Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010306
ISSN 2790-0916 Vol. 1, Issue 3: 46-54

Distributed Transcoding System Integrating HDFS and MapReduce

Amarn Charun*

Prince Sattam Bin Abdul Aziz University, Saudi Arabia

**corresponding author*

Keywords: HDFS, MapReduce, Distributed Systems, Transcoding System

Abstract: With the development of TV data services, the video conversion equipment in the past was concentrated, and the storage capacity and processing functions did not have scalability, and could not adapt to large-scale video processing. The video editing task is an information-intensive task, and a large number of video transcoding tasks are transferred to a distributed system, making large-scale video transcoding work possible. The purpose of this article is to integrate the research of HDFS and MapReduce distributed transcoding system, fully consider the advantages of HDFS and MapReduce, combine video encoding and decoding technology and video submission method, and propose a group-based classification GOP. Video clips are transcoded individually, and the entire transcoding system is easily configured and managed through a workflow. The results show the impact of changing the number of cluster nodes on transcoding performance, with 10 and 50 GB datasets having faster transcoding speeds in the system compared to 1, 2, 4 and 6 GB datasets. Effects of changing block size and manual copy number on transcoding performance. The experimental results show that the system achieves the best transcoding time performance when the block copy number is set to 4.

1. Introduction

As mobile applications will become more and more important, video, as the most important tool on the Internet, will also play an important role in mobile applications. However, the current mobile video application does not play a role, mainly because the formats supported by mobile phones are limited, and the screen size, display capability and processing capability of each mobile terminal are different, and the demand for video is also different [1]. Since video conversion requires processing the entire split video file as one record, all video must be streamed during transcoding [2].

In the face of large-scale data level, MapReduce has become the most widely used parallel

programming model. Hosseini B proposed an understandable parallel and distributed fuzzy weighted clustering method at each stage. Although the proposed method can be used for many data clustering purposes, it has been applied to multiple expression clustering to reveal functional relationships of genes in biological processes. Following MapReduce, the proposed method also proposes a new similarity measure that benefits from a combination of ordered weighting and Spearman's correlation coefficient. In the proposed method, genes of access density are added to build clusters. The final cluster result is then obtained by merging these clusters. The voting system finds the best weights among all possible outcomes for different variables and thus the most efficient clusters. The entire algorithm is implemented on a distributed computing platform and is scalable to process data of any size stored in cloud infrastructure [3]. Martin D proposed MRQAR, a new generalized clustering technique for finding quantitative association rules in large amounts of data, whose design follows a MapReduce process using Apache Spark. MRQAR is incrementally trained to run any sequential quantitative regression algorithm on big data problems without reconfiguring such algorithms. As a case study, we integrate the multi-objective evolutionary algorithm MOPNAR into MRQAR to validate the general MapReduce process proposed in this work. Results obtained in an experimental study on five big data problems show that MRQAR is able to obtain a compact set of quality rules in time [4]. Neshatpour K demonstrates a complete end-to-end implementation of a big data analytics application on a hybrid CPU + FPGA architecture. Choosing the best architecture that provides the highest acceleration for big data applications requires a deep understanding of each application. Therefore, we developed MapReduce implementations of K-means, K-Nearest Neighbors, SVM, and Naive Bayes in the Hadoop streaming environment and found it suitable to develop FPGA-compatible non-Java mapper functions to increase growth. Ambient - depends on the hardware. Further analysis of the various components of Hadoop MapReduce to identify candidates for hardware acceleration [5]. It can be seen that this paper integrates HDFS and MapReduce to study the distributed transcoding system with certain innovation.

In this paper, the large video transcoding system combining HDFS and MapReduce technology effectively shortens the processing time of large video files, and the computing power and storage capacity of the system can also be freely expanded with the improvement of transcoding requests, especially in - The most mainstream high-definition video. If Hadoop technology is used as a distributed video transcoding platform, it will be able to overcome the problems of low data storage security and low transcoding performance in the traditional centralized video transcoding system, and realize the storage and transcoding performance of large video data. For large-scale video sources stored and backed up on the storage server in the network monitoring system, large-scale transcoding optimization is carried out in the background to adapt to various terminals with different needs, while achieving a smooth viewing experience and cleaning requirements.

2. Research on Distributed Transcoding System Combined with HDFS and MapReduce

2.1. HDFS File System

The HDFS cluster adopts a master-slave system. A cluster has two types of nodes: a single Namenode and multiple Datanodes. A set of basic services for collecting and providing external catalogs of information about files or data. The database node is an underlying entity responsible for managing the actual database storage and retrieval, processing data read and write requests, and periodically reporting data block information to the name node [6-7]. When executing a task, the name node is responsible for providing information such as the location and capacity of the data

node where the data source of the task is located. Data is stored in files in HDFS with a default size of 64 MB [8-9].

2.2. Mapreduce Computing Model

Map/Reduce is an open source cloud computing programming model developed according to the Google computing model. Programs written through this model can perform large-scale data processing in a cluster system with high reliability, high fault tolerance, and high efficiency in parallel, such as data mining, log analysis, etc. [10-11]. The Map Reduce framework also provides a task scheduler configured in the form of a plug-in responsible for unified scheduling of jobs. At the same time, the framework can also monitor tasks and re-execute failed tasks. Map Reduce is similar to HDFS, or even consistent, both are Master/Slave Architecture, one is the Job Tracker as the Name Node in HDFS; the other is the Task Tracker as the Data Node in HDFS. A Hadoop cluster usually consists of a Job Tracker and multiple Task Trackers, Job Tracker is used for task scheduling, and Task Tracker is used for task execution [12-13].

2.3. Principle of Video Compression Coding

In addition to the above overloads, there are also information entropy overload, cognitive overload, visual overload, image area overload, modular texture calculation, etc. [14-15]. Using these correlations, a part of the pixel data can be removed from the data of another part of the pixel, so that the data can be highly compressed, which is beneficial to the storage and transmission of the data [16-17].

Common video compression algorithms mainly use discrete cosine transform (DCT) to reduce spatial redundancy, and use predictive coding and motion compensation techniques to reduce temporal redundancy [18].

3. Design and Research of Distributed Transcoding System Combined with HDFS and MapReduce

3.1. The Overall Framework of the System

As known in Figure 1, the system can be divided into three parts, the client layer, the user interface module, the specific transcoding module. The client terminal is connected to the system through WIFI, GPRS, network cable, etc. After the user connects to the system and logs in through registration, he can manage his own user information, manage historical video information, and perform video transcoding operations as needed. The transcoding module transcodes the corresponding video files according to the user's needs according to the transcoding task submitted by the user.

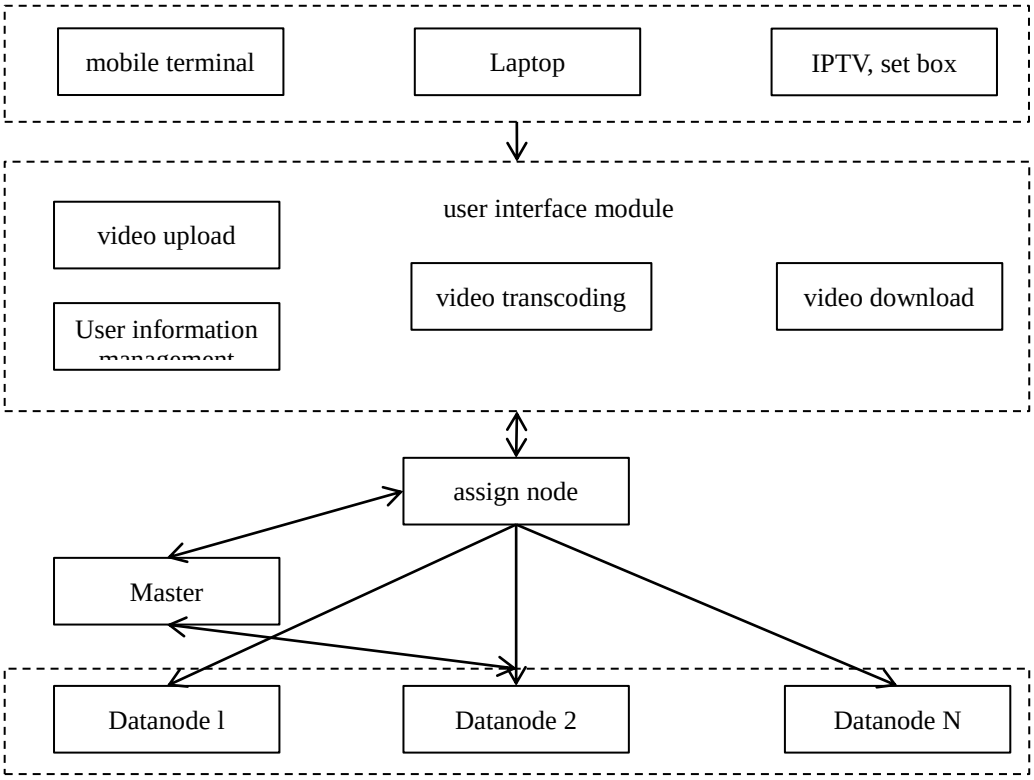

Figure 1. Overall frame diagram of the transcoding system

3.2. Video Codec Conversion Implementation

The main function of the video codec conversion module is to perform codec conversion on the video that is fragmented on HDFS. Since Hadoop is developed based on Java, the open source project Xuggle is selected as the video codec tool for the convenience of invocation. Xuggle encapsulates FFmpeg in Java, and its core codec library is the same as FFmpeg.

3.3. Determining the Weight Coefficient Using AHP

Write the hierarchical model, construct the associated comparison matrix, calculate the largest eigenvalue and the corresponding eigenvector, and normalize each column of the comparison matrix. The result of processing is:

$$\bar{a}_{ij} = a_{ij} / \sum_{i=1}^{n} a_{ij} \tag{1}$$

A common comparison table is summed by one row. After the row is summed, the elements on the right side of the comparison table form a vector, then each element is normal:

$$w_i = \bar{w}_i / \sum_{i=1}^{n} w_i \tag{2}$$

Calculate the largest eigenvalue:

$$\lambda_{\max} = \frac{1}{n} \sum_{i=1}^{n} \frac{(AW)_i}{w_i} \tag{3}$$

where (AW)i represents the ith element of the AW vector.

4. Analysis and Research of Distributed Transcoding System Combined with HDFS and MapReduce

4.1. The Effect of Changing the Number of Cluster Nodes on Transcoding Performance

The purpose of the test is to measure the overall transcoding time and system speed at different cluster sizes. It is defined as: transcoding time of 1 node/transcoding time of N nodes (N is the specific number of nodes). The Hadoop platform is configured with default settings. Table 1 shows the transcoding time speed for different number of node clusters.

Table 1. Speedup of transcoding time

Number of nodes	Data set					
	1GB	2GB	4GB	6GB	10GB	50GB
1	1	1	1	1	1	1
2	1.68	1.71	1.86	1.86	1.87	1.95
4	3.27	3.34	3.68	3.82	1.89	1.94
6	4.32	4.86	5.24	5.56	5.61	5.87

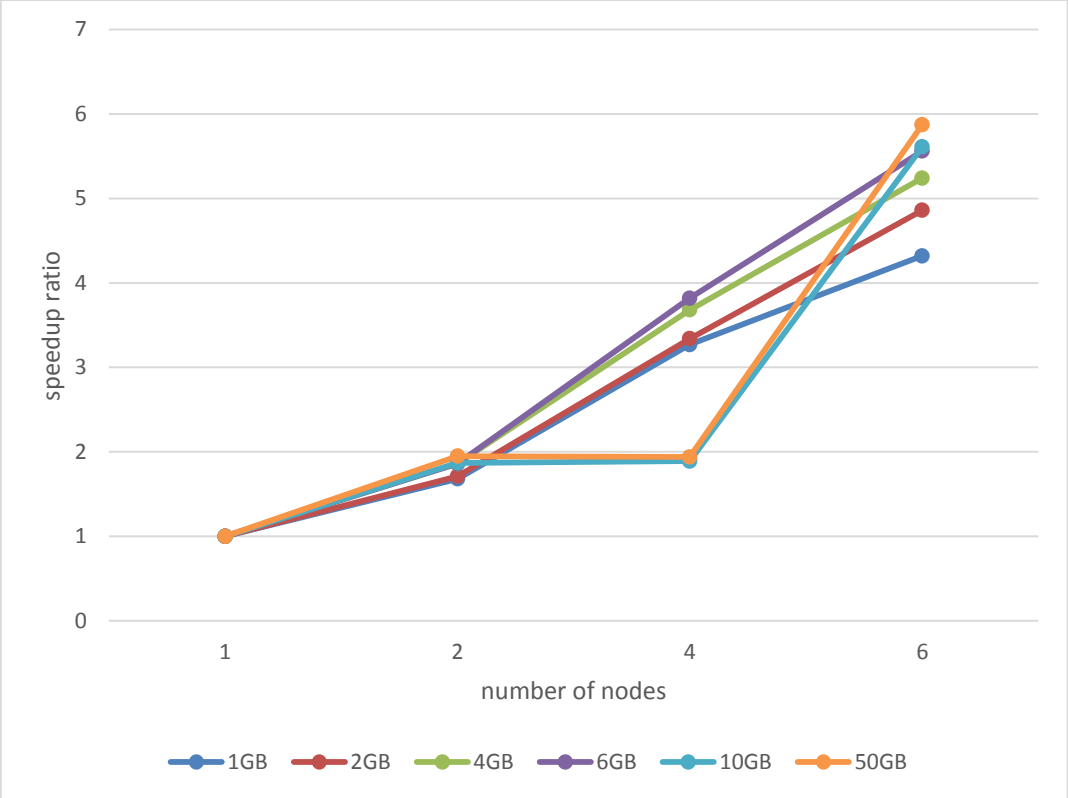

Figure 2. Speedup of transcoding time with different number of nodes

From the results of the speedup ratio in Figure 2, the speedup ratio of the transcoding time of 1, 2, 4, 6, 10, and 50GB is all 1, and when the number of nodes is 2, the speedup ratio of 1GB is 1. The

speedup for transcoding time is 1.68, the speedup for 2GB is 1.71, the speedup for 4GB is 1.86, the speedup for 6GB is 1.86, the speedup for 10GB is 1.87, and the speedup for 50GB is 1.86. The speedup for transcoding time is 1.95. When the number of nodes is 4, the speedup ratio of 1GB transcoding time is 3.27, the speedup ratio of 2GB transcoding time is 3.34, the speedup ratio of 4GB transcoding time is 3.68, the speedup ratio of 6GB transcoding time is 3.82, and the speedup ratio of 10GB transcoding time is 3.82. The speedup is 1.89, and the speedup for 50GB transcoding time is 1.94. When the number of nodes is 6, the speedup ratio of 1GB transcoding time is 4.32, the speedup ratio of 2GB transcoding time is 4.86, the speedup ratio of 4GB transcoding time is 5.24, the speedup ratio of 6GB transcoding time is 5.56, and the speedup ratio of 10GB transcoding time is 5.56. The speedup ratio is 5.61, and the speedup ratio of 50GB transcoding time is 5.87. The distributed transcoding system has excellent performance in its parallelism. Compared with 1, 2, 4, and 6 GB datasets, 10 and 50 GB datasets have greater speedup when transcoding in the system. This means that the system shows good performance when dealing with large-sized datasets.

4.2. The Impact of Changing the Cluster Block Size and the Number of Block Replicas on Transcoding Performance

This paper tests the total time for the system to transcode different datasets under the conditions of different block sizes (default: 64MB) and different number of block replicas (default: 3) in HDFS settings. In the experiment, the set size options of Block are: 32, 64, 128, 256 and 512MB; the set number of copies options are: 1, 2, 3, 4 and 5. Table 2 shows the total transcoding time of the system for different datasets under different number of block replicas.

Table 2. System transcoding time under different number of block copies

Number of block replicas	Data set					
	1GB	2GB	4GB	6GB	10GB	50GB
1	305	592	1356	2768	3214	15830
4	319	534	986	1882	2323	11164
5	326	525	993	1852	2315	11195

From the experimental results in Figure 3, it can be clearly observed that the system obtains the best transcoding time performance, or when the number of replicas of the block is set to 4. because when the number of copies is set to 1, too many replicas will waste the storage space of the system. Setting the number of block replicas to 4 can prevent data loss when nodes fail during distributed transcoding of large-scale videos. situation, making the system more stable and reliable.

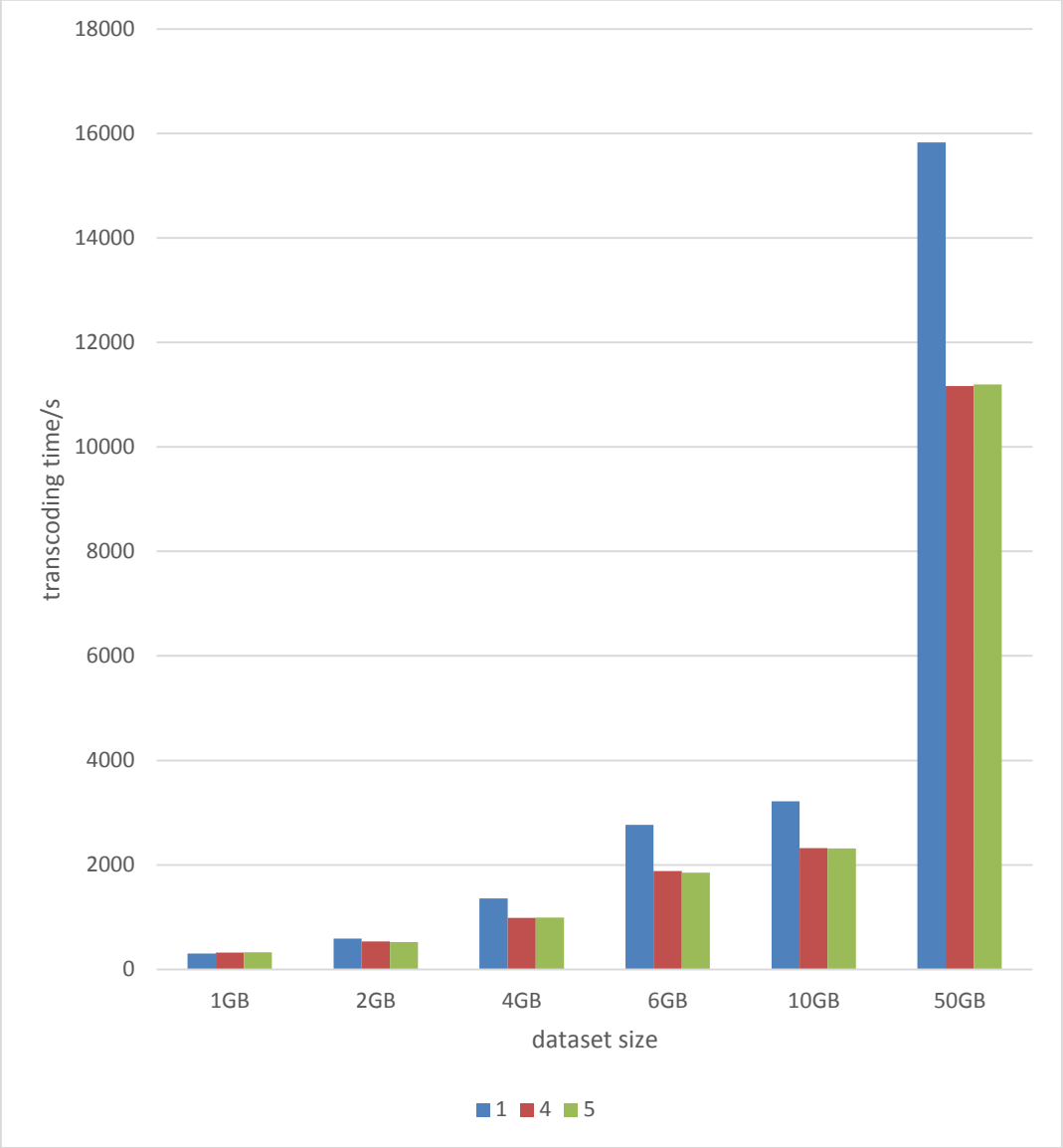

Figure 3. System transcoding time (s) under different number of block copies

5. Conclusion

This paper uses the size and stability of the shared file system to build a large-scale video database, which solves the shortcomings of traditional central storage devices such as high requirements and difficult expansion. According to development and implementation, the amount of video data is huge. Video data size. storage requirements. Through the real-time scheduling of the distributed system, each working node can effectively cooperate according to the dynamic working environment, thereby reducing the overall video transcoding time. The stand-alone transcoding system has higher requirements on computer hardware configuration, while the distributed transcoding system needs to be assembled with ordinary computers, and the conversion speed of the entire video will be faster. Use a divide-and-conquer scheme, dividing the file into appropriately sized chunks and transcoding the entire file by transforming each chunk in parallel.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Veiga J, Exposito R R, Taboada G L, et al. Enhancing in-memory efficiency for MapReduce-based data processing. Journal of Parallel and Distributed Computing, 2018, 120(OCT.):323-338.

[2] Kim J, Kim M H. An efficient parallel processing method for skyline queries in MapReduce. Journal of Supercomputing, 2018, 74(2):1-50.

[3] Hosseini B, Kiani K. FWCMR: A scalable and robust fuzzy weighted clustering based on MapReduce with application to microarray gene expression. Expert Systems with Applications, 2018, 91(jan.):198-210.

[4] Martin D, Martinez-Ballesteros M, Garcia-Gil D, et al. MRQAR: a generic MapReduce framework to discover Quantitative Association Rules in Big Data problems. Knowledge-Based Systems, 2018, 153(AUG.1):176-192.

[5] Neshatpour K, Malik M, Sasan A, et al. Energy-efficient acceleration of MapReduce applications using FPGAs. Journal of Parallel and Distributed Computing, 2018, 119(SEP.):1-17.

[6] Burawis M. Query Optimization: Fund Data Generation Applying NonClustered Indexing and MapReduced Data Cube Numerosity Reduction Method. International Journal of Advanced Trends in Computer Science and Engineering, 2020, 9(1.1 S I):102-109.

[7] Oswalt M S, Ananth J P. MapReduce and Optimized Deep Network for Rainfall Prediction in Agriculture. The Computer Journal, 2020, 63(6):900-912.

[8] Addisie A, Bertacco V. Collaborative Accelerators for Streamlining MapReduce on Scale-up Machines With Incremental Data Aggregation. IEEE Transactions on Computers, 2020, PP(99):1-1.

[9] Moutafis P, Mavrommatis G, Vassilakopoulos M, et al. Efficient processing of all-k-nearest-neighbor queries in the MapReduce programming framework. Data & Knowledge Engineering, 2019, 121(MAY):42-70.

[10] Gimenez-Alventosa V, Molto G, Caballer M. A framework and a performance assessment for serverless MapReduce on AWS Lambda. Future Generation Computer Systems, 2019, 97(AUG.):259-274.

[11] Patil A. Securing mapreduce programming paradigm in hadoop, cloud and big data eco-system. Journal of Theoretical and Applied Information Technology, 2018, 96(3):756-766.

[12] Ciofi D, Albolino S, Dagliana G, et al. Prevalence and multicenter observational study on falls of hospitalized children and Italian, linguistic-cultural validation of the Humpty Dumpty Fall

Scale. Professioni infermieristiche, 2020, 73(4):296-304.

[13] Murlistyarini S, Aninda L P, Widyarti S, et al. Exosomes of Adipose-derived Stem Cells Conditioned Media Promotes Retinoblastoma and Forkhead-Box M1 Protein Expression. Open Access Macedonian Journal of Medical Sciences, 2020, 9(A):422-427.

[14] Yousif R Z, Kareem S W, Baban S M. An approach for enhancing data confidentiality in Hadoop. Indonesian Journal of Electrical Engineering and Computer Science, 2020, 20(3):1547-1555.

[15] Prabowo S, Abdurohman M. Studi Perbandingan Performa Algoritma Penjadwalan untuk Real Time Data Twitter pada Hadoop. Komputika Jurnal Sistem Komputer, 2020, 9(1):43-50.

[16] Yousif R Z, Kareem S W, Abdalwahid S M. Enhancing Approach for Information Security in Hadoop. Polytechnic Journal, 2020, 10(1):81-87.

[17] Ayyathan D M, Koganti P, Marcu-Malina V, et al. SMURF2 prevents detrimental changes to chromatin, protecting human dermal fibroblasts from chromosomal instability and tumorigenesis. Oncogene, 2020, 39(2):1-15.

[18] Mitra A, Kundu A, Chattopadhyay M, et al. On the Exploration of Equal Length Cellular Automata Rules Targeting a MapReduce Design in Cloud. International Journal of Cloud Applications and Computing, 2018, 8(2):1-26.