetam Singh Tomer *Editors*

Data, Engineering and Applications

Volume 2

Springer

Data, Engineering and Applications

Rajesh Kumar Shukla · Jitendra Agrawal ·
Sanjeev Sharma · Geetam Singh Tomer
Editors

Data, Engineering and Applications

Volume 2

Springer

Editors
Rajesh Kumar Shukla
Department of Computer Science
and Engineering
Sagar Institute of Research & Technology
(SIRT)
Bhopal, Madhya Pradesh, India

Sanjeev Sharma
School of Information Technology
Rajiv Gandhi Technological University
Bhopal, Madhya Pradesh, India

Jitendra Agrawal
School of Information Technology
Rajiv Gandhi Technical University
Bhopal, Madhya Pradesh, India

Geetam Singh Tomer
THDC Institute of Hydropower
Engineering and Technology
Tehri, Uttarakhand, India

ISBN 978-981-13-6350-4 ISBN 978-981-13-6351-1 (eBook)
https://doi.org/10.1007/978-981-13-6351-1

Library of Congress Control Number: 2019931523

© Springer Nature Singapore Pte Ltd. 2019
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Contents

About the Editors

Dr. Rajesh Kumar Shukla is a Professor and Head of the Department of Computer Science and Engineering, SIRT, Bhopal, India. With more than 20 years of teaching and research experience he has authored 8 books and has published/presented more than 40 papers in international journals and conferences. Dr. Shukla received an ISTE U.P. Government National Award in 2015 and other various prestigious awards including some from the Computer Society of India. His research interests include recommendation systems and machine learning. He is fellow of IETE, a senior member of IEEE, a life member of ISTE, ISCA, and a member of ACM and IE(I).

Dr. Jitendra Agrawal is a faculty member in the Department of Computer Science & Engineering, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India. His research interests include data mining and computational intelligence. He has authored 2 books and published more than 60 papers in international journals and conferences. Dr. Agrawal is a senior member of IEEE, life member of CSI, ISTE and member of IAENG. He has served as a part of the program committees for several international conferences organised in countries such as the USA, India, New Zealand, Korea, Indonesia and Thailand.

Dr. Sanjeev Sharma is a Professor and Head of the School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, MP, India. He has over 29 years of teaching and research experience and received the World Education Congress Best Teacher Award in Information Technology. His research interests include mobile computing, ad-hoc networks, image processing and information security. He has edited proceedings of several national and international conferences and published more than 150 research papers in reputed journals. He is a member of IEEE, CSI, ISTE and IAENG.

Dr. Geetam Singh Tomer is the Director of THDC Institute of Hydropower Engineering and Technology (Government of Uttarakhand), Tehri, India. He received the International Plato award for Educational Achievements in 2009. He completed his doctorate in Electronics Engineering from RGPV Bhopal and post-doctorate from the University of Kent, UK.

Dr. Tomer has more than 30 years of teaching and research experience and has published over 200 research papers in reputed journals, as well as 11 books and 7 book chapters. He is a senior member of IEEE, ACM and IACSIT, a fellow of IETE and IE(I), and a member of CSI and ISTE. He has also edited the proceedings of more than 20 IEEE conferences and has been the general chair of over 30 Conferences.

Part I
Big Data and Cloud Computing

Efficient MapReduce Framework Using Summation

Sahiba Suryawanshi and Praveen Kaushik

1 Introduction

BigData can be defined as huge quantity of data, in which data is beyond the normal database software system tool to capture, analyze, and manage. The data is within the limits of three dimensions which are data volume, data variety, and data velocity [1]. Primary analysis data contains surveys, observations, and experiences; and secondary analysis data contains client information, business information reports, competitive and marketplace information, business information, and location information that contains mobile device information. Geospatial information and image information contains a video and satellite image and provides chain information containing rating and vendor catalogs, to store and process this information that is done by BigData. To process this variety of data, the velocity is incredibly necessary.

The major challenge is not to store the big datasets in our systems, but, to retrieve and analyze the large data within the organizations, that too, for information stored in various machines at completely different locations [1]. Hadoop comes in a picture in these situations. Hadoop has been adopted by many people leading companies, for example, Yahoo!, Google, and Facebook along with various BigData programs, for example, machine learning, bioinformatics, and cybersecurity. Hadoop has the power to analyze the info very quickly and effectively. Hadoop works best on semi-structured and unstructured data. Hadoop has MR and Hadoop distributed file system [2]. The HDFS can provide a storage for clusters, and once the info is stored within the HDFS then it breaks into number of small pieces and distributes those small items into number of servers that are present within the clusters, wherever each server stores

S. Suryawanshi (✉) · P. Kaushik
Department of Computer Science and Engineering, Maulana Azad National Institute of Technology (MANIT), Bhopal, India
e-mail: sahiba686@gmail.com

P. Kaushik
e-mail: kaushikp@manit.ac.in

© Springer Nature Singapore Pte Ltd. 2019

R. K. Shukla et al. (eds.), *Data, Engineering and Applications*,
https://doi.org/10.1007/978-981-13-6351-1_1

these small pieces of whole information set and then for each piece of information a copy stored on more than one server, this copied information set will be retrieved once the MR is process and within which one or a lot of *Mapper* or *Reducer* fails to process [3].

MR appeared as the preferred computing framework for large processing because of its uncomplicated programming model and the execution is done in parallel automatically. MR has two computational phases, particularly *mapping* and *reducing*, that is successively carried through many *maps* and *reduce* tasks unalike. The *map* reads input data and manages to create <*key, value*> pairs depending on the input data. This <*key, value*> pairs are the intermediary outputs within the native machine. Within the *map* phase, the tasks begin in parallel which generates <*key, value*> pairs of intermediate data by the input splits. The <*key, value*> pairs are kept on the native (local) machine and well ordered into various data partitions as one for each *reducer* phase. *Reducer* is liable for processing the intermediary outcomes which receive from various *mappers* and generating ultimate outputs within the *reducer* phase. Every *reducer* takes their part of information from data partitioning coming from all the *mapper* phases to get the ultimate outcome. In between the *mapper* phase and the *reducer* phase, there is a phase, i.e., *shuffle phase* [4]. During this, the info created at the *mapper* phase is ordered, divided, and shifted to the suitable machine execute the *reducer* phases. The MR is performed over a distributed system composed of the master and a group of workers. The input is split into chunks that are allocated to the *mapper* phases [5]. The *map* tasks are scheduled by a master to the workers, which consider the data locality. The *map* tasks provide an output which will be divided into a number of pieces according to the number of *reducers* for the job. Record with the similar intermediary *key* should go to the same partition so that it will guarantee the correctness for the execution. All the intermediary <*key, value*> pairs' partitions are arranged and delivered with the task of *reducer* that needs to be executed. By default, the constraint of data locality does not take into consideration while doing the scheduling tasks for the *reducer*. As the result, the quality of data at the *shuffle* phase, which needs to transfer via a network, is significant. Using tradition, a hash-based function which is used for partitioning the intermediary data in *reducer* tasks is not traffic-efficient because topologies of network and size of data corresponding to each key are not considering it. Thus, this paper proposes a scheme which sums up the intermediate data. The proposed scheme will reduce the data size that has to be sent to the *reducer*. By reducing the size of data, the network traffic at reduce function will minimize. Even though the combiner also performs the same function, the combiner operates on the generated data by *map* task individually which thus fails to operate between the multiple tasks [4]. For summing up the data, it put summation function. Summation function can be put in both either within the single machine or among different machines. For finding the best suitable place for summation function, it uses distributed election algorithm. At the *shuffle* phase, the summation function will work simultaneously, and then it removes the user-irrelevant data. In this, the data of volume and traffic is reduced up to 55%, and then it sends to the *Reducer*. It is more efficient way to process the data, for those jobs which have hundreds and thousands of key ends, and each of the keys is associated with number of values.

The rest of the paper is organized as follows. In Sect. 2, we review recent related work. Section 3 provides the proposed model. Section 4 analyzes the result. Finally, Sect. 5 concludes the paper.

2 Literature Survey

In this section, different techniques for optimization generally applied in the *MapReduce* framework and BigData are discussed. The paper also discussed the attributes of various techniques for optimization and how BigData processing is improved by these techniques.

In [6], the author examined that whether the network optimizing can make a better performance of the system and realize that utilizing the high network and low congestion in a network; good performance can also be achieved parallel with a job in the optimizing network system. In [7], the author gives purlieus, a system which allocates the resources in MR, which will increase the MR Job's performance in the cloud, via positioning intermediary data to the native machines or nearer to the physical machines. This reduces the traffic of data within the *shuffle* part produced in the cloud data center. In [8], paper designs a good key partition approach, in which the distribution of all frequencies of intermediate keys is watched, and it will guarantee that the fair distribution in *reduce* tasks, in which it inspects the partitioning of intermediary key and distribution of data with the key and its respective value among all the machines of *map* to *reducer*, for the data correctness, is also examined. In [9], the author gives two effective approaches (load balancing) to skew the data handling for MR-based entity resolution. In [10], the author proposes MRCGSO mode; it adjusts very well with enlarging data set sizes and the optimization for speedup is very close. In [11], the author relates the parallel and distributed optimization algorithm established on alternating direction method of a multiplier for settling optimization problem in adapting communication network. It has instigated the authorized framework of the extensive optimization problem and explains the normal type of ADMM and centers on various direct additions and worldly modifications of ADMM to tackle the optimization problem. In [12], the author pays attention to accuracy using cross-validation; the paper gives sequential optimize parameters for plotting a conspiracy of accuracy. In [13], the author gives a method to initiate Locality-Sensitive Bloom Filter (LSBF) technique in BigData and also discusses how LSBF can be used for query optimization in BigData. In [14], the author initiates the optimization algorithms using these rules and models can deliver a moderate increase to the highest productivity on inter-cloud and intra-cloud transfers. Application-level transfer adapts parameters just like analogousness pipelining, and concurrency is very needful mechanisms for happening across data transfer bottlenecks for scientific cloud applications, although their optimal values juncture on the environment on which basis the transfers are generally done. By using actual models and algorithms, these can spontaneously be optimized to achieve maximum transfer rate. In [15], the author offers an algorithm for cache consistency bandwidth

optimization. In this perspective, the user data shift is optimized without consideration of the user expectation rate. The debated algorithm differentiates with trending data transmission techniques. In [16], the author recommends improved computing operators focused on smart grids optimization for BigData through online. This settles a problem of generic-constrained optimization by utilizing a module based on the MR framework, which is a trending computing platform for BigData management commenced at Google. Online answer to urged optimization problems is a necessary requirement for a secure, reliable smart grid operation [17]. Many authors proposed methods for optimizing MR, but very less work is done for optimizing MR by reducing the traffic generated, while the data is sent to *reducer* phase. So we proposed a method which is based on distributed summation.

3 Proposed Method

In Hadoop MR, normally the *reduce* task resides in different machines/racks. As the massive intermediate data goes to *reduce* task, it will create heavy traffic in the network. By analyzing the intermediate output, we saw that there are redundant <*key, value*> pairs. So we can sum up all the similar <*key, value*> pairs before sending it to the *reducer*, which will minimize the quantity of intermediate data. The *summation* can be done in both either within the machine or among different machines.

Summation at single machine: The *summation* function can put at each machine in which it will sum up the similar <*key, value*> pairs within the same machine before it is sent to the *reducer.* As a result, the data from each machine will minimize by *summation* function that will minimize the traffic too.

Summation among different machines: When the *summation* function put at each machine, it will reduce the size of data. But for massive data, it needs many *mappers* and *reducers*. There may chances that at *reducer* there might be number of inputs even though the inputs are already minimized by *summation*. But due to number of *map* functions, it will also create traffic at *reducer*. For that, it will sum up data among different machines.

In Fig. 1a, it needs to send three rows of data, wherein Fig. 1b it needs to send two rows of data to the *summation* function reside in different machines. As a result, if the position of node where the *summation* is done will change, the traffic cost will also change; it is the extra challenge to handle.

Architecture: Hadoop has a master node (Job Tracker), and number of slave nodes (Task Trackers) located on remaining nodes. The job tracker handles all the submitted jobs and takes the decision for scheduling and parallelizing the work across the Hadoop cluster. And the task trackers do the work in parallel, allotted by the job tracker. For summating the intermediate data, it needs two things; one piece of code for summation, i.e., *summation phase,* and a manager who will handle the location of that code. The manager resides in job tracker, which has the information where the *summation code* will place for more efficient processing. This architecture will minimize the network traffic in *shuffle* phase.

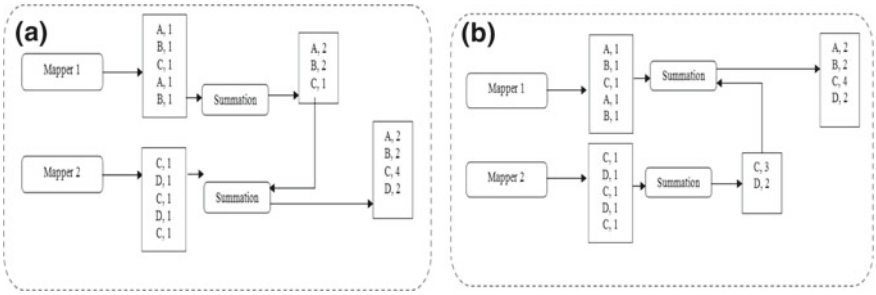


Fig. 1 MapReduce using summation among different machines

Summation phase: In the framework, the *summation phases are* located between *shuffle* phases and *reducer* phases. All the intermediate data acts as input to this and generated output is sent to the *reducer*. It performs the summation of similar <*key, value*> pairs in such a way that each key has single summated pair value. After that, output of *summation phase* along with the similar key has to be delivered to a single *reducer.* In the architecture, the execution of summation is managed by the task tracker at each node. When the manager who is placed in job tracker sends the request to generate the *summation code* at task tracker, task tracker will initialize the instance and specify the tasks attached to the request. Finally, when the task is completed, the *summation code* is removed by task tracker and conveys the message to the manager.

Manager: The manager has two main issues—where the *summation code* resides and routes so that the summated intermediated data will generate less amount of traffic.

Summation code placement—Number of *summation codes* will be generated to reduce the traffic along with the path; the path will define from where the intermediate data will go among different machines. To do this, manager has two main questions; by answering that, it will minimize the traffic during *shuffle* phase:

- On which machine the *summation code* will generate for minimum traffic?
- To which machine the intermediate data come from different machines, i.e., what is the route?

For answering these questions, manager needs the whole information about the *map* and *reduce* function along with the positions and the frequency of intermediate data (volume). Furthermore, manager also requires the information about the resources of slave nodes, i.e., the availability of memory and CPU for processing of summation. All these information will be sent by the task tracker to job tracker with the heartbeat. It is sent by task tracker to job tracker so that job tracker has knowledge of whether the task tracker alive or not; here, alive means its working condition. According to that information, the manager will send the info about the *summation code* and the route. The manager has information about the positioning of all the task trackers, so it will send the request to find the central node for summing

up the data in different machines by creating small clusters. As the slave nodes get the request, one of them elects itself as the central node and finds whether any of other is interested, finds the central node, and informs the manager. Algorithms 1 and 2 are as follows:

Assume the clusters are connected to each other in a ring form; each node can send information to its next node only so that all nodes have the information and it will create less traffic. But if there is any node failure, then it will bypass it.

Distributed algorithm does not assume the existence of the previous central node; every time according to the requirement, it will change which depends on the frequencies of <*key, value*> pairs and the availability of resources. It will choose a node among a group of different nodes in different machines as a central node.

Assume each node has their own IDs, and the priority of node *Ni* is *i*, which is calculated according to the availability of resources and the frequency of <*key, value*> pairs, i.e., volume.

Background: Any node *Ni*(among the nodes to which manager sends the function for processing data) tries to find any other active node which is more suitable, by sending a message; if no response comes in *T* time units, *Ni* tries to elect itself. Details are as follows:

Algorithm 1 for sender node *Ni* that select suitable node

1. *Ni* sends an "*Elect Ni*" with *Pi*
2. *Ni* waits for time *T*
   a. If *Pi* < *Pj* it receives "*elect Nj*"
      i. update central node as *Nj*
   b. If no response comes then *Ni* will be selected as the central node.

Algorithm 2 for node receiver *Nj*

1. *Nj* receive "*elect Ni*"
2. If *Pj* > *Pi*
   a. Send "*elect Nj*"
3. Else forward "*elect Ni*"

4 Implementation

The performance baseline is provided by the original scheme (i.e., no summation is provided) and by the proposed method. We create an Oracle VM virtual machine; we configure it with the required parameters (here we use two processors, 3 GB RAM, 20 GB memory) and settings to act as a cluster node (especially the network settings). This referenced virtual machine is then cloned as many times as there will be nodes in the Hadoop cluster. Only a limited set of changes are then needed to finalize the node to be operational (only the hostname and IP address need to be defined). We have created pseudo-cluster. Our prototype has been implemented on Hadoop 2.6.0.0.

In Fig. 2, it gives output sizes of *mappers* on nodes for respective actual sized. It executes the proposed algorithm using the same data source for comparison in Hadoop environment. It shows the data size for different files of size 250 MB, 500 MB, 750 MB, and 1 GB are minimized after applying the *summation* and the file size reduced to 180 MB, 220 MB, 335 MB, and 450 MB, respectively. As a result, the reduction ratios are 28%, 32%, 48%, and 55% for file size of 250 MB, 500 MB, 750 MB, and 1 GB, respectively. The proposed method will work more efficiently as the size increases, and thus it will work well for BigData.

To compute, the capability of the proposed algorithm by comparing traditional hash-based function is shown here. Hash-based partition without *summation*, as default method in Hadoop, makes the traditional hash partitioning for the intermediate data, which are sent to *reducers* without summing up intermediate data. And our proposed method is *summation* within same machine and *summation* among different machines, in which before sending to *reduce* function the intermediate data will be summed up so that it will minimize the size (traffic); sometimes, after summing up, intermediate data at each machine data at *reducer* will be huge because of many *mappers*, so it can be minimized if the summing up will be done among different machines also as per requirements. In Fig. 3, the performance is shown which takes the same file and performs the traditional method, summation on single machine and *summation* among different machines. Here, if the number of keys is increased, the traffic in *shuffle* phase also increases. For example, for 20 keys, traditional, *summa-*

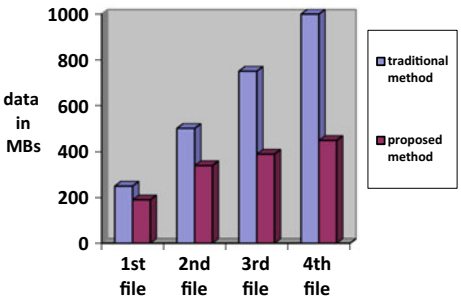


Fig. 2 Data size at reducer

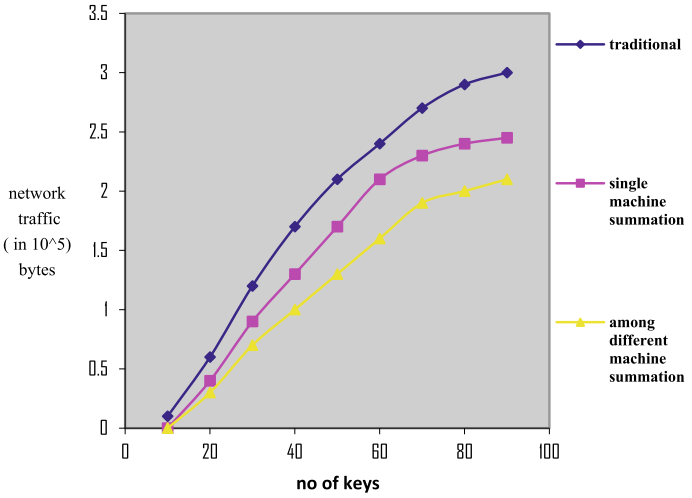


Fig. 3 Traffic cost

tion in single, and *summation* in different machines generate 0.6 * 10^5, 0.4 * 10^5, and 0.3 * 10^5 bytes, respectively.

5 Conclusion

The significance of proposed scheme is discussed, i.e., summation in Hadoop MR to process the BigData that minimizes the network traffic produced by intermediate data: intermediate data is output of *map* function. For verification, we have given an architecture where summation functions can be easily attached to the existing MR framework. How the positioning of summation code among various machines will affect the size is also shown, for which we give a distributed election algorithm that fills to find the best suitable positions for the central summation function among various machines. By applying the proposed method, it will reduce the size of data up to 55%.

The implantation for the proposed scheme is in a pseudo-cluster for computing the behavior; it can compute on heterogeneous distributed clusters.

References

1. Philip Chen, C.L., Zhang, C.-Y.: Data-intensive applications, challenges, techniques and technologies: a survey on big data (2014). Science Direct
2. Apache Haddop HDFS Homepage. http://HADOOP.apache.org/hdfs

3. White, T.: Hadoop: The Definitive Guide, 1st edn. O'Reilly Media (2009)
4. Patel, A.B., Birla, M., Nair, U.: Addressing big data problem using Hadoop and map reduce. IEEE (2013)
5. Ke, H., Li, P., Guo, S., Guo, M.: On traffic-aware partition and aggregation in MapReduce for big data applications. IEEE Trans. Parallel Distrib. Syst. (2015)
6. Blanca, A., Shin, S.W.: Optimizing network usage in MapReduce scheduling (2013)
7. Palanisamy, B., Singh, A., Liu, L., Jain, B.: Purlieus: locality-aware resource allocation for MapReduce in a cloud. ACM (2011)
8. Ibrahim, S., Jin, H., Lu, L., Wu, S., He, B., Qi, L.: Leen: locality/fairness-aware key partitioning for MapReduce in the cloudm. IEEE (2011)
9. Hsueh, S.-C., Lin, M.-Y., Chiu, Y.-C.: A load-balanced MapReduce algorithm for blocking-based entity-resolution with multiple keys (2014)
10. Al-Madi, N., Aljarah, I., Ludwig, S.A.: Parallel glowworm swarm optimization clustering algorithm based on MapReduce. In: 2014 IEEE Symposium on Swarm Intelligence (2014)
11. Liu, L., Han, Z.: Multi-block ADMM for Bigdata optimization in smart grid. IEEE (2015)
12. Liu, Y., Du, J.: Parameter optimization of the SVM for Bigdata. In: 2015 8th International Symposium on Computational Intelligence and Design (ISCID) (2015)
13. Bhushan, M., Singh, M., Yadav, S.K.: Bigdata query optimization by using locality sensitive bloom filter. IJCT (2015)
14. Ramaprasath, A., Srinivasan, A., Lung, C.-H.: Performance optimization of Bigdata in mobile networks. In: 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE) (2015)
15. Ramprasath, A., Hariharan, K., Srinivasan, A.: Cache coherency algorithm to optimize bandwidth in mobile networks. Lecture Notes in Electrical Engineering, Networks and Communications. Springer Verlag (2014)
16. Yildirim, E., Arslan, E., Kim, J., Kosar, T.: Application-level optimization of Bigdata transfers through pipelining, parallelism and concurrency. In: IEEE Transactions on Cloud Computing (2016)
17. Jena, B., Gourisaria, M.K., Rautaray, S.S., Pandey, M.: A survey work on optimization techniques utilizing map reduce framework in Hadoop cluster. Int. J. Intell. Syst. Appl. (2017)

Secret Image Sharing Over Cloud Using One-Dimensional Chaotic Map

Priyamwada Sharma and Vedant Sharma

1 Introduction

Cloud computing has consistently enabled the infrastructure providers to move very rapidly toward new facilities. Such movement has attracted the world toward "pay as you go" model defined by cloud service providers. Almost every task that a user wants to perform is now available over cloud. Such an advancement in cloud computing has made organizations and professionals to migrate their business models and tasks over cloud. However, such swift growth of data toward cloud computing has been raising security and privacy concerns. The users can send their data over cloud for processing and storage without any difficulty at their end, but how secure is the data to be stored over cloud? Such questions are raising a major difficulty in the realization of cloud computing and it is needed to be considered solemnly.

Cryptography is the science of converting plain text to another form which is hard to understand [1]. Communication of encrypted messages such that only willful recipient can decrypt is the main objective of cryptography. Cryptography involves communicating parties, each with a secret key shared between them. Complex computations are used to attain encryption in such a manner that only the anticipated recipient can decrypt the data. Data can be anything and especially in today's world; most of the data is shared in the form of digital images. Hence, when storing the images over untrusted third parties, there is a need arised to secure them from unauthorized access and modifications. One way to secure digital images is image encryption. The secret image is encoded using complex mathematical formulations such

P. Sharma (✉)
School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India
e-mail: priyamwada14@gmail.com

V. Sharma
University Institute of Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India
e-mail: vedant1998@gmail.com

© Springer Nature Singapore Pte Ltd. 2019

R. K. Shukla et al. (eds.), *Data, Engineering and Applications*,
https://doi.org/10.1007/978-981-13-6351-1_2

that a distorted random-like structure can be obtained in the form of an encrypted image.

On the other hand, visual cryptography is emerging as a new way to secure the images where shares of the plain secret image are generated. Secret image is recovered when all the shares are stacked together. Both image encryption and visual cryptography sound quite the same, but they are different in the sense they realize security to the image. Partitioning of the plain secret image into multiple shares such that no image information is revealed from any of the shares is the primary objective of visual cryptography [2]. The shares can then be transmitted through any untrusted medium. At the receiving side, all the shares are stacked and plain secret image is reconstructed. Introduced by Naor and Shamir [3], visual cryptography was proposed as an easy and feasible approach for secretly sharing the images without needing any difficult cryptographic operations. The concept of visual cryptography can be used to securely store the secret image over untrusted cloud infrastructure. Secret image shares can be generated such that no information can be revealed and stored over cloud.

Simple illustration of visual cryptography is shown in Fig. 1 in which the plain secret image is divided into n shares. After the concept of visual cryptography came to light, several schemes have been proposed. In order to achieve more security of image shares, Shankar and Eswaran [4] proposed a scheme where individual matrices of red, green, and blue channels of the secret image are created by considering the true color band RGB of the secret image pixels. Every value of R, G, and B channels is minimized to further create the submatrices. Division operation is used for the purpose and floor and ceiling functions are used to create two matrices from a single one. In [5], the authors proposed a scheme that uses cellular automata (CA) for visual cryptography. Cellular automaton is a model constituting parallel entities that cooperate with each other and manipulate their progress in a discrete way. If cellular automaton is to be represented in a way that each entity has the state of either 0 or 1, such cellular automata are known as binary cellular automata. A set of rules is used to define state of the cells of cellular automata and acts as secret key for encryption as well as decryption of the plain secret image. Visual cryptography is considered amongst one of the modern cryptographic techniques to share plain secret image. However, the received image size is doubled or more than the original secret image which usually limits the technique.

Additional works other than visual cryptography include share generation using certain polynomial function or any other medium. A secret sharing algorithm to securely transmit the plain image was proposed in [6]. The scheme reduces size of the image shares while maintaining their security. Shamir's secret sharing polynomial was used with pixel values as random coefficients. The scheme was good to share a secret image while reducing amount of traffic on the network. However, it was claimed in [7] that security of the scheme [6] has a limitation of dependency on the permutation stage. In [7], the authors presented a scheme to overcome the limitation of permutation stage. But the scheme [7] still has a limitation of revealing secret image in initial shares. Therefore, further improvements are presented in [8] by repeatedly modifying the share numbers using modulo prime function.

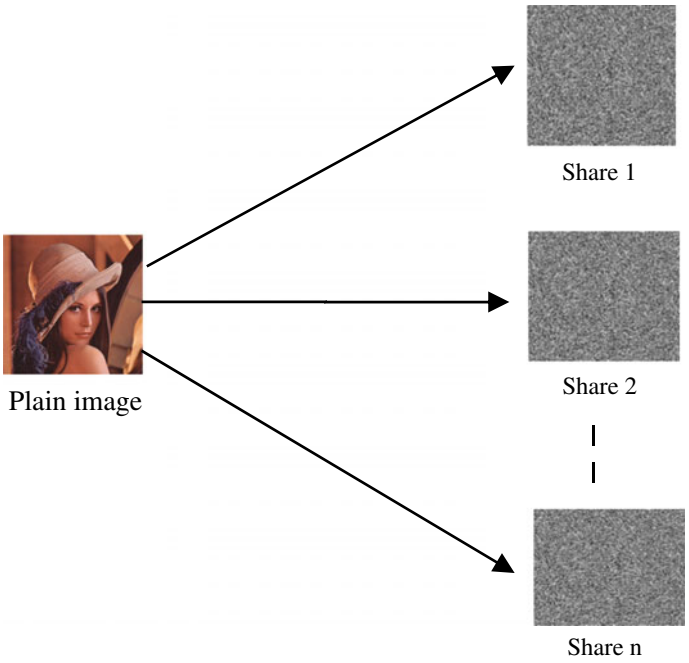


Fig. 1 Visual cryptography

In this paper, a simple and efficient secret image sharing scheme is presented such that a plain secret image can be stored over cloud infrastructures. The proposed scheme uses one-dimensional chaotic logistic map to generate image shares which can be further stored on the cloud. Rest of the paper is organized as follows: chaotic logistic map and bifurcation of one-dimensional chaotic logistic map are described in Sect. 2. Proposed scheme is discussed in detail in Sect. 3. Section 4 presents results and security analysis. Finally, Sect. 5 concludes the paper.

2 Chaotic Logistic Map

Chaotic maps are used as a new way of cryptography. Due to essential properties such as unpredictability, ergodicity, and sensitivity to primary conditions, chaotic systems are emerging very rapidly in building the cryptosystems [9]. A one-dimensional chaotic logistic map is defined as follows:

$$x_n = \mu * x_{n-1} * (1 - x_{n-1}) \tag{1}$$

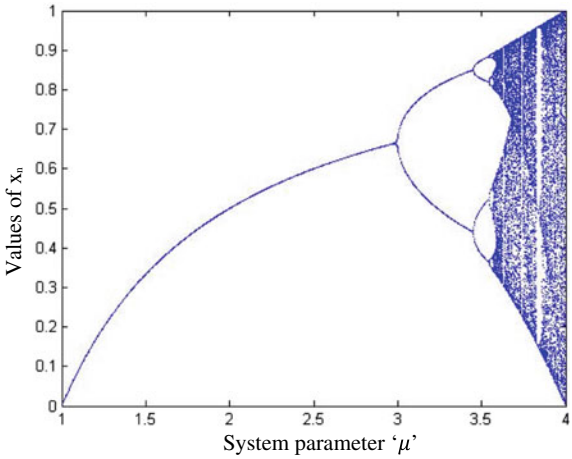


Fig. 2 Bifurcation of one-dimensional chaotic logistic map

where μ is the system parameter. Bifurcation of one-dimensional chaotic logistic map depicted in Eq. (1) is shown in Fig. 2, and it can be observed from Fig. 2 that chaotic map possess high random behavior around $\mu = 3.999$. Due to such a high random behavior, we have used 3.999 as value of the system parameter. Chaotic systems require initial conditions to generate random sequences and a slight modification in initial values or system parameter leads a major change in the generated chaotic sequences. Due to such a high random behavior of chaotic maps, a number of chaos-based image encryption schemes have been proposed during the last decade [9–16].

3 Proposed Scheme

In the proposed scheme, secret image is divided into two shares using chaotic keystream derived from one-dimensional chaotic logistic map. Two main phases of the proposed scheme are described as follows:

3.1 Computation of Initial Value of Chaotic Map Using Secret Key

At the sender's end, 132-bit secret key is used to compute the initial value of the chaotic map. The steps are as follows:

(1) Let hexadecimal representation of the secret key is
$k = k_0, k_1, k_2 \ldots k_{32}$
where k represents the hexadecimal value.
(2) Now, convert the key into binary form as

$b = b_0, b_1, b_2 \ldots b_{131}$
where b represents the binary value.

(3) Calculate the value x_{01} as
$x_{01} = b_0 \times 2^{131} + B_1 \times 2^{130} + B_2 \times 2^{129} \ldots + B_{131} \times 2^0$

(4) Now evaluate the initial value x_1 as
$x_1 = (x_{01}/2^{132}) \bmod 1$
This initial value is used to execute one-dimensional chaotic logistic map depicted in Eq. (1).

3.2 Image Encryption and Transmission of Shares to the Cloud

Here, we assume that the two cloud infrastructures are unaware of each other such that there should be no possibility of collision attack. Once initial values of the chaotic map are computed, secret image can be encrypted to form the shares such that the shares can be transmitted for storage on cloud. First of all, execute the chaotic map to get key matrix as follows:

initialize n=1;
while (true)
$x_{n+1} = \lceil 3.999 \times x_n \times (1 - x_n) \times g \rceil$
$n = n + 1$
end

where g is the highest gray-level value of the secret image and n represents the chaotic sequence number. A chaotic key matrix x of the same size as of the secret image is obtained when the complete loop is executed. After executing the chaotic map for *rows* * *columns* times, perform the following:

initialize m=1;
while (true)
$e_m = e_m \oplus x_m$
$m = m + 1$
end

The resulting image e and chaotic key matrix x can be transmitted to the cloud now [18]. In other words, it can be said that e and x are the shares of secret image and without knowledge of both the shares, secret image cannot be recovered. No information about the secret image is revealed and can be clearly observed from Fig. 3. Complete workflow of the proposed scheme is shown in Fig. 4.

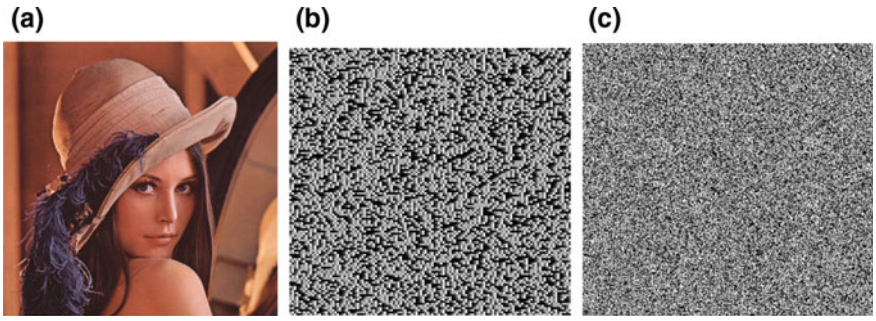
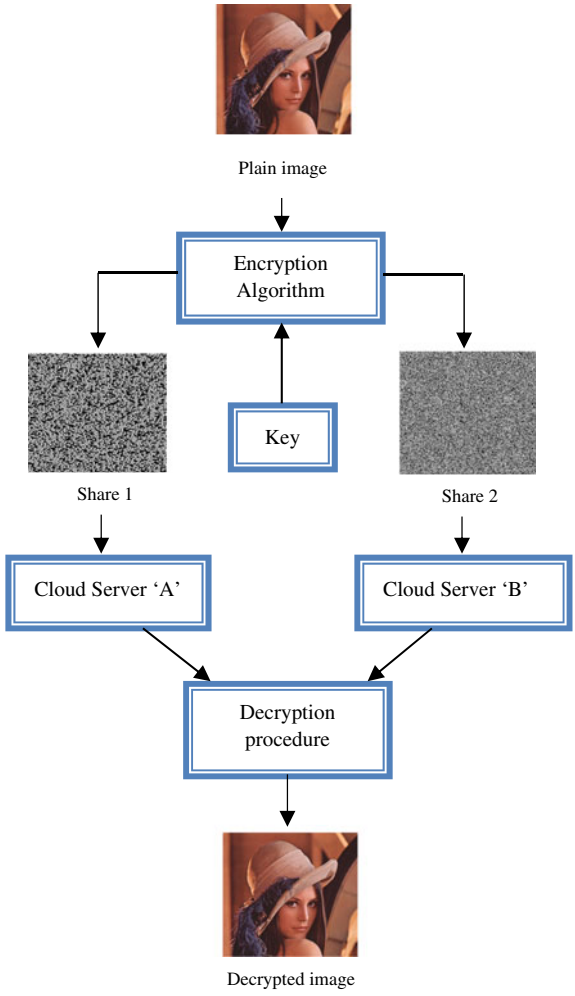


Fig. 3 **a** shows the plain image, **b** and **c** show the non-revealing image shares generated by the proposed scheme

Fig. 4 Block diagram of the proposed scheme



4 Experimental Results

In this section, we can simulate proposed scheme in Matlab 2012b. This section can be categorized into number of subsections such as key space analysis, key sensitivity analysis, Histogram analysis, and correlation coefficient analysis. After simulation, we can compare the result of proposed scheme with existing scheme in Table 1 and Fig. 7.

4.1 Key Space Analysis

Sensitivity to the secret key is very important for a good image encryption algorithm, and the key space should be large enough to make brute force attacks infeasible. Key analysis for the proposed image cipher has been performed and carried out with results summarized as follows: The proposed image cipher has 2^{132}-bit secret key. An image cipher with such enormous key space is adequate for resisting all kinds of brute force attacks.

4.2 Key Sensitivity Analysis

For an ideal image cipher, a slight change in secret key should produce a completely different encrypted image. In order to perform key sensitivity test, we encrypted secret image with secret key k_1 as "67D5EA180B4CF3FA3BF4AD1E27CF2 D7B" and then again we encrypted the same secret image with a slight change in secret key as k_2 "67D5E**D**180B4CF3FA3BF4AD1E27CF2D7B". The results are shown in Fig. 5.

Table 1 Correlation coefficients of two adjacent pixels of the plain image "Lena" and corresponding encrypted shares

	Encrypted shares by the proposed scheme		Encrypted image by [17]
	Share 1	*Share 2*	
Horizontal	0.0137	0.0098	−0.01516
Vertical	−0.0189	−0.0114	0.01396
Diagonal	0.0109	0.0134	0.02180

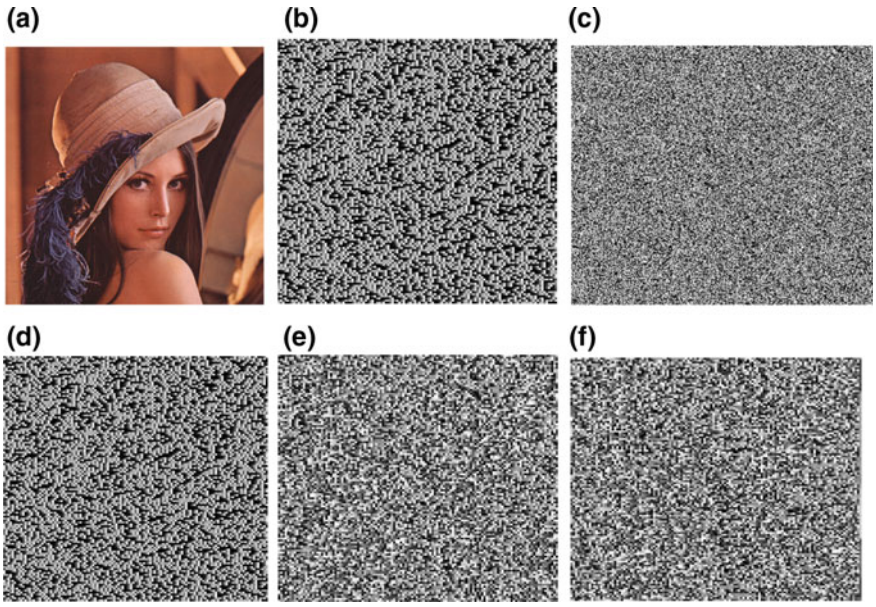


Fig. 5 Key sensitivity analysis: **a** plain image; **b–c** shares generated by key k_1; **d–e** shares generated by key k_2; **f** decrypted image using modified key

4.3 Histogram Analysis

Histogram is the graphical representation of an image. It exemplifies how pixels in an image are distributed by plotting the number of pixels at each color intensity level. For good image cipher, histogram should be uniformly distributed such that no information about the secret image can be revealed spatially. Histogram analysis of the plain and encrypted shares by the proposed scheme is shown in Fig. 6, and it can be clearly observed that histogram of the image shares is fairly uniform and significantly different from the respective histogram of the plain image.

4.4 Correlation Coefficient Analysis

Correlation between adjacent pixels is usually high in the plain image Fig. 7. On the contrary, correlation between adjacent pixels should be as low as possible for the encrypted image. Horizontal, vertical, and diagonal adjacent pixels are considered for computing the correlation coefficients. The correlation coefficient of two adjacent pixels is calculated as:

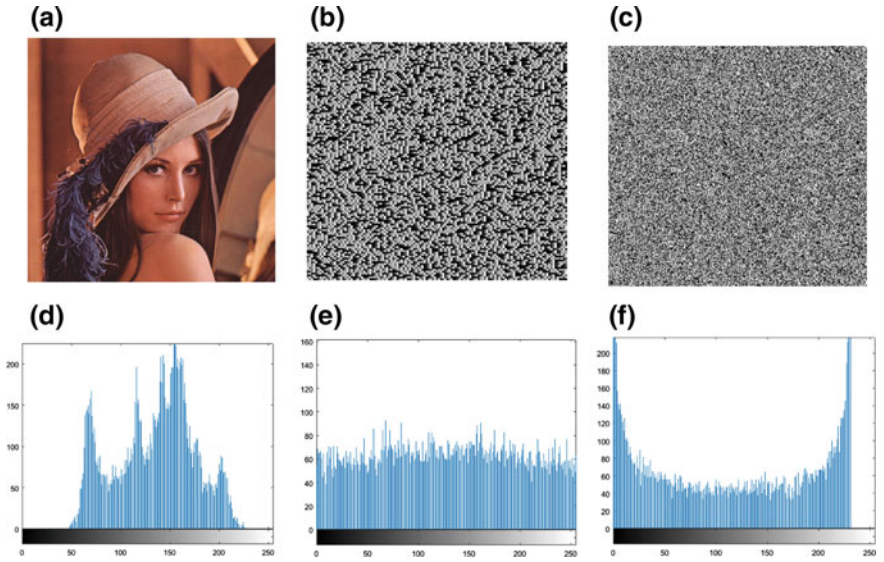


Fig. 6 Histogram analysis: **a** shows the plain image "Lena". **b–c** show the corresponding image shares. **d–f** show histograms of images shown in **a–c,** respectively

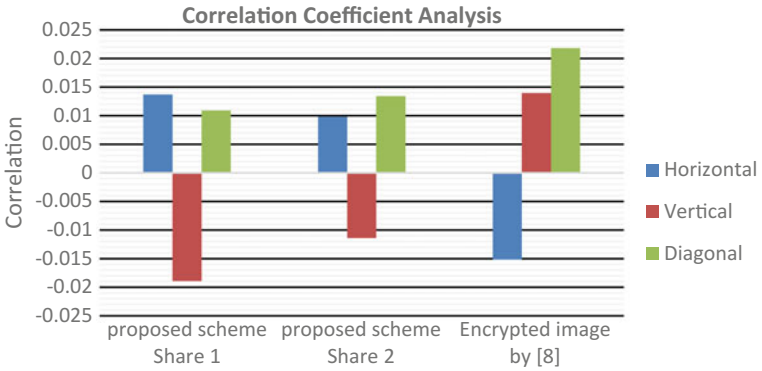


Fig. 7 Correlation coefficient analysis

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))^2 \tag{3}$$

$$cov(x, y) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))(y_i - E(y)) \tag{4}$$

$$r_{xy} = \frac{cov(x, y)}{\sqrt{D(x)} \times \sqrt{D(y)}} \tag{5}$$

where x and y are the values of two neighboring pixels of the image and, $cov(x, y)$, $D(x)$ and $E(x)$ designates the covariance, variance, and mean.

5 Conclusion

Rapid growth of multimedia applications in today's world has raised security issues for communication and storage of digital images. The situation becomes more complex when the images are to be stored on untrusted third parties like cloud infrastructures. Hence, security of such image data is very important and should be considered primarily. A technique to securely store and transmit digital images over cloud is presented in this paper. One-dimensional chaotic logistic map is used to divide the secret image into two obfuscated shares. Experimental results showed that image shares do not reveal any information about the secret image. Various tests including keyspace, key sensitivity, histogram analysis, and correlation coefficient analysis have been performed which demonstrates that the proposed scheme is good to resist various attacks.

References

1. Schneier, B.: Applied Cryptography: Protocols Algorithms and Source Code in C. Wiley, New York, USA (1996)
2. Liao, X., Lai, S., Zhou, Q.: A novel image encryption algorithm based on self-adaptive wave transmission. Sig. Process. **90**(9), 2714–2722 (2010)
3. Naor, M., Shamir, A.: Visual cryptography. In: Advances in Cryptology—EUROCRYPT'94, pp. 1–12. Springer, Berlin/Heidelberg (1995)
4. Shankar, K., Eswaran, P.: Sharing a secret image with encapsulated shares in visual cryptography. Procedia Comput. Sci. **70**, 462–468 (2015)
5. Yampolskiy, R.V., Rebolledo-Mendez, J.D., Hindi, M.M.: Password protected visual cryptography via cellular automaton rule 30. In: Transactions on Data Hiding and Multimedia Security IX, pp. 57–67. Springer, Berlin/Heidelberg (2014)
6. Thien, C.-C., Lin, J.-C.: Secret image sharing. Comput. & Graph. **26**(5), 765–770 (2002)
7. Alharthi, S., Atrey, P.K.: An improved scheme for secret image sharing. In: IEEE International Conference on Multimedia and Expo (ICME), pp. 1661–1666 (2010)
8. Alharthi, S.S., Atrey, P.K.: Further improvements on secret image sharing scheme. In: Proceedings of the 2nd ACM Workshop on Multimedia in Forensics, Security and Intelligence, pp. 53–58 (2010)
9. Yoon, J.W., Kim, H.: An image encryption scheme with a pseudorandom permutation based on chaotic maps. Commun. Nonlinear Sci. Numer. Simul. https://doi.org/10.1016/j.cnsns.2010.01.041 (2010)
10. Tong, X., Cui, M.: Image encryption with compound chaotic sequence cipher shifting dynamically. Image Vis. Comput. **26**, 843–850 (2008)

11. Behnia, S., Akhshani, A., Mahmodi, H., Akhavan, A.: A novel algorithm for image encryption based on mixture of chaotic maps. Chaos, Solitons Fractals **35**, 408–419 (2008)
12. Pareek, N.K., Patidar, V., Sud, K.K.: Image encryption using chaotic logistic map. Image Vis. Comput. **24**, 926–934 (2006)
13. Patidar, V., Pareek, N.K., Sud, K.K.: Modified substitution–diffusion image cipher using chaotic standard and logistic maps. Commun. Nonlinear Sci. Numer. Simul. **15**, 2755–2765 (2010)
14. Jolfaei, A., Mirghadri, A.: Image encryption using chaos and block cipher. Comput. Inf. Sci. **4**(1), 172–185 (2011)
15. Guarnong, C., Mao, Y., Chui, C.K.: A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos, Solitons Fractals **21**(3), 749–761 (2004)
16. Chen, G.R., Mayo, Y.B., et. al.: A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos Solitins & Fractals **21**, 749–761 (2004)
17. Abdo, A.A., Lian, S., Ismail, I.A., Amin, M., Diab, H.: A cryptosystem based on elementary cellular automata. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 136–147 (2013)
18. Xiang, T., Hu, J., Sun, J.: Outsourcing chaotic selective image encryption to the cloud with steganography. Digit. Signal Process. **43**, 28–37(2015)