A Locality-aware Task Scheduling Algorithm for Video Transcoding over Heterogenous MapReduce Cluster

Xiangjun Deng
Hunan University
Changsha, China
dxj3968415@hnu.edu.cn

Jing Huang
Hunan University
Changsha, China
jingh@hnu.edu.cn

Renfa Li
Hunan University
Changsha, China
lirenfa@hnu.edu.cn

ABSTRACT
MapReduce is emerging as a very promising technology for high computational task, for instance video transcoding. Because of the complexity of video decoding and encoding in transcoding, we divide the video transcoding job into several segments, then mapping these segments over heterogenous MapReduce cluster to handle. In this paper, we propose a locality-aware heuristic transcoding task scheduling algorithm to balance the segment transmission time and the task transcoding time, LA-MCT, which classifies all segments as local and remote segments, and then maps these segments to an optional machine using the MCT load balancing strategy. The optimal scheduling schema could be selected by traversing all local and remote segments pairs. Large number of numerical simulation experiments validate that the proposed algorithm can generate the minimum entire finish time compared with the existing algorithms.

CCS Concepts
• **Computer systems organization → Cloud computing** • **Computing methodologies → MapReduce algorithms** • **Information systems → Multimedia streaming**

KEYWORDS
MapReduce; Video Transcoding; Locality-aware; Task Scheduling

1. INTRODUCTION
1.1 Background
Video content has been produced, transported and consumed by more and more ways and devices in recent years. When a target device does not support the current format, has limited storage capacity that mandates a reduced file size, or need to convert or obsolete data to a better-supported or modern format, a unified mechanism called video transcoding is required. It can convert the encoded video stream into another video stream for video content adoption. Currently, a general video transcoder can achieve the following main purposes: bit-rate reduction in order to meet network bandwidth availability, resolution reduction for display size adoption, temporal transcoding for frame rate reduction and error resilience transcoding for insuring high QoS [1, 2], as is shown in Fig. 1.

Video transcoding is a computationally heavy process that requires tremendous computing power of the processor. When transcoding large size video on a single-core processor based transcoder, it is hard to achieve acceptable entire finish time of a job due to the limited computing power. Therefore, many studies have focused on parallel transcoding over multi-core processor to speed up transcoding process [3-5]. However, due to the difference of hardware, the computing power over multi-core processor is often difficult to extend with the increase of computing requirements.

MapReduce [6] is a high scalability distributed cloud programming paradigm that can use thousands of computers to handle a complex job in parallel, and is an efficient solution for parallel transcoding acceleration. The framework implementation of MapReduce such as Hadoop is designed and optimized to perform in large homogeneous clusters, but it performs poorly on heterogeneous clusters [7-11]. When a transcoding job is executed on a heterogeneous cluster, the job can not effectively utilize the computing resources of the cluster, resulting in the entire finish time much longer than the optimal value if there is no specific optimization for the cluster. Therefore, it is of great significant to study parallel transcoding acceleration over heterogeneous MapReduce cluster.

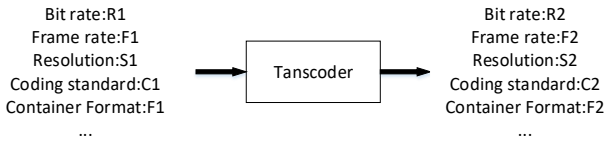


Figure 1. A general transcoder

1.2 Motivation
The entire finish time is the key performance metric in the MapReduce-based video transcoding job. Many efforts are devoted to reduce the entire finish time of a job with the First Come First Serve (FCFS) scheduling strategy [12-14]. The FCFS can reduce the entire finish time by launching tasks in parallel, but it has further improvement scope because it may result in the load balancing problem. In view of this, some load balancing scheduling algorithms were proposed for MapReduce-based video transcoding homogeneous cluster [15-18]. These

ICMSSP '18, April 28–30, 2018, Shenzhen, China

ACM ISBN 978-1-4503-6457-7/18/04…$15.00
https://doi.org/10.1145/3220162.3220191

algorithms can minimize the entire finish time but obtain poorly performance in heterogeneous cluster.

Considering the heterogeneity of transcoding machines in a cluster, two algorithms called Maximizing Minimal Complete (Max-MCT) and Minimum Longest queue Finish Time (MLFT) aim at minimizing the entire finish time of transcoding job over heterogeneous MapReduce cluster were proposed [19, 20]. But MAX-MCT and MLFT neglect the video transmission overhead, resulting in the video transcoding scheduling model is not precise enough. When it comes to the video transmission overhead, Zhao et al. proposed a locality-aware scheduling algorithm called Prediction-based and Locality-aware Task Scheduling (PLTS), it utilize the data locality and combines the benefits of Max-Min and Min-Min to minimize the entire finish time [21]. Although the PLTS algorithm is the state-of-art task scheduling algorithm based on load balancing, cluster heterogeneity and video transmission overhead, however, it did not balance segment transmission time and the task transcoding time. Therefore, there is still plenty scope for minimizing the entire finish time.

1.3 Main Contributions

In this paper, we aim to minimize the entire finish time while taking data-locality into consideration for video transcoding job over heterogeneous MapReduce cluster. We formulate the optimization problem as a Job Shop Scheduling (JSS) problem and solve it by a heuristic algorithm called Locality-aware Minimal Complete Time (LA-MCT). In our algorithm, we introduce two concepts, local segment set and remote segment set. The local segment set means that segment in the set is allowed only to be mapped to the machine that stores its replica. In contrast, the segment from the remote segment set is allowed only to be mapped to the machine that not stores its replica. For a segment, the segment complexity is the product of the computation capacity and the time of transcoding operation. The algorithm continuously reduces the sum of the segments complexity of local segment set from the maximum to zero, corresponding to the sum of the complexity of the remote segment set is increasing. Considering each change of the sum of the complexity of the local segment set, the algorithm can determine a pair of local segment set and remote segment set, and then use the MCT procedure to map the segments in local segment set and remote segment set to heterogeneous cluster successively. In order to find the optimal solution, we traverse all the possible cases of the sum of the segment complexity of the two sets. Finally, the minimum entire finish time is obtained. The contributions of this study are summarized as follows.

(1) We propose a heuristic algorithm called LA-MCT, which can balance the segment transmission time and the task transcoding time to minimize the entire finish time.

(2) Experiments using simulation applications are extensively conducted and the results of these experiments consistently verify that the proposed LA-MCT can generate minimum entire finish time compared with the existing algorithms under different scale conditions.

The rest of this study is organized as follows. Section 2 re-views the related work. The video transcoding system architecture is presented in Section 3. Section 4 describes problem formulation. Section 5 proposes our task scheduling algorithm LA-MCT, Section 6 verifies the proposed algorithm by large number of experiments. Section 7 concludes this study.

2. RELATED WORK

Video transcoding is a time-consuming and resource-consuming process. With the dramatic increase in the size of video content, the traditional video transcoding system [3-5] has insufficient storage capacity and scalability. Therefore, some researchers are committed to video transcoding services built on heterogeneous cloud environment [12-14]. Huang et al. proposed a cloud-based proxy that delivers high-quality Internet streaming [12], which defined the video transcoding process as an on-line scheduling problem. However, they didn't take the sub-task launching overhead into consideration. [13] proposed a MapReduce-based system that can convert various video codec formats into the MPEG-4 video format. Li et al [14] implemented a cloud transcoder. It only requires users to upload transcoding request instead of video content, and then downloads the original video content from the Internet, transcodes video according to user needs, and transfers the transcoded video back to the user at high data rates through accelerated in-cloud data transfer. Three works mentioned above only shorten the transcoding time from the perspective of parallelization without considering the load balancing of tasks in heterogeneous cloud cluster.

Scheduling independent tasks to heterogeneous distributed computing system with the goal of minimizing the time is similar to the JSS problem, which is an NP-hard problem [22]. In order to solve such a complex problem, eleven heuristic algorithms had been compared and the e client one is Min-min algorithm [23]. Moreover, Min-min algorithm is equal to minimum complete time (MCT) algorithm in video transcoding model owing to the time span of transcoding is in proportion to segment complexity. But these algorithms are general-purpose algorithms that do not take into account the nature of video transcoding. Therefore, F. Lao presented a model that considered the task launching overhead and addressed the problem using the Max-MCT algorithm [19]. The algorithm abstracts the assignment problem into the virtual knapsack problem and assigns complex segments to powerful machine to reduce their time waste on sub-task launching overhead. Lin et al. [20] proposed a novel parallelizing video transcoding framework, which includes task pre-analysis, adaptive threshold segmentation and minimal finish time (MFT) scheduling. Based on this framework, they proposed a load balancing scheduling algorithm called MLFT, the algorithm shortens the entire finish time by dividing high complexity task into multiple sub-tasks and redistributing tasks in the allocation queue. Unfortunately, the researchers in [19, 20] did not consider the transmission overhead of segments between machines. A locality-aware scheduling algorithm can spend time primarily on transcoding instead of video content transmission. Therefore, in [21], a locality-aware task scheduling algorithm were proposed called PLTS, which combines the benefit of two traditional heuristic algorithm, Max-min and Min-min, to average the entire finish time and make load balancing over heterogenous cluster. However, the algorithm does not balance the segment transmission time and task execution time. Therefore, there is still plenty scope for minimizing the entire finish time.

3. SYSTEM ARCHITECTURE

Fig. 2 shows the MPEG bit-stream structure in an abstract way. Macroscopically, the video is composed of a series of video sequences, a video sequence consists a group of pictures (GOP) header and some GOPs. In video coding, GOP structure specifies the order in which intra-frames and inter-frames are arranged. The GOP is a collection of successive pictures within a coded video

stream. Each coded video stream consists of successive GOPs, from which the visible frames are generated. Encountering a new GOP in a compressed video stream means that the decoder does not need any previous frames in order to decode the next ones. Therefore, GOP is an atomic unit that can perform video transcoding procedure independently.

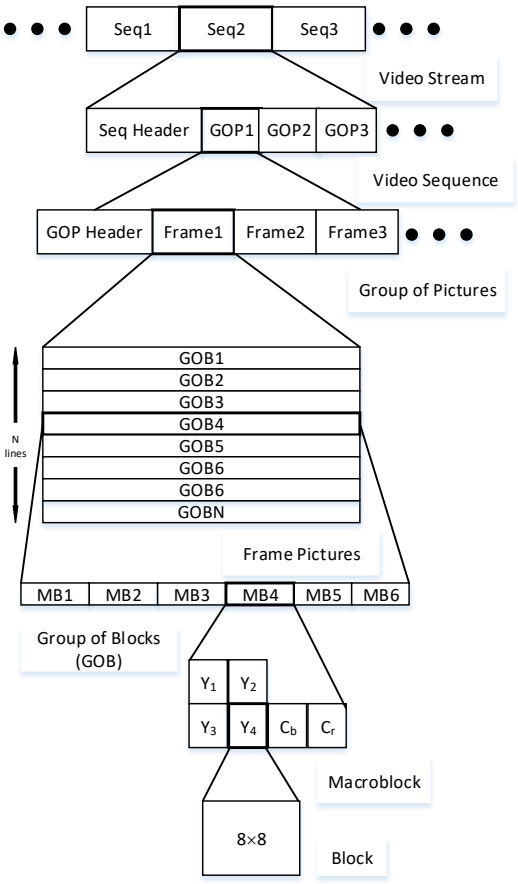


Figure 2.The structure of MPEG video stream

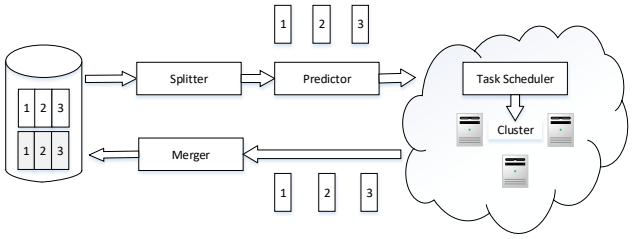


Figure 3.MapReduce-based video transcoding framework

Fig. 3 shows our MapReduce-based cloud video transcoding framework, which mainly consists of splitter, predictor, task scheduler and merger. Because the transcoding process of segment is independent of each other, the transcoding process of segment on the machine is called a task. In a word, the task is an instance of transcoding in the machine. the segment and task are the same concept in this article, but for the sake of description we refer to task as segment in some scenarios.

When the system receives a transcoding request, it uses splitter to split the video into several video segments at GOP level. In general, comparing to the segment transmission time and transcoding time, is negligible. On the one hand, the segment contains several GOPs instead of one, because if a task contains only one GOP, the entire transcoding job will generate too many tasks, while in Hadoop, the life cycle of a MapReduce task main includes two stages: launching, executing. Obviously, the smaller the granularity of segment, the greater the launching stage is task-independent, too many tasks will waste too much time on the task to launch, thereby increasing the execution time of the entire transcoding job. One the other hand, a segment can not contain too many GOPs. If the number of GOPs in a segment reaches the value that makes up a video sequence, segment granularity such as video sequence result in higher latency of the system affecting the streaming behavior of the video being processed. The reason is that it takes more time to process a video sequence than a GOP or a frame [1, 24].

After the splitter divide the job into segments, the system uses predictor to predict segment complexity. For a segment, the segment complexity is the product of the computation capacity and the time of transcoding operation. We can utilize the complexity prediction algorithm proposed in [21] to estimate the video segment complexity precisely, therefore, this is not the focus of this paper. In this paper, the complexity of each segment is assumed. Different GOP contain different numbers of frames, so the complexity of each frame tends to be different, so the segments are heterogeneous. Generally, the cluster providers several machines with different computation capacity, so the hadoop cluster is always heterogeneous. The capacity of every machine can be estimate and normalized by the historical video transcoding logs.

In our video transcoding system, a task scheduler is designed to map independent transcoding tasks onto cluster. Each machine in the cluster perform a series transcoding tasks assigned by the scheduler independently. Once a task execution is completed, the transcoded segments will be transferred to the merger, and the merger concatenate all transcoded segments together after all these segments arrival. Finally, a video sequence will be generated and stored in the disk.

4. PROBLEM FORMULATION

For a given video content, the time spent on splitter, predictor and merger are negligible [21]. We only focus on the video transocding process at the cluster and regard it as a scheduling problem.

A Hadoop cluster consists of heterogeneous machines $J=(1,2,...,M)$, where M is the size of set J. The transcoding capacity of each machine is denoted as $\mathrm{P}=(p_1,p_2,...,p_M)$. The splitter divide the video sequence into N segments $V=(v_1,v_2,...,v_N)$, the size of segments from V is denoted as $S=(s_1,s_2,...,s_N)$. The complexity of each segment can be predicted by predictor, and is denoted as $\mathrm{C}=(c_1,c_2,...,c_N)$.The sum of the segments complexity in C is $C_{sum}=\sum_{c_i\in C} c_i$;

In general, the transcoding time of segment c_i performed in machine p_j is composed of segment transmission time, task-launching overhead and segment transcoding time in a MapReduce job. The segment transmission time is related to s_i and the bandwidth between the storing machine and transcoding

machine. the task-launching overhead is a task-independent constant and denoted as o_m . The transcoding time is proportional to segment complexity c_i and inversely proportional to the capacity of machine p_j . Then the expect transcoding time (ETT) of segment v_i on machine m can be calculated by:

$$ETT(v_i, m) = \frac{c_i}{p_m} + \mathrm{d}^i_{m',m} + o_m \tag{1}$$

where $d^i_{m',m}$ is transmission time of segment v_i from machine m' to m , and it can be calculated by:

$$\mathrm{d}^i_{m',m} = C_{m',m} \times \frac{s_i}{B_{m',m}} \tag{2}$$

where $C_{m',m} \in \{0,1,3\}$, when $C_{m',m} = 0$, it represents the task assigned to a node-level local machine, the segment transmission time here is 0 , otherwise, the segment transmission time inversely proportional to the network bandwidth $B_{m',m}$. $C_{m',m} = 1$ means the task assigned to a rack-level local machine and $C_{m',m} = 2$ represents to a remote-level machine.

After all the tasks are mapped to the corresponding machines using the task scheduling algorithm, each machine will perform tasks independently. The tasks set on machine j can be denoted as θ_m , and the expect tasks finish time of θ_m is:

$$\begin{aligned} EFT_{\theta_m} &= \sum_{v_i \in \theta_m} ETT(v_i, m) \\ &= \sum_{v_i \in \theta_m} (\frac{c_i}{p_m} + d^i_{m',m}) + |\theta_m| \times \theta_m. \end{aligned} \tag{3}$$

The task scheduling result is denoted as $\Theta = \{\theta_1, \theta_2, ..., \theta_M\}$. Once all the tasks in Θ is done, the transcoding job is complete, and the expect entire finish time of job FT can be denoted as

$$FT = \max_{\theta_m \in \Theta}(EFT_{\theta_m}). \tag{4}$$

The goal of our algorithm is to find the scheduling strategy to minimize the entire finish time of job, which is bounded by the maximal finish time of all the machines, and the optimization problem can be formulated as

$$\begin{aligned} &\min_{\Theta}(FT) \\ &s.t. \Theta = \{\theta_1, \theta_2, ..., \theta_M\} \\ &\bigcup_{\theta_m \in \Theta} \theta_m = J \\ &\forall \theta_{m1}, \theta_{m2} \in \Theta, \theta_{m1} \bigcap \theta_{m2} = \phi. \end{aligned} \tag{5}$$

The formulated optimization problem in Eq. (5) is similar to the Job Shop Scheduling problem (JSS). The JSS problem has been proved to be NP-hard, so our scheduling problem is also NP-hard [21]. It is complicated and time consuming to obtain a global optimal solution. If there is no effective algorithm, people have to traverse all the possible solutions, resulting in a time complexity of $O(N^M)$. Clearly, it is unrealistic to find the optimal solution.

5. TASK SCHEDULING ALGORITHM

In this section, a novel heuristic algorithm LA-MCT is proposed. The core idea of our algorithm is based on data-locality and the load balancing strategy. The algorithm iteratively divides all segments into local segment set and remote segment set. According to the characteristics of the two task sets, we map the segments to an optional machine using the MCT load balancing strategy. Finally, the optimal division schema can be found by comparing the FT in each iteration.

For each segment, there are 3 replicas in HDFS by default. According to whether the video is transcoded locally, V is divided into two segment subsets local segment set L and remote segment set R . Considering a segment from V , if only allowed to be mapped to the machine where stores its replica, then we classify it into the local segment set. All such segments are made up of $L = \left(c_1, c_2, ..., c_{|L|}\right)$. In contrast, if a segment is allowed only to be mapped to the machine where does not store its backup, then it is classified into the remote segment set. All such segments form $R = \left(c_1, c_2, ..., c_{|R|}\right)$. $|L|$ and $|R|$ is the size of set L and R . The sum complexity of all segments in L and R are denoted as C_L and C_R . Evidently, they satisfy the following equation

$$\begin{aligned} C_{sum} &= \sum_{c_l \in L} c_l + \sum_{c_r \in R} c_r \\ &= C_L + C_R. \end{aligned} \tag{6}$$

Algorithm 1 The Segment Set Division Algorithm

Input: Segments set $V = \{v_1, v_2, ..., v_N\}$, Complexity set $C = \{c_1, c_2, ..., c_N\}$, The sum complexity of local segment set C_L

Output: Local segment set L, Remote segment set R

```
Set L = φ; Set R = φ;
temp = 0; isLocal = false;
for (each segment v_i in V) do
    if (isLocal) then
        temp = temp + c_{v_i};
        if (temp ≤ C_L) then
            L = L ⋃ v_i;
        else
            Split v_i into two segments v_{i_1} and v_{i_2}, satisfy c_1(the complexity of v_{i_1}) less than and as close as possible to C_L − (temp − c_{v_i});
            L = L ⋃ v_{i_1}; R = R ⋃ v_{i_2}; isLocal = false
        end if
    else
        R = R ⋃ v_i;
    end if
end for
return L, R.
```

Let's initialize $C_L = C_{sum}$, $C_R = 0$, the algorithm keeps reducing C_L until it is less than 0. Considering the possibly large C_L , a step value C_{step} should be provided for the application to reduce the value of C_L . However, it is a complicated problem to decide the size of C_{step} according to the size of C_L . If C_{step} is too large, it will reduce the accuracy of the computation result; If C_{step} is too small,

it will increase the computation time of the algorithm. Therefore, it is necessary to achieve a reasonable trade -off between precision and computing time. For this reason, we let C_{step} be the minimum segment complexity in V .

For each round of iteration, a new C_L value will be generated, and there is an C_R corresponding to it according to Eq. (6), and

then we use the task set division to determine local segment set L and remote segment set R. Due to the dispersion of complexity c_i in C, for a given pair of C_L and C_R, the procedure select a segment from the set V, added to the local segment set L, until the sum complexity of all segments in L less than and as close as possible to C_L; Apparently, the remaining segments in V form the set R. Algorithm 1 describes the step of division of task set V into L and R. In Line 9 and Line 10, since a segment contains multiple independent GOP units, we can further split the segment to meet the requirement of the sum complexity of all segments in L less than and as close as possible to C_L.

Algorithm 2 The MCT Algorithm

Input: Capacity of Machines set $P = \{p_1, p_2, ..., p_N\}$, Segment v_i, Segment complexity C_i

Output: The minimal complete time Min, The machine m that has minimal complete time

1: Get the set of corresponding machines $machines$
2: $min = \infty$;
3: **for** (each machine m in $machines$) **do**
4: Calculate $ETT(v_i, m)$ and EFT_{θ_j} by Eq. (1) and (3);
5: **if** $(ETT(v_i, m) + EFT_{\theta_j} < min)$ **then**
6: $Min = min; m = j$;
7: **end if**
8: **end for**
9: **return** Min,m.

Algorithm 3 The LA-MCT Algorithm

Input: Capacity of Machines set $P = \{p_1, p_2, ..., p_N\}$, Segments set $V = \{v_1, v_2, ..., v_N\}$, Complexity set $C = \{c_1, c_2, ..., c_N\}$

Output: The scheduling result $\Theta_{best} = \{\theta_1, \theta_2, ..., \theta_M\}$

1: $\forall m \in J$, Set $\theta_m = \phi$; $C_L = C; minFt = \infty; \Theta_{best} = null$
2: Sort the set P by descending order of capacity;
3: Get the C_{step} in C;
4: **while** $(C_L \geq 0)$ **do**
5: Generate L and R using Algorithm 1;
6: Sort the segments in L,R by descending order of complexity, respectively;
7: **for** (each segment v_i in L) **do**
8: Get the machine m that has minimal complete time using Algorithm 2;
9: Assign segment v_i to machine m;$\theta_j = \theta_j \bigcup v_i$;
10: **end for**
11: **for** (each segment v_i in R) **do**
12: Get the machine m that has minimal complete time using Algorithm 2;
13: Assign segment v_i to machine m;$\theta_j = \theta_j \bigcup v_i$;
14: **end for**
15: Calculate FT by Eq. (4);
16: **if** $(minFt < FT)$ **then**
17: $minFt = FT; \Theta_{best} = \Theta$;
18: **end if**
19: $C_L = C_L - C_{step}$;
20: Clear the scheduling result Θ
21: **end while**
22: **return** Θ_{best}.

When the set L and R are generated by the Algorithm 1, we traverse the segments from L and R in descending complexity order, and then employ MCT procedure to map these segments. Sorting in descending order of complexity and then mapping tasks can make long tasks execute first. Allowing long tasks to take precedence prevents some processors from being idle at the end of the job execution. In order to avoid repeatedly using a list of MCT procedure, we define it in Algorithm 2.

When all the segments from L and R are mapped to the relevant machine, we can get the value of FT. We compare each round of FT to find the optimal schema. The step of LA-MCT algorithm are provided in Algorithm 3.

On the basis of above analysis, the LA-MCT algorithm described in Algorithm 3 is proposed. the time complexity is $O(C_{sum} \times N \times M)$.

6. EXPERIMENTS

Considering the ideal distribution of video segments, that is, all segments are transcoded locally and the load distribution is optimal, in other words, the data transmission overhead of a transcoding job is 0. Therefore, in theory, when N segments are performed video transcoding on M machines, the optimal entire finish time is

$$FT_{opt} = \frac{\sum_{i=1}^{N} c_i}{\sum_{j=1}^{M} p_j} + o_m \times \frac{N}{M}. \quad (7)$$

Obviously, in practical environment, some segments need to be transferred to remote machines to transcode to balance the load, in other words, the segment transmission time of a job is often greater than zero. The goal of our algorithm is to make FT as close as possible to FT_{opt}. The factor E_{ft} is employed to evaluate the efficiency of different algorithms, which is denoted as

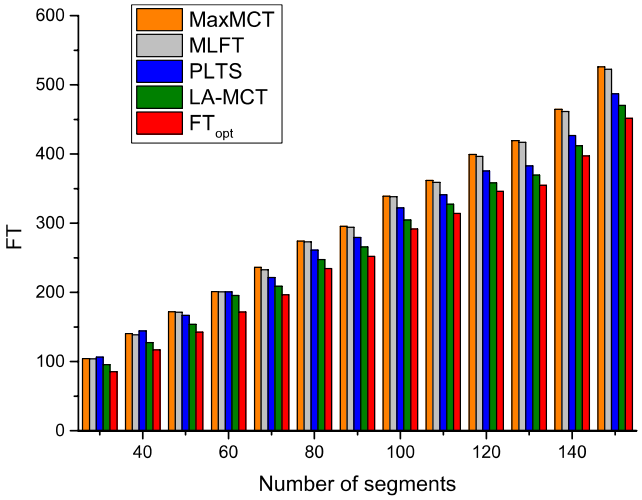
$$E_{ft} = \frac{FT - FT_{opt}}{FT_{opt}} \times 100\% \quad (8)$$

According to the Equation (8), the smaller the E_{ft} is, the more uniform the load distribution is, and the better the algorithm is.

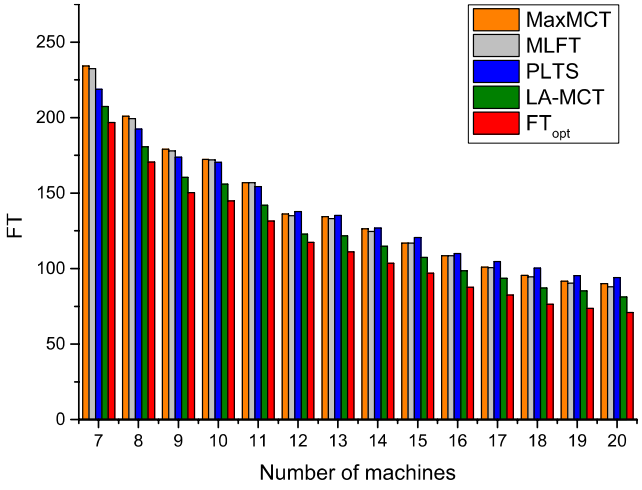
In order to verify the efficiency of the LA-MCT algorithm, we employ Java program to conduct simulation experiments to evaluate different scheduling strategies. The segment complexity ranges from 40 to 100 and the maximum segmentation granularity of each segment ranges from 5 to 20. To rule out chance, the complexity and the maximum segmentation granularity of each segment are randomly generated. The computing capacity of each machine ranging from 1.0 to 5.0 is also randomly generated, and it means that $1.0 \leq p_m \leq 5.0$. All segments are stored 3 replicas on 3 random selected machines. Given a segment which is stored on machine m', will be transcoded on m. If segment is a local task, then $C_{m',m}$ = 0. If m' and m are running on the same rack, then $C_{m',m}$ = 1, and the network bandwidth between them is 1000 Mbps.

Otherwise, $C_{m',m}$ = 3 and the bandwidth is 100 Mbps. The task launching overhead o_m is 5 seconds.

In our experiments, previous studies on video transcoding task scheduling algorithms including Max-MCT, MLFT and PLTS are also implemented as benchmarks. We evaluate the entire finish time of transcoding job by varying the number of machines and



(a) different number of segments



(b) different number of machines

Figure 4.The FT under varying settings

segments, respectively. For each situation, we schedule them 10 times and then pick out the average of the entire finish time.

Fig. 4 shows the FT of video transcoding under varying settings, and Table 1 shows the performance comparisons E_{ft} between the proposed algorithm and other three algorithms. In Fig. 4(a), we use 10 machines to form a heterogenous cluster, and we vary the number of segments from 30 to 150. In Fig. 4(b), the number of segments fixed at 50, by constantly adjusting the number of machines in the cluster to compare the FT of these algorithms. In Table 1, We list the factor E_{ft} of four algorithms when the number of machines is 10 and the number of segments ranges from 30 to 150.

It is observed that the LA-MCT algorithm outperform the others in all cases. Owing to both Max-MCT and MLFT algorithms take advantage of the idea of allocating complex segments to computationally powerful machines, but they neglect the data communication overhead of complex segments, resulting in the algorithm less than ideal. The PLTS algorithm is not as good as the Max-MCT and MLFT algorithm when the ratio between the number of segments and the number of machines is too small, but it is obviously better than the other two algorithms as the ratio increases. The reason is that the PLTS algorithm uses the Min-Min

algorithm to map tasks so that the tasks are concentrated on a higher computation capacity machines while executing a job with a small data amount. The LA-MCT algorithm can determine how many tasks need to be performed locally and how many tasks are executed remotely, which balances the segment transmission time and task transcoding time and then achieves load balancing. In a word, our algorithm is closer to FT_{opt} than other three algorithms.

Table 1. The E_{ft} for each algorithm

Number of segments	MaxMCT	MLFT	PLTS	LA-MCT
30	21.5	21.4	26.4	12.0
40	20.0	18.6	20.0	9.5
50	19.9	18.8	15.1	7.9
60	18.7	17.9	15.2	6.8
70	19.4	18.8	13.0	6.1
80	18.6	17.3	10.5	4.9
90	17.7	17.3	9.6	5.3
100	16.0	16.0	9.7	4.2
110	18.0	17.1	9.5	4.8
120	17.2	16.3	7.9	4.6
130	17.9	17.2	7.8	4.0

7. CONCLUSION

In this paper, we propose a locality-aware task scheduling algorithm LA-MCT for parallelizing video transcoding over heterogenous MapReduce cluster. The algorithm can balance the segment transmission time and task transcoding time. Large number of simulation experiments show that LA-MCT can generate minimum entire finish time compared with the existing algorithms under different scale conditions.

8. ACKNOWLEDGMENTS

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have helped to improve the manuscript. This work was partially supported by the National Key Research and Development Plan of China under Grant No. 2016YFB0200405 and 2012AA01A30101, the National Natural Science Foundation of China with Grant Nos. 61672217, the CERNET Innovation Project under Grant No. NGII20161003, and the China Postdoctoral Science Foundation under Grant No. 2016M592422.

9. REFERENCES

[1] A. Vetro, C. Christopoulos, and Huifang Sun. 2003. Video transcoding architectures and techniques: an overview. IEEE Signal Processing Magazine 20, 2, 18–29.

[2] C. C Wust, L Ste ens, R. J Bril, and W. F. J Verhaegh. 2004. QoS control strategies for high-quality video processing. In Euromicro Conference on Real-Time Systems. 3–12.

[3] D. M. Barbosa, J. P. Kitajima, and W. Weira. 1999. Parallelizing MPEG video encoding using multiprocessors. In Xii Brazilian Symposium on Computer Graphics and Image Processing. 215– 222.

[4] Yen-Kuang Chen, Eric Q. Li, Xiaosong Zhou, and Steven Ge. 2006. Implementation of H.264 encoder and decoder on personal computers. Journal of Visual Communication and Image Representation 17, 2, 509–532. https://doi.org/10.1016/j.jvcir.2005.05.004

[5] Bongsoo Jung and Byeungwoo Jeon. 2008. Adaptive slice-level parallelism for H.264/AVC encoding using pre macroblock mode selection . Journal of Visual Communication and Image Representation 19, 8 (2008), 558–572.

[6] Je rey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. ACM, 107–113.

[7] Zhao Li, Yao Shen, Bin Yao, and Minyi Guo. 2015. OFScheduler: A Dynamic Network Optimizer for MapReduce in Heterogeneous Cluster. International Journal of Parallel Programming 43, 3, 472–488.

[8] Faraz Ahmad, Srimat T. Chakradhar, Anand Raghunathan, and T. N. Vijaykumar. 2012. Tarazu:optimizing MapReduce on heterogeneous clusters. 61–74.

[9] Yaoguang Wang, Weiming Lu, Renjie Lou, and Baogang Wei. 2015. Improving MapReduce Performance with Partial Speculative Execution. Journal of Grid Computing 13, 4, 587–604.

[10] Julio C. S. Anjos, Ivn Carrera, Wagner Kolberg, Andre Luis Tibola, Luciana B. Arantes, and Claudio R. Geyer. 2015. MRA++: Scheduling and data placement on MapReduce for heterogeneous environments. Future Generation Computer Systems 42, 22–35.

[11] Zhenhua Guo and Geo rey Fox. 2012. Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization. In Ieee/acm International Symposium on Cluster, Cloud and Grid Computing. 714–716.

[12] Zixia Huang, Chao Mei, Li Erran Li, and Thomas Woo. 2011. CloudStream: Delivering high-quality streaming videos through a cloud-based SVC proxy. In INFOCOM, 2011 Proceedings IEEE. 201–205.

[13] Myoungjin Kim, Hyeokju Lee, Seungho Han, and Hanku Lee. 2012. Towards E cient Design and Implementation of Hadoop-based Distributed Video Transcoding System in Cloud Computing Environment.

[14] Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Zhi Li Zhang, and Yafei Dai. 2012. Cloud transcoder:bridging the format and resolution gap between internet videos and mobile devices. In International Workshop on Network and Operating System Support for Digital Audio and Video. 33–38.

[15] Rui Cheng, Wenjun Wu, Yihua Lou, and Yongquan Chen. 2014. A Cloud-Based Transcoding Framework for Real-Time Mobile Video Conferencing System. In *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. 236– 245.

[16] He Ma, Beomjoo Seo, and Roger Zimmermann. 2014. Dynamic scheduling on video transcoding for MPEG DASH in the cloud environment. In ACM Multimedia Systems Conference. 283–294.

[17] Tewodors Deneke, Habtegebreil Haile, Sbastien Lafond, and Johan Lilius. 2014. Video transcoding time prediction for proactive load balancing. In IEEE International Conference on Multimedia and Expo. 1–6.

[18] Rui Cheng, Wenjun Wu, Yihua Lou, and Yongquan Chen. 2014. A Cloud-Based Transcoding Framework for Real-Time Mobile Video Conferencing System. In IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. 236– 245.

[19] Feng Lao, Xinggong Zhang, and Zongming Guo. 2012. Parallelizing video transcoding using MapReduce-based cloud computing. In IEEE International Symposium on Circuits and Systems. 2905–2908.

[20] Song Lin, Xinfeng Zhang, Qin Yu, Honggang Qi, and Siwei Ma. 2013. Parallelizing video transcoding with load balancing on cloud computing. In IEEE International Symposium on Circuits and Systems. 2864–2867.

[21] Hui Zhao, Qinghua Zheng, Weizhan Zhang, and Jing Wang. 2016. Prediction-based and Locality-aware Task Scheduling for Parallelizing Video Transcoding over Heterogeneous MapReduce Cluster. IEEE Transactions on Circuits and Systems for Video Technology PP, 99, 1–1.

[22] M. R. Garey, D. S. Johnson, and Ravi Sethi. 1976. The Complexity of Flowshop and Jobshop Scheduling. Mathematics of Operations Research 1, 2, 117–129.

[23] Tracy D. Braunt, Howard Jay Siegel, Noah Beck, Ladislau L. Boloni, Muthucumaru Maheswarans, Albert I. Reuthert, James P. Robertson, Mitchell D. Theys, Bin Yao, and Debra Hensgeno. 2000. A Comparison Study of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Ileterogeneous Distributed Computing Systems, Vol. 61. 810837.

[24] Jun Xin, Chia Wen Lin, and Ming Ting Sun. 2005. Digital Video Transcoding. Proc. IEEE 93, 1, 819–832.