IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 15, 2019, accepted December 18, 2019, date of publication December 31, 2019,
date of current version February 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963343

Application and Performance Optimization of MapReduce Model in Image Segmentation

MAOZHEN LI[1,2], LU MENG[1], JIAYING WANG[3], YONG JIN[1], BINYU HU[1], AND YOUXING CHEN[1]

[1]School of Information and Communication Engineering, North University of China, Taiyuan 030051, China
[2]Department of Electronic and Computer Engineering, Brunel University London, Uxbridge UB8 3PH, U.K.
[3]School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia

Corresponding author: Yong Jin (xiandaijiance601@163.com)

This work was supported in part by the Shanxi Scholarship Council of China under Grant 2016–084, and in part by the Shanxi Province Natural Science Foundation under Grant 201901D111155.

ABSTRACT With the increase of glass detection speed, some defects of MapReduce distributed computing framework are exposed, and the processing speed and timeliness cannot meet the requirements of glass-defect detection in industrial technology. Based on the MapReduce distributed computing framework, this paper designs a threshold segmentation method to complete the segmentation of glass-defect images. By improving the replication placement strategy and pipeline scheduling mechanism, the computing and storage are localized, and the timeliness of data processing is accelerated. The experimental results show that the improved MapReduce computing framework has an average increase of 14.8% in processing speed. It can detect the glass ribbon running at 800m/h and also detect the number, position and type of defects on the glass ribbon.

INDEX TERMS Defects detection, data locality, image segmentation, MapReduce model, pipeline scheduling.

I. INTRODUCTION

With the development of digital technology, digital image processing is widely used in industrial production. Many applications need to analyze current production status in a timely manner, and traditional digital image detection systems are hard to meet the needs of industrial production [1]. Taking the glass production industry as an example, during the period of production processes such as raw material processing, preparation, melting, clarification and cooling, due to the destruction of the process system or the errors in the operation process, the glass ribbon from the annealing kiln often exists defects of different types and sizes. The cross-cutting machine will cut off the section which contain too many defects or does not meet the requirements of national standards. So according to the characteristics of the glass production line, cross-cutting machine should be installed at the exit of the annealing kiln [2].

Due to the faster transmission speed of the glass ribbon, a large amount of high-resolution image data will be generated in a short time. To realize the timely defect detection of the glass ribbon on the production line, it is necessary to use a timely and uninterrupted online detection system which matches the production speed. The emergence of MapReduce has alleviated the problem of big data processing to some extent. The MapReduce framework was first used to process textual data collections, it has been recently used to process large images such as remote sensing images, high resolution images etc [3]. It is a data model based on key/value pairs. This model divides complex distributed computing into two phases: Map phase and Reduce phase. The Map phase is usually calculated locally in the data store and then maps the output results to the corresponding Reduce tasks. The Reduce phase summarizes the results of the Map phase and finally arrive at calculation results [4].

As the demand for data processing increases, some flaws in the distributed computing framework MapReduce are exposed [5]. MapReduce has its performance bottleneck: between Map and Reduce, there is an invisible intermediate processing part. For example, in order to distribute different results to the corresponding processing nodes, all the results

The associate editor coordinating the review of this manuscript and approving it for publication was Yongtao Hao.

need to be summarized on each node and then sorted. Each node intercepts the data in the corresponding interval (shuffle phase) [6]. This process is the key for MapReduce to operate correctly, but it affects the speed of the system processing.

In order to improve the performance of MapReduce computing framework. Some recent work focused on overlapping different phases of MapReduce to enhance job performance. Guo *et al.* [7] proposed iShufflfle, a framework that overlaps the map and shufflfle phases by predicting the distribution of partitions. Besides, Wang *et al.* [8] introduced a merge algorithm to avoid data repetition and disk access and also proposed the use of a pipeline with which to overlap the shufflfle, merge, and reduce phases. Other researchers have also proposed optimizations on the data locality. In [9], the authors conducted a comprehensive investigation of the data locality of Hadoop jobs and analyzed its impact on job performance. The authors in [10] optimized MapReduce jobs based on data locality. Some approaches were also proposed to optimize the execution process of MapReduce from other aspects, such as a large number of temporary result data generated by multiple Map tasks of the same job on the Map node are combined as a total, replacing the original MapReduce architecture to merge the result data of a single Map task [11]. Generally, the existing systems lack of research on improving the performance of MapReduce on large scale images processing. For example, the Hadoop Image Processing Framework only allows uploading images at a very slow rate [12].

Based on the above background, this paper uses the MapReduce parallel framework as the research basis to realize the threshold segmentation of defect images. By adding streaming data processing module and data partitioning module, localization of computing and storage is realized, and accelerate the timeliness of data processing. We also take a large number of glass images which are collected online as test objects, realizing timely and accurate detection of various glass defects.

II. PRINCIPLE OF SYSTEM OPERATION

The system consists of a light source, an encoder, a high-speed linear array CCD, a high-speed data acquisition card, and a Hadoop cluster processor. The light source and the high-speed linear array CCD camera are respectively located below and above the glass ribbon and on the same vertical plane. The light emitted by the light source passes through the glass ribbon and is received by the CCD camera. The high-speed data acquisition card collects the light intensity signal continuously through the dual DMA mode, converting it into grayscale images data and transmitting to the upper computer. The tasks are submitted by a client to the resource manager. The data collected by the CCD camera is transmitted to the Namenode of the Hadoop cluster. The Namenode is responsible to run DataNodes, and maintains the images data stored on the HDFS. Then, the DataNodes running assigned map tasks which complete the segmentation algorithm. Finally, the final processing results are returned to the Namenode. Once the defect is excessive, the cross-cutting machine which controlled by the system will remove the section with too many defects.

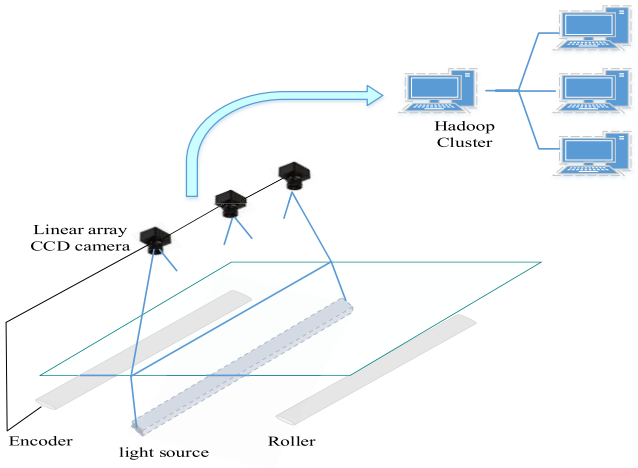


FIGURE 1. Glass defect online inspection design.

III. MapReduce-BASED CLUSTERING FOR IMAGES ANALYSE

A. DESIGNING THE IMAGE STORAGE STRUCTURE

Image segmentation is the key technology based on MapReduce defect detection. The difficulty is to ensure that the segmentation results of Map and Reduce tasks can be restored to the original images [13]. In the process of MapReduce model calculation, if the image files in the image sequences are not preprocessed, the MapReduce calculation model will directly block the images. Since each partition block does not have corresponding index number, coordinate number and other information, the aggregation of defect detection results cannot be completed after the block defect detection. The method adopted in this paper is to pre-process the glass defect image before the MapReduce task is started, and save the meta-information after the block. The meta-information includes the index information and coordinate information of the block image data which relative to the original image data. In the process of distributed defect detection, each image segmentation is processed separately by the map task. After the image segmentation is detected, the data information can be quickly and accurately aggregated in the Reduce tasks by reading the pre-stored meta information.

The size of the preprocessed images is far less than the default file block size of HDFS, which will affect the storage resource utilization of Hadoop. At the same time, accessing a large number of small images will also increase file addressing time and reduce read and write speed. Therefore, this paper uses the HipiImageBundle class of the HIPI interface to upload the local images through the file traversal method. The image file forms a storage structure which containing data and indexes. The hib file stores displacement and index information, and the hib.dat file stores images data. The designed storage structure is shown in Figure 2. Through the

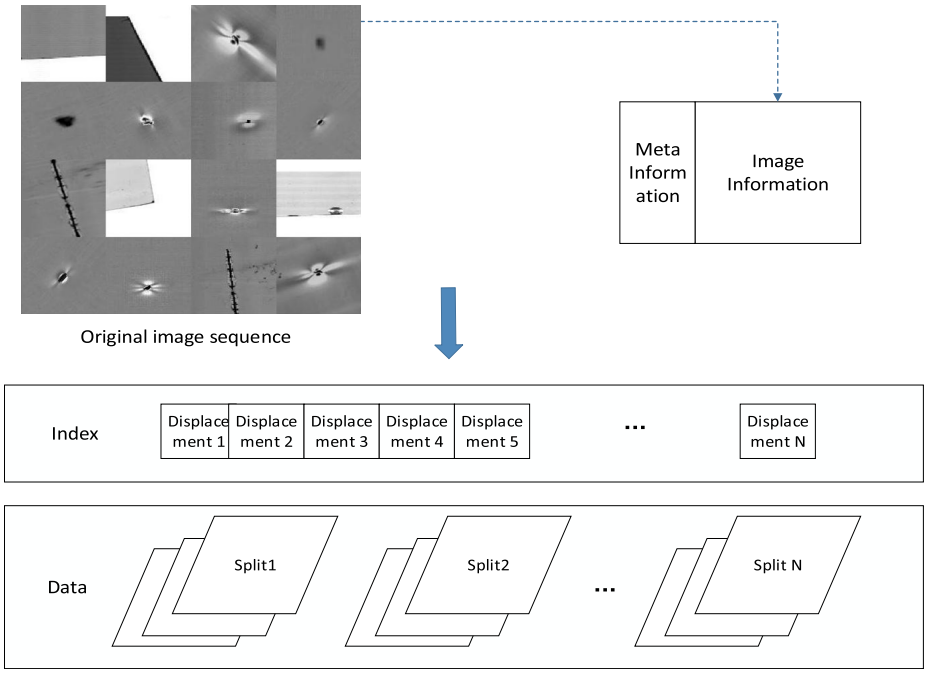


FIGURE 2. Image storage structure design.

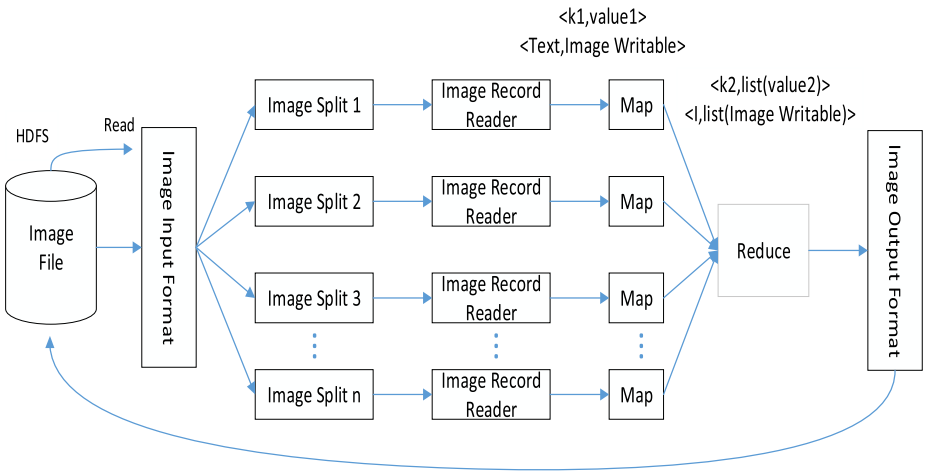


FIGURE 3. The processing of Image file.

HIPI method, the impact of large number of image fragments on Hadoop scalability and performance is alleviated.

B. DESIGNING THE MapReduce FUNCTIONS

The MapReduce implementation process of defect detection is as follows: First, each image in the images sequence is divided into multiple small image segments, and the image fragments are distributed to the Hadoop data nodes. Then, each Map process on the data node complete the defect segmentation tasks. Finally, the detected image fragments are aggregated in the Reduce process to obtain the final detection result. The workflow of MapReduce is shown in Figure 3:

In the Map stage, image fragments are read through the ImageInputFormat interface. The ImageRecordReader function is responsible for inputting and reading the records, obtaining the split records, and generating input fragments. The MapReduce program passes the input <key, value> pairs to the map tasks, and uses algorithm of threshold segmentation to complete the defect detection for each image. After the map tasks are finished, the detection results are output as a key value pair <Text, image>, and the results are sent to the reduce tasks, wherein the key saves the index number and the coordinate number obtained from the image segmentation meta information, the value saves the defect detection result of the images.

In the Reduce stage, for the metadata information is saved in the key and the detection results in the value, the detection result is merged according to the index number and the pixel coordinate which stored in the metadata, and the detection results are saved in different folders of the HDFS.

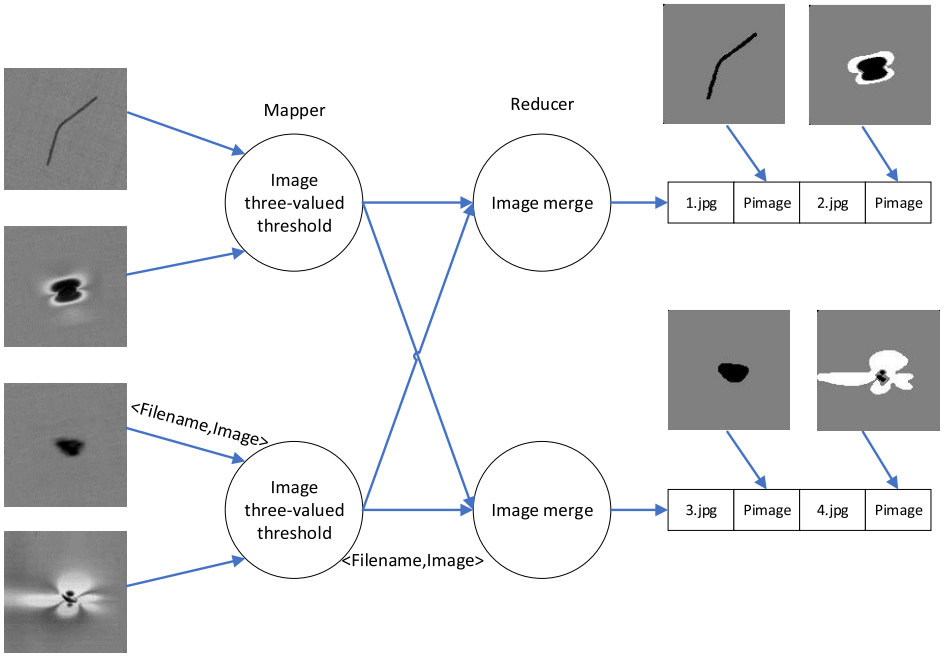


FIGURE 4. The processing of Image file.

Once the execution of the processing function for the images are completed, the reduce program start merging the images after the execution of the processing and other operations.

As shown in Fig. 4, the figure shows the specific running process of the MapReduce program, and the stored images are taken as an input, the obtained new images are processed as an output. For other image input types, only the corresponding file input format needs to be set, and different file output formats can be set to obtain different storage forms. The process of MapReduce processing internally does not change. The above process is equally applicable to other image batch tasks, except for the image processing algorithms used in Mapper. The Reducer is mainly used to merge the output of Mapper in the whole process above. It combines the output images into one large file, which avoids storing multiple small files on HDFS. If the output of new images is relatively small and needs to be stored separately, the number of task reducers can be set to 0, so that new images processed by Mapper will be directly output. Without the Reducer process, data transmission, grouping, sorting, and networks are not required, and the efficiency of the algorithm is improved.

IV. OPTIMIZATION OF THE MapReduce PROCESSING

The MapReduce job is mainly composed of the Map phase and the Reduce phase. The intermediate data generated by the data nodes in the Map phase needs to be transmitted to the compute nodes in the Reduce phase through the network. This intermediate phase is called Shuffle [14]. The resource consumption of the network bandwidth caused by the data transmission in the Shuffle phase and the data storage

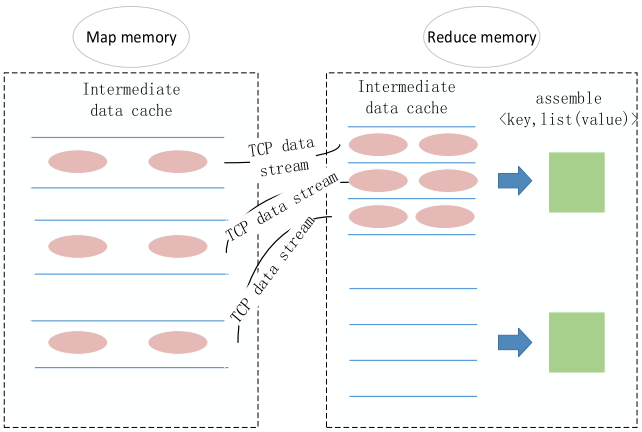


FIGURE 5. Pipeline scheduling.

in the Reduce phase is time consuming. How to reduce unnecessary network bandwidth usage in the Map phase is the key to improving the efficiency of MapReduce job execution. The network bandwidth usage in the Map phase is related to its data localization and scheduling method. Therefore, improving data localization and using pipeline scheduling can effectively improve the execution efficiency of MapReduce jobs.

A. DATA LOCALIZATION

Data localization refers to dividing the data set to each node without redundancy, each node runs independently, and processes the data by dividing the data set [15]. Hadoop based on MapReduce uses static replication-placement technology, that is, in a distributed file system, the number of each

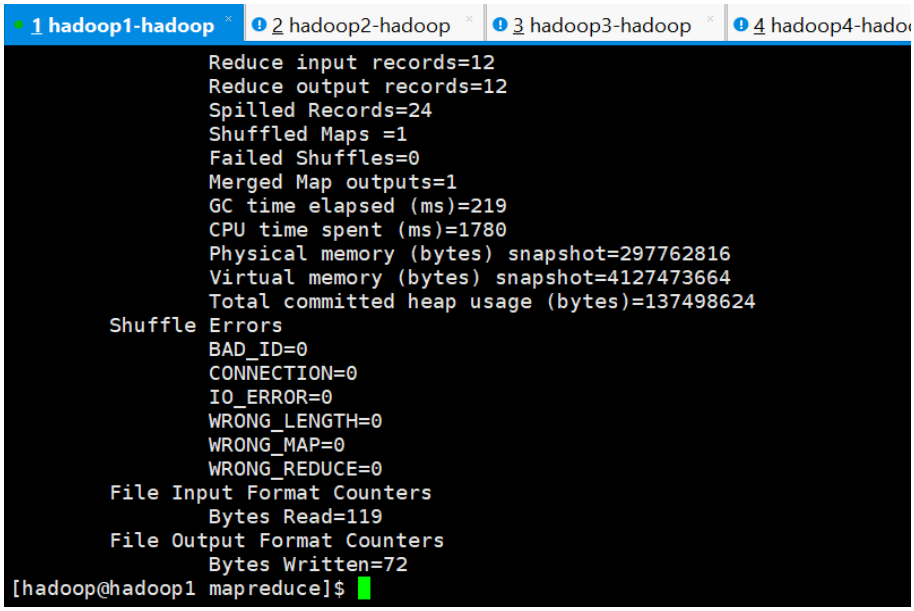
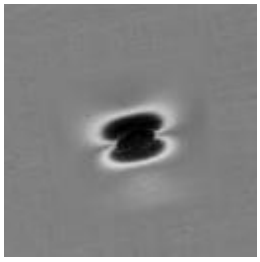
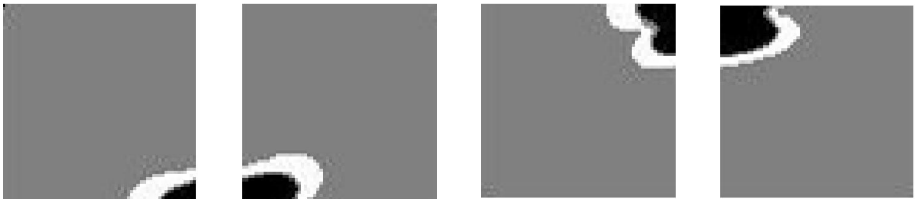


FIGURE 6. Map-Reduce processing results on Hadoop cluster.



(a)Original glass defect image



(b) Block detection intermediate result

(c) Threshold segmentation result

FIGURE 7. Defect detection result based on MapReduce.

replication is three. Although this static replication-placement method can meet the service requirements of general applications in Hadoop clusters, it does not fully utilize the storage space of the cluster to a certain extent, which reduces the data localization of tasks and the performance of task schedulers.

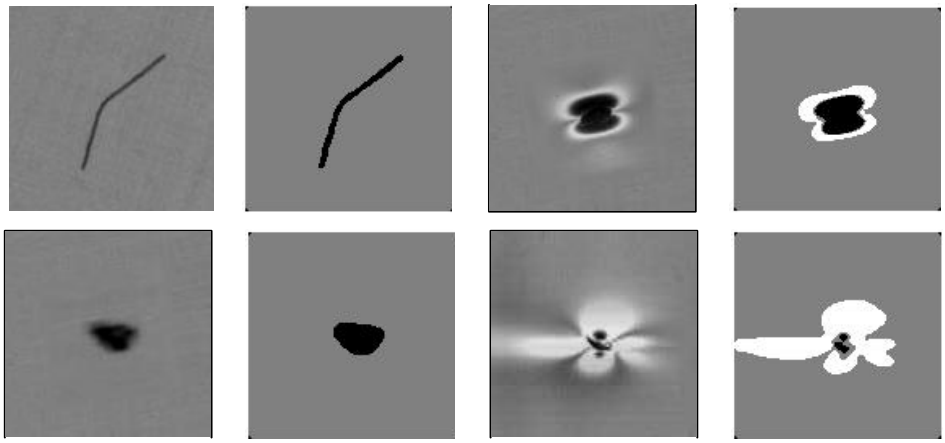


FIGURE 8. Defect images segmentation result based on MapReduce.

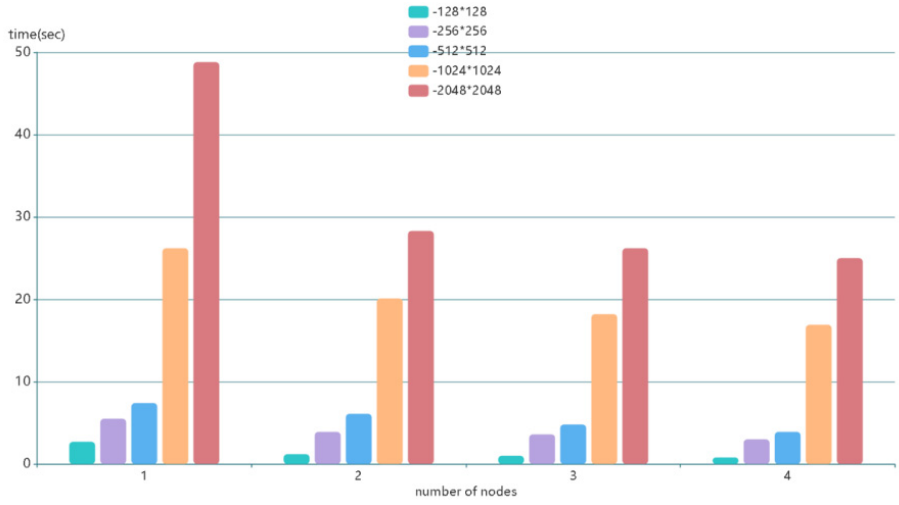


FIGURE 9. Processing time of using Hadoop with different slave nodes.

In view of the above problems, this section improves the localized copy placement algorithm. The algorithm dynamically changes the number of data block replications by calculating the access frequency of data blocks in HDFS, which increases the replications of data blocks with high access frequency and reduces the replications of low-frequency. Increasing the replication of data blocks with high access frequency can improve the effectiveness of data localization. Through this algorithm, the data localization of the task is improved, thereby improving the efficiency of the traditional MapReduce framework. Algorithm 1 shows the pseudo code of the improved framework.

Algorithm 1 is described as follows:

First, sort the compute nodes in the Hadoop cluster based on the historical data access frequency. Then, initialize the data block and iterate over the collection of the data blocks. If the block access frequency is higher than the average value of the data blocks, and the number of tasks that need to access the data blocks is higher than the current number of the data blocks, increase the number of block replication. Last, if the data block satisfies the condition of increasing the replication, the DataNode with the low frequency of the data block is sequentially taken out from the DataNode queue, and a replication is added to the node, so as to repeatedly find the appropriate one. The DataNode increases the remaining replications until the number of replication that need to be increased is completed.

B. PIPELINE SCHEDULING

At the same time, this paper also improve the pipeline mechanism for MapReduce, that is, the intermediate results which obtained in the Map phase can be quickly transferred to the Reduce phase, making the MapReduce operation pipelined [16]. The specific design is to make each Reduce task establish a TCP link with all Map tasks in the initialization phase. When the Map produces an output, it is delivered to the corresponding TCP link and sent to the specified processing node

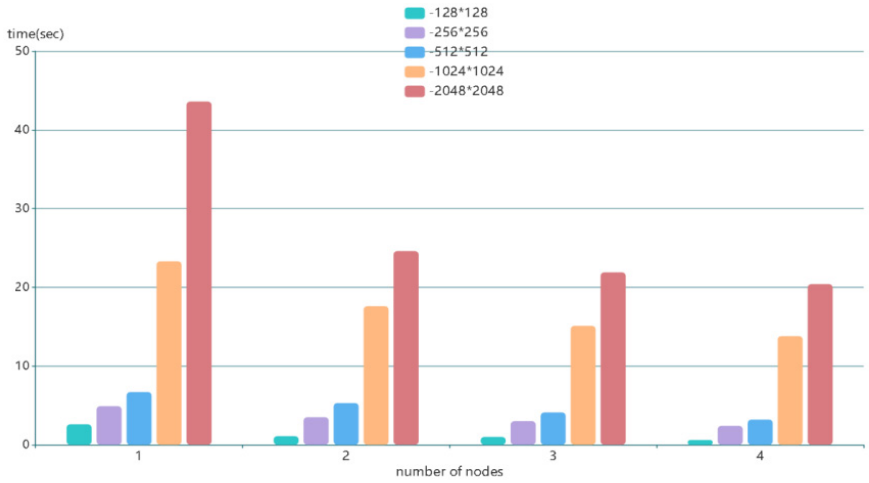


FIGURE 10. Processing time of using optimization MapReduce with different slave nodes.

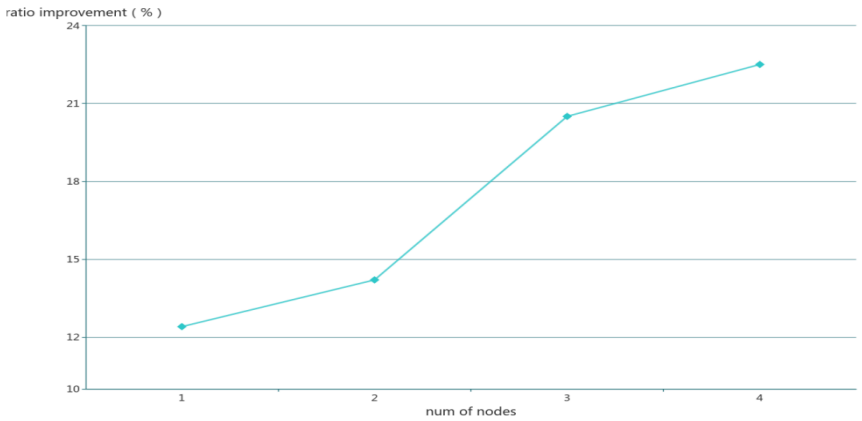


FIGURE 11. Ratio improvement using 512*512 images with different slave nodes.

through the interval partition. Figure 5 shows the improved pipeline mechanism of MapReduce. Pipeline scheduling can improve CPU usage, and selecting the appropriate multi-threading can also maximize data throughput. The improved MapReduce framework passes the calculation results directly to the next stage, which also alleviate the I/O load of the current nodes.

V. EXPERIMENTS

The experimental Hadoop cluster consists of one master node and four slave nodes. The reason for this configuration is to consider the memory space of the cluster and the needs of the actual test. The software and hardware settings for the node are shown in Table 1 and Table 2. In addition, we use glass defect images as the experimental object, and the average size of the image used in the experiment was about 235 KB.

The processing results are shown in Figure 6, we can see the memory usage of the cluster and the execution time of the program. The image processing results are shown in Figure 7. Figure 7(a) is an image of a glass defect containing a inclusion, Figure 7(b) is an intermediate result of block defect detection on MapReduce, and Figure 7(c) is a final test obtained by aggregating the intermediate results. Fig. 8 shows

TABLE 1. Software specifications.

	128*128	256*256	512*512	1024*1024	2048*2048
1	2.1	5.5	7.4	26.2	48.8
2	1.2	3.9	6.1	20.1	28.3
3	1	3.6	4.8	18.2	26.2
4	0.8	3	3.9	16.9	25

TABLE 2. Hardware specifications.

	128*128	256*256	512*512	1024*1024	2048*2048
1	1.9	4.9	6.7	23.3	43.6
2	1.1	3.5	5.3	17.6	24.6
3	1	3	4.1	15.1	21.9
4	0.6	2.4	3.2	13.8	20.4

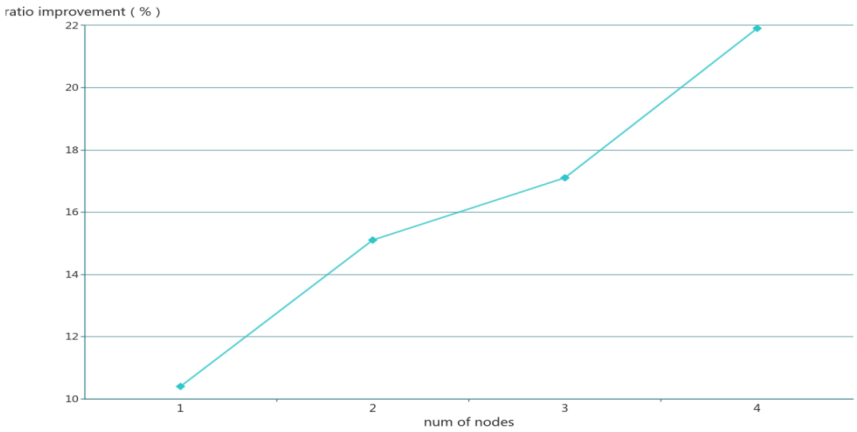


FIGURE 12. Ratio improvement using 1024*1024 images with different slave nodes.

Algorithm 1 Localized Replication Placement Algorithm

```
sort Query(DataNode);
Addreplicate=0;
for (take each elment) do
    old_replicate=replicate_factor(b_i);
    if(F(b_i )>averageFre(b_i ))
    if(Num(Query)>=replicate_factor(b_i ))
      Addreplicate=Num(Query)-replicate_factor(b_i );
    else
      Addreplicate+ =1;
    end if
    NewBlockQuery.add(b_i,Addreplicate);
    update replicate_factor(b_i );
    while(Addreplicate!=0)
      for each take Query(DataNode) do
        if(FreeSpace(D)& & NotExist(b_i))
          AddBlock();
          Addreplicate– –;
      end
    end
end
```

that the threshold segmentation results of scratches, inclusions, stains, and tumors.

In order to verify that the improved MapReduce framework has improved processing efficiency and timeliness, the improved MapReduce framework runs on the same Hadoop cluster with the same amount of computing nodes. Selecting glass defect images with resolutions of 128*128, 256*256, 512*512, 1024*1024 and 2048*2048, and process the defect images of the same resolution for each experiment to detect the efficiency of the two frame's operations. It can be seen from Fig.9 and Fig.10, with different resolutions, the processing time is significantly faster with different resolutions, and the main reason for the faster processing speeds of resolutions 1024*1024 and 2048*2048 is not only because of the amount of image data becomes larger, but also related to the size of the slice when the Mapper is started.

TABLE 3. The result of computing defect image under Hadoop.

	128*128	256*256	512*512	1024*1024	2048*2048
1	2.1	5.5	7.4	26.2	48.8
2	1.2	3.9	6.1	20.1	28.3
3	1	3.6	4.8	18.2	26.2
4	0.8	3	3.9	16.9	25

TABLE 4. The result of computing defect image under optimization MapReduce.

	128*128	256*256	512*512	1024*1024	2048*2048
1	1.9	4.9	6.7	23.3	43.6
2	1.1	3.5	5.3	17.6	24.6
3	1	3	4.1	15.1	21.9
4	0.6	2.4	3.2	13.8	20.4

Through the experimental data, we found that the linear resolution of cluster computing time and node number are more obvious when the image resolution is 512*512 and 1024*1024. Therefore, the speedup ratio of MapReduce architecture under different computing nodes is improved by comparing the two resolution image data. It can be seen from the comparison between Fig. 11 and Fig. 12 that with the increase of the number of nodes, the operation efficiency of MapReduce will be improved, and the acceleration ratio is about 1.1 to 1.23. The more number of cluster nodes, the higher speedup ratio will be, which shows the improved scheduling plays an important role on the multi-node clusters. The main reason for the acceleration ratio change: First, the more cluster nodes, the higher the speedup ratio, which shows that the improved scheduling algorithm is more effective on multi-node clusters. Secondly, the output data must be backed up by the network. Using pipeline scheduling will improve the efficiency of backup. Finally, the actual Hadoop cluster is separated from the node CPU and IO, and can be read and written at the same time. The cluster is built on the virtual machine, and the performance is different from

the complete distributed cluster, which affects the overall computing performance. In summary, the improved MapReduce framework speeds up the processing performance, so the improvements made in this article are obvious compared to traditional Hadoop's MapReduce framework.

VI. CONCLUSION

This paper implements the image threshold segmentation based on Hadoop framework by studying the application of distributed system to image processing. By designing the logical structure and image read/write interface of MapReduce parallel computing framework, the threshold segmentation of massive glass defect images are completed.

In addition, based on the original MapReduce parallel computing framework, we further improved the intermediate processing process Shuffle, and complete the Shuffle operation parallelization through data localization and pipeline scheduling to improve the operation delay. Experiments show that the data processing speed of improved MapReduce parallel computing framework is not only significantly improved, but also the timeliness of the system is improved. In our future work, we will focus on three areas. First, we will try some image segmentation methods to find the best segmentation effect. Secondly, while improving the MapReduce delay, it will also reduce the accuracy of the operation. How to improve the contradiction between the two is also an important direction in the future work. Third, we will study how to extend the Hadoop cluster adaptively. The number of the data nodes in the cluster can be adjusted according to the size of the task. In this way, the system can save computing resources when the input data is small, and expand when the input data is large, so as to ensure the stability of the system performance.

REFERENCES

[1] Y. Jin, Z. Wang, L. Zhu, J. Yang, and B. Wei, "Study on glass defect inspection based on projecting grating method," *J. Test. Eval.*, vol. 41, no. 2, Mar. 2013, Art. no. 20120008.

[2] K. Sachdeva and A. Girdhar, "A technique for glass defect detection," *Int. J. Innov. Res. Develop.*, vol. 2, no. 13, pp. 25–31, 2013.

[3] J. Xing, R. Sieber, and M. Kalacska, "The challenges of image segmentation in big remotely sensed imagery data," *Geographic Inf. Sci.*, vol. 20, no. 4, pp. 233–244, Oct. 2014.

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[5] S. Vemula and C. Crick, "Hadoop image processing framework," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2015, pp. 506–513.

[6] K. Bok, J. Hwang, J. Lim, Y. Kim, and J. Yoo, "An efficient MapReduce scheduling scheme for processing large multimedia data," *Multimedia Tools Appl.*, vol. 76, no. 16, pp. 17273–17296, Aug. 2017.

[7] Y. Guo, J. Rao, D. Cheng, and X. Zhou, "IShuffle: Improving Hadoop performance with shuffle-on-write," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1649–1662, Jun. 2017.

[8] Y. D. Wang, X. Y. Que, W. K. Yu, D. Goldenberg, and D. Sehgal, "Hadoop acceleration through network levitated merge," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, New York, NY, USA, 2011, pp. 1–10.

[9] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality in MapReduce," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2012, pp. 419–426.

[10] Y. R. Zhao, W. P. Wang, D. Meng, X. F. Yang, S. B. Zhang, J. Li, and G. Guan, "A data locality optimization algorithm for large-scale data processing in Hadoop," presented at the IEEE Symp. Comput. Commun. (ISCC), Cappadocia, Turkey, 2012, pp. 72–81

[11] W. C. Kim, C. Baek, and D. Lee "Measuring the optimality of Hadoop optimization," *Comput. Sci.*, to be published.

[12] H. Vo, J. Kong, and D. Teng, "Cloud-based whole slide image analysis using MapReduce," in *Proc. VLDB Workshop Data Manage. Anal. Med. Healthcare*. Cham, Switzerland: Springer, 2016, pp. 62–77

[13] E. Fujishima and S. Yamaguchi, "Improving the I/O performance in the reduce phase of Hadoop," in *Proc. 3rd Int. Symp. Comput. Netw. (CANDAR)*, 2015, pp. 82–88.

[14] J. J. Li, J. Wu, X. L. Yang, and S. Q. Zhong, "Optimizing MapReduce based on locality of K-V pairs and overlap between shuffle and local reduce," presented at the 44th Int. Conf. Parallel Process., Beijing, China, 2015.

[15] J. Li, J. Wang, B. Lyu, J. Wu, and X. Yang, "An improved algorithm for optimizing MapReduce based on locality and overlapping," *Tinshhua Sci. Technol.*, vol. 23, no. 6, pp. 744–753, Dec. 2018.

[16] Y. Khalil, M. Alshayeji, and I. Ahmad, "Distributed whale optimization algorithm based on MapReduce," *Concurrency Comput. Pract. Exp.*, vol. 31, no. 1, p. e4872, Jan. 2019.

MAOZHEN LI received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 1997. He is currently a Professor with the Department of Electronic and Computer Engineering, Brunel University London, U.K. His main research interests include high performance computing, big data analytics, and intelligent systems with applications to smart grid, smart manufacturing, and smart cities. He has over 180 research publications in these areas, including four books. He has served over 30 IEEE conferences. He is also a Fellow of the British Computer Society and the IET. He is on the editorial board of a number of journals.

LU MENG received the master's degree from the School of Information and Communication Engineering, North University of China, in 2020. His research interest is in big data analytics.

JIAYING WANG is currently pursuing the master's degree with the School of Electrical Engineering and Telecommunications, University of New South Wales. Her research interests are in the fields of image processing and optical fiber communications.

YONG JIN received the Ph.D. degree from the North University of China, in 2013. He is currently a Professor with the School of Information and Communication Engineering, North University of China. His research interests are in the areas of image processing, online inspections, and big data analytics.

BINYU HU received the master's degree from the School of Information and Communication Engineering, North University of China, in 2020. Her research interests are in deep learning and image processing.

YOUXING CHEN received the Ph.D. degree from the North University of China, in 2010. He is currently a Professor with the School of Information and Communication Engineering, North University of China. His research interests are in the areas of image processing, signal processing, and nondestructive testing.