DIGITAL VIDEO WATERMARKING ON CLOUD COMPUTING ENVIRONMENTS

Chu-Hsing Lin*, Chen-Yu Lee**, Shih-Pei Chien*

* Department of Computer Science, Tunghai University,
181, Sec 3, Taichung Port Road, Taichung, Taiwan,
E-mail: chlin@go.thu.edu.tw

** Department of Computer Science, National Chiao-Tung University,
1001 Ta-Hsueh Road, HsinChu, Taiwan
E-mail: chenyu@cs.nctu.edu.tw

ABSTRACT

Due to the need of rapidly distribute digital video, the use of digital watermarking to protect the digital copyright has become very crucial in the era of the Internet. Further, the process for video watermarking is very time consuming. For fast implementation of digital video watermarking, in this article we proposed the use of the Hadoop distributed computing system, where the original video is split, for the different requirements to realize watermarks embedding. The Hadoop distributed file system together with task decomposition are discussed, and the results obtained using the software MapReduce are analyzed. Both technologies are applied to the designed system and we found that the Hadoop distributed computing features play a key role in reducing the required computation time.

KEYWORDS

Video watermarking, Cloud computing, Hadoop, MapReduce, Unseen visible watermarking.

1. INTRODUCTION

The intellectual property (IP) of digital images has become an important issue in the digital age. Since digital image is of characteristic being easily copied or modified, the protection of digital image is a very crucial research topic in information security.

Digital watermarks can be divided into two categories: visible watermarks [1] [2] and invisible watermarks [3] [4] [5]. As the name suggested, an embedded visible watermark can be easily identified by the naked eye and its main purpose is the declaration of copyright or ownership [6] [7]. However, its disadvantage is that it destroys the appearance of the original image. In addition, advances in image processing techniques have made it easy to remove visible watermarks; therefore, visible watermarks are less suitable for the current digital environment.

On the other hand, an embedded invisible watermark cannot be easily recognized with the naked eye and a special method is required to remove it. Although this results in an increased watermark security, it is not ideal from a convenience perspective for the identification of images, where the visibility of the watermark is important.

In 2007, Huang et al. [8] proposed the unseen visible watermarking (UVW), which was a breakthrough in the watermarking field. The disadvantage of this technique is that the watermark itself has to be a binary image. Because of this restriction of the UVW scheme, its practical application is limited. In 2011, Lin et al. [9] presented the unseen visible gray watermarking (UVGW), and found that it is more convenient than the UVW to be applied to today's digital imaging technology. A color watermark is converted to a grayscale watermark that can be easily embedded in its cover image, and the degree of recognition of the color watermark can be retained.

Hadoop is with characteristics of reliability, scalability, and distributed computing, it is also one kind of open source software. Here, we exploit some of the features. Hadoop distributed file systems (HDFS) exhibit low latency access to high throughput data applications. Moreover, with MapReduce, [10][11] which is HDFS on a software development framework, distributed processing of large datasets is performed. There is no single point of computing power, and MapReduce is used to support watermark computation in this research.

In this article, Hadoop is applied as a solution to provide customers with not only flexibility of computing capacity but also a methodology for high-performance computing that requires a high throughput IO. The rest of the article is organized as follows. Section 2 depicts the background of the unseen visible grayscale watermarking that is used in our implementation. Section 3 gives the system architecture and the steps of the proposed cloud watermarking system. Section 4 shows the experimental results, and finally, Section 5 gives the conclusion.

2. BACKGROUND

Unseen visible gray watermarking (UVGW) [9] based on Gamma Correction was proposed in 2011. In the research, the authors proposed an improved unseen visible watermark scheme using multi-parameters adjustable Gamma Correction to reduce the distortion, which can increase the color saturation. From the experiments, it showed high PSNR values of the stego-images. Comparing to UVW, the UVGW showed better characteristic for applications in today's digital imaging technology. In real application, a color floating watermark can be easily converted to a grayscale watermark. Then the watermark is embedded in the cover image, and the stego-image is of high perception.

The watermark embedding process for the UVGW is described briefly as follows:

Step 1 ： Perform a gamma correction on the original image I to obtain $G(i)$, and find the largest gradient value i^* of $G(i)$.

$$G(i), 0 \le i \le 255, 0 \le G(i) \le 255 \tag{1}$$

$$i^* = \arg\max_i \nabla G(i) \tag{2}$$

Step 2 ： Perform a de-noising operation to obtain I'_j in Eq. (3), where the δ_j value is the de-noising parameter for the maximum value of δ_{max}.

$$I'_j = D(I_G, \delta_j), \delta_j \le \delta_{\max} \tag{3}$$

Step 3 ： Choose the best embedding region R.

Step 4 ：Perform the de-noising until Eq. (6) is satisfied.

$$P = \arg\max \sum_{x=x_0}^{x_0+W_l-1} \sum_{y=y_0}^{y_0+W_w-1} \left| I'_j(x,y) - i^* \right| \tag{4}$$

$$R = \{ I'_j(x,y) \mid x_0 \le x \le x_0 + W_l - 1, \quad y_0 \le y \le y_0 + W_w - 1 \} \tag{5}$$

$$S = \sum_{x=x_0}^{x_0+W_l-1} \sum_{y=y_0}^{y_0+W_w-1} D\left(I'_j(x,y) - i^*\right) \le W_l \times W_w \times T \tag{6}$$

$$D(k) = \begin{cases} 1, \ if \ k = 0 \\ 0, \ otherwise \end{cases} \tag{7}$$

In the RGB color mode, saturation has a certain proportion of R, G, and B values. One of the colors was chosen as the color index, which had the maximum value by Eq. (8) in the embedded region. If a pixel had the same R, G, and B values, the index was determined to be the color that appeared most around the pixel.

$$MAX = \max(R, G, B) \qquad MIN = \min(R, G, B) \tag{8}$$

We performed the analysis on a grayscale watermark that is used to identify the distribution of gray watermarks, as in Eq. (9).

$$L = \left(avg \sum_{x=x_0}^{W_l-1} \sum_{y=y_0}^{W_w-1} W(x,y) \right) \times \sum_{x=x_0}^{W_l-1} \sum_{y=y_0}^{W_w-1} W(x,y) \tag{9}$$

The intensity adjustment of each pixel is according to its index in the watermark, which is embedded in the region. The dynamic adjustment prevents the adjusted color from exceeding 255. In the grayscale watermark, the pixel intensity value is adjusted in accordance with the gray level; hence, the adjustment variable ω in Eq. (10) was determined by L. When the above was completed, the grayscale watermark was successfully embedded in the original image.

$$\tilde{I}(x,y) = \begin{cases} I'_j(x,y) - \omega, & if \ \left(I'_j(x,y) + \omega\right) > 255 \\ I'_j(x,y) + \omega, & otherwise \end{cases} \qquad \omega = W(x - x_0, y - y_0) \pm L \tag{10}$$

3. CLOUD VIDEO WATERMARKING

In this section, we introduce the system architecture and method of implementation for the proposed cloud video watermarking system. In our video watermarking, the Hadoop infrastructure plays a key role in the entire

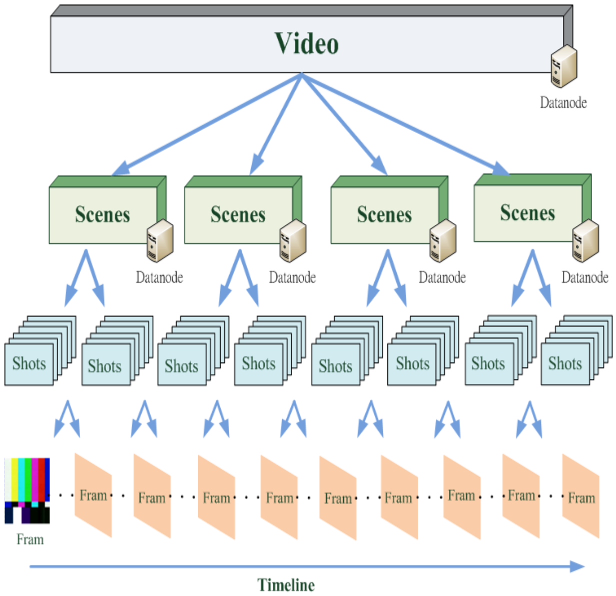

Figure 1 ：Video Segmentation

system. Because video watermarking consumes a lot of computing resources, the greatest advantage of our design is that the system provides a high aggregate data bandwidth, which is scaled to many nodes in a single cluster. To attain this target, we implemented the idea by using the following steps and performed some experiments.

The video portion of a general movie is composed of a large and continuous static image screen, the main movie screen in academia detection, and multi-static screen eigenvalue differences in the analysis of the film to identify points at which segmentation occurred in the video. The film is segmented or the main screen is removed to create the hierarchy of the video browsing framework.

Figure 1 showed that the navigation structure of the entire movie from top to down and can be divided into the movie, scenes, clips, and screen. The static screen is a two-dimensional plane, and the film can be seen on the timeline, which is a three-dimensional cube consisting of multiple planes. Every piece of film is composed of a number of scenes, and each scene can be divided into several fragments.

The film length can be split after the use of time to control a few seconds to obtain an image and can also be set to obtain the size of the image, and video clips, and the end of the treatment of these pictures in the default folder.

Figure 2 showed the flow of the cloud video watermarking scheme. We employ the Hadoop

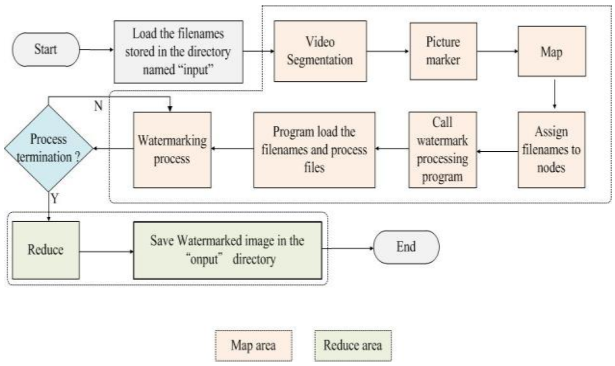

Figure 2：The flow of cloud video watermarking

infrastructure and the MapReduce technology on video watermarking and specify the input and output locations, supply map, and reduced functions. Then, the job client of the Hadoop will submit the job from the namenode and configure it to the jobtracker, which will distribute the software to the slaves, schedule the tasks, and monitor the status of the tasks. It will also provide status and the related information to the job client. The job client will take the job, execute the assignment, and send feedback to the namenode. The results of each job client will be sent to the namenode and shown in the logs in the sub-directory named "logs" of the Hadoop directory.

4. EXPERIMENTAL RESULTS

4.1 Experimental Specification

We used a 166 MB video and imported it to the "input" directory that will be processed by the watermarking job. Then, we split it into 5240 pictures, where the size of each image was 480 × 360 pixels. These images will be executed over the Hadoop architecture, in which the watermark embedding process and extracting process are executed eight times and each time a different number of datanodes is used for performance measurement.

Totally, we have a cluster of 8 nodes in our system, and each node is indicated by Node1, …, and Node 8, respectively. The hardware and software specifications of the computing nodes are shown in Table 1.

4.2 Case with Images Distributed Evenly

This section presents the experimental results of execution time on performing video watermarking with the various numbers of nodes. Two methods were used to measure the execution time. The first method divides the 5240 images evenly; in this way, each node will process the same number of images, and the result of the time consumed will include the wait for the slowest host to finish its job. The results are shown in Figure 3. It is apparent that all images were processed in a single node, and it took 8318 minutes to finish the watermarking job, whereas the execution time for the cluster with 8 nodes was 2105 minutes.

4.3 Case with Optimized Computing Capacity

The second method distributes the 5240 images into different numbers of nodes according to the HPC challenge benchmark (HPCC) [12] and presents the results in the Gflops, which represents the measurement in billions of floating point operations per second (FLOPS) that a computer's microprocessor can handle. In this method, the numbers of image are distributed by the host's Gflops, we optimized and sorted the computing node in decreasing order for efficient distribution. After the adjustment using Gflops values generated from the HPC challenge benchmark, the node loading was optimized by the computing capability. Table 2 showed the execution results optimized by the HPC challenge benchmark.

Obviously, in the first method, computing from node1 to node 3 took the same amount of time as the second method to finish the watermarking job, whereas the best execution time for a cluster with 8 nodes was 2105 minutes. In Figure 4, when the distributed number of images was optimized by the HPC challenge benchmark, the best execution time for a cluster with 4 nodes decreased from 5341 minutes to 4627 minutes and the time taken for the cluster computing with 8 nodes decreased from 2105 to 1762 minutes. This means that the number of images distributed by the optimization of the HPC challenge benchmark will enhance the efficiency of this implementation.

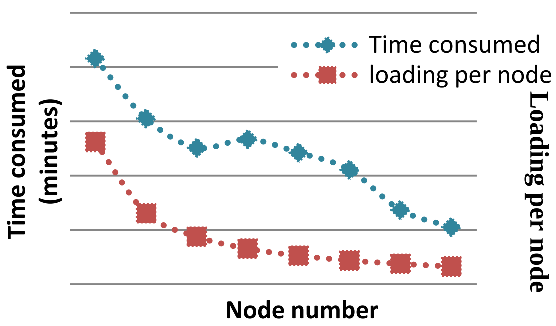

Figure 1 : Execution result chart of images distributed evenly

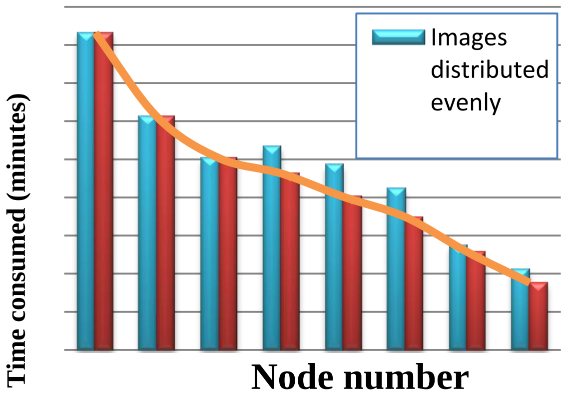

Figure 2 : Comparison of two methods

Table 1 : H/W and S/W specifications of each node

Node	Processor	Memory	Gflops	OS	JAVA
Node 1	Intel i5 CPU650 × 2	1G	1.22e+01	ubuntu-11.10	jre 1.6.0_24
Node 2	Intel i5 CPU650 × 2	1G	1.23e+01	ubuntu-11.10	jre 1.6.0_24
Node 3	Intel i5 CPU650 × 2	1G	1.22e+01	ubuntu-11.10	jre 1.6.0_24
Node 4	Intel i5 CPU650 × 2	1G	1.27e+01	ubuntu-11.10	jre 1.6.0_24
Node 5	Intel i5 CPU650 × 2	1G	1.24e+01	ubuntu-11.10	jre 1.6.0_24
Node 6	Intel i5 CPU750 × 2	1G	1.23e+01	ubuntu-11.10	jre 1.6.0_24
Node 7	Intel i5 CPU750 × 2	1G	1.23e+01	ubuntu-11.10	jre 1.6.0_24
Node 8	Intel i5 CPU750 × 2	1G	1.23e+01	ubuntu-11.10	jre 1.6.0_24

Table 2：Execution results optimized by the HPC challenge benchmark

Computing nodes	# of Nodes	Node Loading (Image numbers)	Time consumed (minutes)
Node 1	1	5240	8318
Node 1,2	2	2620	6106
Node 1,2,5	3	1747	5027
Node (1,2,5),4	4	(1421×3),977	4627
Node (1,2,5,6), 4	5	(1121×4),756	3561
Node (1,2,5,6),(4,7)	6	(1028×4),(564×2)	2732
Node (1,2,5,6),(4,7),3	7	(962×4),(512×2),368	2163
Node (1,2,5,6),(4,7),(3,8)	8	(902x4),(427×2),(389×2)	1762

5. Conclusion

In the article, we offer a better way to reduce the time consumption for applications with large computing load such as video watermarking. In addition, the proposed method can also be used daily by users over the cloud environment not only by Hadoop but also others applications powered by Google and Yahoo. The computation of digital watermarking technology in cloud environments will play a very important role in the protection of intellectual property in the age of digitalization.

The experimental results showed that the cloud computing environment offers a better solution than a single node to satisfy the requirements for high computing resources. Besides, our proposed architecture has a wide range of uses for data encryption. The proposed method and architecture will be helpful in enhancing data security in private companies, such as online encryption for electronic commerce, and may be used to design a security gateway to detect illegal attempts to upload the watermarked files on the Internet.

References

[1] Liu, R., Tan, T.: An SVD-based Watermarking Scheme for Protecting Rightful Ownership. IEEE Transactions on Multimedia 1, 121--128 (2002).

[2] Tang, X., Yang, L., Yue H., Yin, Z.: A Watermarking Algorithm Based on the SVD and Hadamard Transform. In: Proc. International Conference on Communications Circuits and Systems (2005).

[3] Ge, X., Hao, T.: Research on Application of Immune Digital Watermark Algorithm. In: Proc. International Conference on Computer Science and Software Engineering (2008).

[4] Wolfgang, R. B., Podilchuk C. I., Delp, E. J.: Perceptual Watermarks for Digital Images and Video. Proceedings of the IEEE 87(7), 1108--1126 (1999).

[5] Zheng, L., Xie, K., Li, Y., Liu, H., Li, T.: A Digital Watermark Scheme for Vector Graphics. In: Proc. International Conference on Image Analysis and Signal Processing (2010).

[6] You, F. C., Li Y., Tu, X. Y.: Research on Mobile Agent-based Digital Watermark Tracking System and Communication Security. In: Proc. International Conference on Wavelet Analysis and Pattern Recognition (2007).

[7] You F. C., Li, F. Z.: The Research of Digital Copyright Protecting Mechanisms Based on Mobile Agent and Digital Watermark. In: Proc. International Conference on New Trends in Information and Service Science (2009).

[8] Huang, C. H., Chuang, S. C., Huang, Y. L., Wu, J. L.: Unseen Visible Watermarking: A Novel Methodology for Auxiliary Information Delivery via Visual Contents, IEEE Transactions on Information Forensics and Security 4(2), 193--206 (2009).

[9] Lin, C. H., Lee, C. Y., Lu, S. Y., Chien, S. P.: Unseen Visible Watermarking for Gray Level Images Based on Gamma Correction. In: Proc. International Conference on Future Generation Communication and Networking. CCIS, vol. 265, pp. 236-243, Springer (2012).

[10] Jaliya, E., Shrideep P., Geoffrey, F.: Mapreduce for Data Intensive Scientific Analyses. In: Proc. IEEE Fourth International Conference on eScience (2008).

[11] Ranger, C., Raghuraman, R., Penmetsa, A., Bradski G., Kozyrakis, C.: Evaluating Mapreduce for Multi-core and Multi-Processor Systems. In: Proc. IEEE 13th International Symposium on High Performance Computer Architecture (2007).

[12] HPC Challenge Benchmark, http://icl.cs.utk.edu/hpcc/index.html