A Prediction-Based Transcoding System for Video Conference in Cloud Computing

Yongquan Chen[1]

Abstract. We design a transcoding system that can provide dynamic transcoding services for various types of video conferencing clients. It can be deployed to many kinds of cloud computing platform and the scheduler can change the setting of any transcoding task to achieve adaptive transcoding requirement form mobile clients. In this paper, we first analyzed the difference between video conference transcoding and on-demand video. We determined that transcoding service for video conference must be real-time and existing scheduling algorithms are not suitable for video conference transcoding. Therefore, we propose a prediction model that can predict how much computing resources that a transcoding task may require. This prediction model is based on the parameters from source video or the transcoding task. Based on the information from resource monitor and prediction model, the scheduling algorithm can make a more appropriate scheduling decision and keep the load of each virtual machine in transcoding system in a good balance.

Keywords: transcoding; video conference; real-time; prediction model; load balance

1 Introduction

With the development of mobile networks and smart devices, mobile clients have become a kind of popular terminal in video conference. But due to the high demands of video quality and low latency in high-definition video conference, to get a better meeting experience, transcoding for mobile devices has become an inevitable solution [1]. Considering about the differences of transcoding between video conference and video on-demand [2], there are three things that we need to notice.
1. Sensitive to latency. Different from video on-demand, high latency in video or voice interaction will lead to communication problem even misunderstood. But for

[1] Yongquan Chen
State Key Laboratory of Software Development Environment, Beihang University Room G612, New Main Building, 37 Xueyuan Road, Haidian District Beijing 100191, China
e-mail: cyqyong@nlsde.buaa.edu.cn

video on demand, they will process the video after it is uploaded instead of a real-time transcoding when the user is viewing.
2. Storage and deliver. In video conference, the multimedia data come from the network and after transcoding the data must be sent to client immediately [3].
3. Dynamic transcoding. In video conference, we also need to implement other transcoding service such as ROI extraction, scale of resolution. On the other hand, adaptive transcoding is an important feature for the mobile client, and therefore, the transcoding service must be able to be changed during its life cycle [4].
These three differences mean that technology for video on-demand transcoding could not apply to video conference transcoding simply. Therefore we present a transcoding system for mobile clients in video conference which can be deployed in a private cloud computing platform in this paper.

2 System Overview

Figure 1 demonstrates the architecture of out transcoding system. Resource Monitor can collect the usage of the virtual machine's computing resource periodically and send it to Load Balance Scheduler which will use this information to schedule the transcoding tasks. Load Balance Scheduler can receive transcoding requests from mobile clients and assign transcoding task to transcoder by sending schedule commands. After the transcoder gets ready for working, the Input Dispatcher will forward the multimedia data which comes from network to the transcoder. And the Output Dispatcher will send the data after transcoding to mobile clients.

The transcoder process will register itself to scheduler first and periodically send a message to show that it is running normally. We have implemented several kinds of transcoding filters as it shown in table 1. And developer can add a new kind transcoding filter easily by implementing the interface we define.

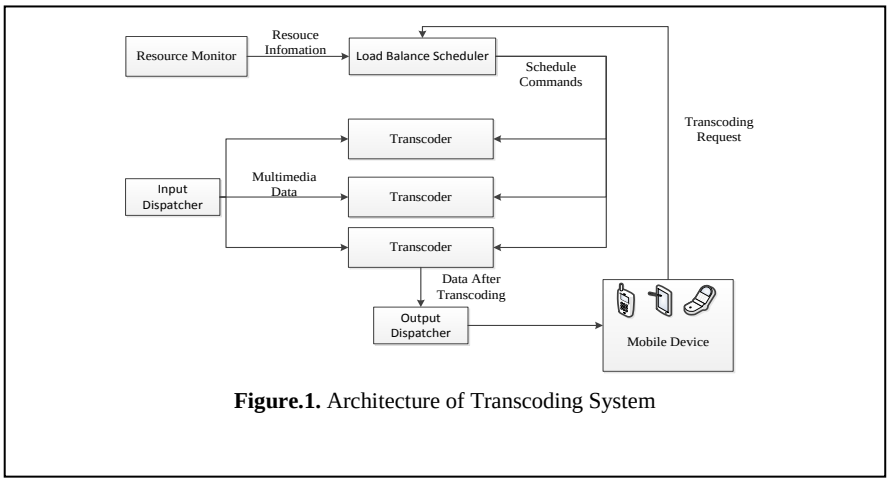


Figure.1. Architecture of Transcoding System

Table 1 transcoding filter in our system

Name	Description
Unpack Transcoder	Unpack data packets in transport protocol such as RTP/RTCP
Pack Transcoder	Pack compressed data into transport protocol such as RTP/RTCP
Decoder Transcoder	Decode compressed data to raw data
Encoder Transcoder	Encode raw multimedia data
Scale Transcoder	Change the video resolution
ROI Extract Transcoder	Extract Region of Interest as a new video stream

By resetting the parameters of each transcoding filter in transcoder dynamically, we can implement an adaptive transcoder.

3 Prediction-based Load Balancing

In this section we design a prediction-based load balancing model which can improve the efficiency of the cloud computing platform which our transcoding system will be deployed on and ensure a low latency.

3.1 Prediction Model of Load Balancing

Prediction model could predict the resource usage that a new transcoding task may consume. The CPU usage percent is the most important computing resource to judge whether a new transcoding task can be add to a virtual machine. Table 2 defines all the parameters used to predict the CPU usage. In our video conference system, the video resolution criteria which are usually used are 1920x1080, 1280x720, 640x480, 320x240 and 176x144, the frame rate are 30fps, 25fps, 20fps, 15fps and 10fps. Firstly we collect the average CPU usage percent of each kind transcoder and the parameters that listed in table 2, and then we build a linear prediction formula for this transcoder by linear regression [6].

Table 2 Parameters in Prediction Model

Parameters	Definition
width	The width of raw frame in pixel
height	The height of raw frame in pixel
fps	Frame rate per second of a video stream
framesize	The size of a raw video frame in byte, *framesize = (width * height * 1.5)/1024*
streamsize	The size of input or output stream to a transcoder in byte.

For the decoder and encoder of H.264, the CPU usage is primarily affected by the size of input stream [5]. The input of decoder transcoder is compressed video data and scheduler can get the *streamsize* from the video sender before prediction. In Figure 2(a), the x-axis is the *streamsize* of which the unit is kilobytes per second

and y-axis is the CPU usage percent. As it is shown in Figure 2(a), the CPU usage percent value varies linearly with the *streamsize*, and in different resolution, the line has different slope, while when the resolution is smaller than 640x480, the prediction result can be consider as a constant value. So we can define the prediction formula of decoder as below:

$$P_{decoder}= \begin{cases} a* \text{input } streamsize + b & width>320\&\&height>240 \\ 0.5 \end{cases} \quad (1)$$

Based on data from Resource Monitor, We can get the value of a and b through a linear regression and the result is shown in table 3.We can see that the R value of linear regression of decoder is as high as 0.99, therefore we have obtained a good linear model to describe the relationship between the CPU usage percent and input bitrate. If the resolution of the input video is below 640x480, such as 320x240 and 176 x144, the CPU usage average will too small to be predicted. The experiment data show that the average of the CPU will not exceed 0.5%.

The CPU usage of encoder is mostly decided by the *framesize* and *fps* of the input video as it is shown in Figure 2(b), the x-axis is the input *streamsize* which equal with *fps***framesize*, its units is also KB/S, the y-axis is the CPU usage percent. Similar to decoder, we can use linear regression to construct a linear prediction formula for encoder and the linear regression result is listed in table 4.

Table 3 Linear Regression of Decoder

Resolution	**a**	**b**	**R**
1920x1080	0.016699	0.22206	0.994246
1280x720	0.012241	0.188587	0.991533
640x480	0.011794	-0.23669	0.982355

The same as decoder, the CPU usage percent of encoder can be regarded as a constant. Therefore the prediction formula will be:

$$P_{encoder}= \begin{cases} a_1 * fps * framesize + b & width>320\&\&height>240 \\ 0.5 \end{cases} \quad (2)$$

The prediction of the scale transcoding is more complicated than decoder and encoder. There are three factors that can impact the CPU usage of the scale transcoder which are input *framesize*, output *framesize* and *fps*. Figure 2(c) shows the relationship of CPU usage and fps in different scale resolution pair, in which "1080to720" means the scale pair is a 1920x1080 video frame to a 1280x720 one, x-axis is the *fps* and y-axis is the CPU usage percent. Figure 2(d) shows the increase trend of CPU usage percent with the output *framesize*, the x-axis is the bytes in output frame of which the unit is KB/S. Now we can define a prediction formula for scale as below:

$$P_{scale}= a_1 * \text{input } framesize + a_2 * \text{fps} + a_3 * \text{output } framesize + b \quad (3)$$

The linear regression result is in table 4, the R value is 0.9654 and that means this formula is still accurate enough. We do not use *streamsize* to predict because that the R value is only 0.72.

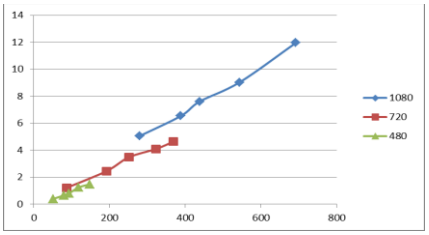
Table 4 Linear Regression of Encoder

Resolution	**a**	**b**	**R**
1920x1080	0.000196	1.48	0.97205
1280x720	0.000179	0.8745	0.990621
640x480	0.000279	-0.695	0.991605

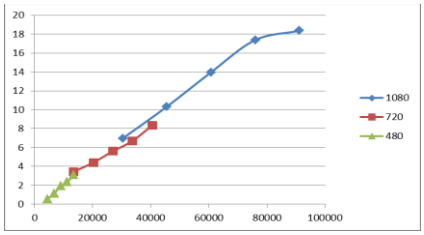
Table 5 Linear Regression of Scale

$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	**b**	**R**
0.000389	0.10866	0.00172	-1.87619	0.9654

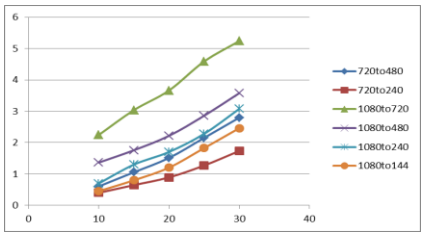
In the prediction of ROI transcoder, the input video *framesize* has little impact on the CPU usage. Figure 2(e) and Figure 2(f) illustrate the influence of *fps* and output *framesize* on the CPU usage. The linear regression result is listed in table 6. We can see that the R value is more than 0.98 and that means we have an accurate result.



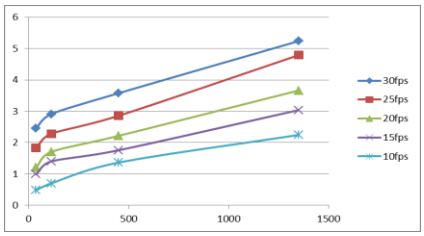
(a) Decoder and instreamsize.



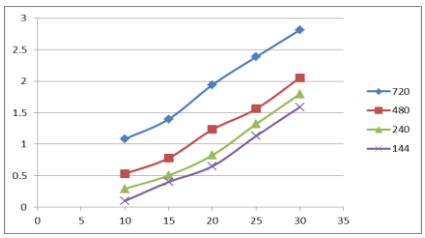
(b) Encoder and instreamsize.



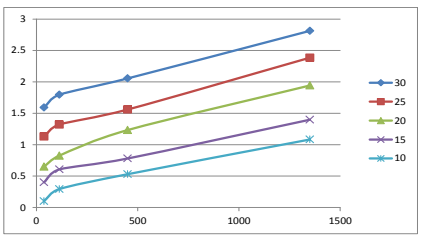
(c) Scale and fps.



(d) Scale and framesize.



(e) ROI and fps



(f) ROI and ROI size

Figure.2. CPU usage of each kind of transcoder and its prediction parameters

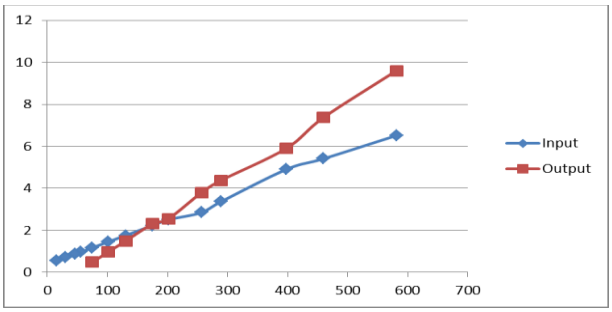


Figure.3. the CPU usage of I/O threads

Table 6 Linear Regression of ROI

a_1	a_2	b	R
0.079	0.000836	-0.768796	0.983859

The I/O threads of the transcoding process will also consume some CPU resource. Figure 3 shows the relationship between CPU usage and the I/O stream size, the x-axis is the input or output stream size and the unit is KB/S, the y-axis is the CPU usage. And table 7 is the linear regression result for I/O threads.

$$P_{I/o} = a * streamsize + b \quad (5)$$

Table 7 Linear Regression of I/O

	a	b	R
Input	0.010568	0.37668	0.984454
Output	0.01074	0.215219	0.990321

With the prediction formula of each kind of transcoder, we can predict the CPU consume of a new transcoding task by formula. And the total CPU usage of the transcoding process will be predicted by formula. Based on the prediction formula above, we can predict the total transcoding process with a linear equation as formula (6).

$$P_{totalCPU} = CPU_0 + \sum P(i) \quad (6)$$

CPU_0 is current CPU usage percent of transcoding service, and *P(i)* is the prediction CPU usage of the transcoding operation in new transcoding task.

Load balancing algorithm can predict the CPU usage of a new transcoding task, and with the resource state data from resource monitor, the algorithm can know whether a virtual machine has enough CPU resource to hold the transcoding task. In the experimental section of this paper, we will evaluate the prediction model.

3.2 Design of the Load Balance Algorithm

Similar to knapsack algorithm, our load balancing algorithm will select a virtual machine which has the most computing resource and make sure no virtual machine will have too heavy load. The load balancing algorithm is described as follows:

Step 1: When there is a transcoding request from mobile client, firstly, search in existing transcoding tasks and check that whether there is a transcoding task for the same video has a similar output and if it is, there is no need to add new task.

Step 2: If the transcoding request is a parameter update for an existing transcoding task, prediction model will predict the computing resource usage after the update. If the resource usage is too much, scheduler will move this task to a new virtual machine which has enough resource to hold it.

Step 3: If load balancing have to select a virtual machine for current transcoding request, scheduler will sort virtual machine list according to the descending order by available computing resources. Then the prediction model will predict resource usage after the new transcoding task is added on the first machine. If it has enough resource, the scheduling is over.

Step 4: If there is no fit virtual machine, the scheduler will migrate part of the tasks on the currently best fit machine to other machines until there has enough resource for the new task.

Step 5: there is a threshold that when the difference of CPU usage between the virtual machine which has the heaviest load and the virtual machine that has the lightest load exceeds the threshold, the scheduler will also migrate part of the tasks of the heaviest virtual machine to others.

The experiment and analysis section will show the effect of this load balance algorithm.

4 Experiment and Evaluation

This section is experiments and evaluation, and we will describe the settings of the cloud computing platform.

4.1 Experimental Settings

We build a small private cloud computing platform by Microsoft Hyper-V [7], and the physical machine' CPU is Intel(R) Core(TM) i5-750 @2.67GHz. The operating system on physical machine is windows 2008R2, the operating system of the virtual machine is windows 2003R2 and the virtual CPU is the same with the physical CPU. All the video streams before transcoding are recorded in real video conferences with *MVision* system which is developed by our laboratory.

4.2 Evaluation of Prediction Model

Limited by the length of this paper, we only use several transcoding tasks to verify the accuracy of prediction model. Table 9 lists the parameters of each transcoding tasks with their description. Firstly, we only run one task each time and get the real CPU usage of the task from resource monitor. Table 10 is the comparison of the prediction CPU usage of the task and real one.

Table 9 Transcoding Tasks

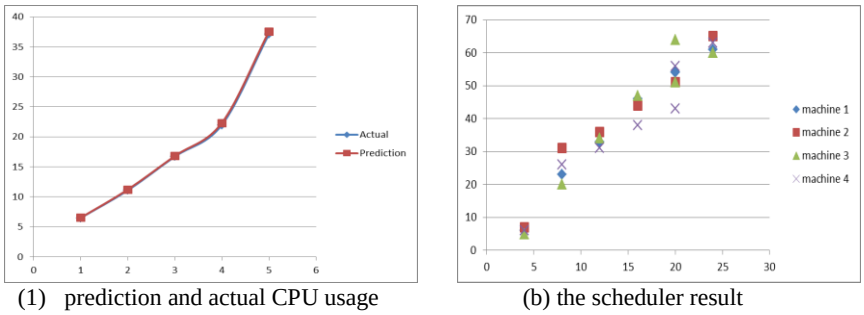
Index	Parameters	Description
1	Input: width=1920 height=1080 fps=10 Bitrate=59.36 KB/S Output: width=640 height=480 fps=10 Bitrate=19.38KB/S	Scale a 1080P input video to 480P and encode it with a bitrate 19.38KB/S
2	Input: with=1280 height=720 fps =15 Bitrate=75.22KB/S Output: width=320 height=240 fps =15 Bitrate=12.74KB/S	Extract a 320x240 ROI from input video and encode it with a bitrate 12.74KB/S
3	Input: width=640 height=480 fps=20 Bitrate=80.98KB/S Output: width=640 height=480 fps=20 Bitrate=26.02KB/S	Decode-Encode a 640x480 video and reduce the bitrate to
4	Input: width=640 height=480 fps=25 Bitrate=152.95KB/S Output: width=176 height =144 fps25 Bitrate=18.16KB/S	Scale a 480P input video to 144P and encode it with a bitrate 18.16KB/S
5	Input: width=1920 height=1080 fps =30 Bitrate=339.84KB/S Output: width=320 height=240 fps=20 Bitrate=14.5KB/S	Scale a 1080P input video to 240P and encode it with a bitrate KB/S and reduce the fps from30 to 20

Table 10 Comparison of the Prediction Result

Index	Prediction CPU (%)	Real CPU (%)	D-value
1	6.5	6.3	0.2
2	4.7	4.4	0.3
3	5.7	5.4	0.3
4	5.6	5.1	0.5
5	15.5	15.1	0.4

From Table 10 we can see that the prediction model can see that the prediction model can accurately predict the CPU usage of each transcoding task, and the maximum error does not exceed 0.5%.

We will add these five transcoding tasks to the same virtual machine one by one to verify the prediction formula (6). Figure 4(a) is the experiment result, the red line is the prediction result and the blue line is real CPU usage percent, and the x-axis is the count of tasks on the virtual machine, the y-axis is the CPU usage percent. We can see that these two lines are almost coincident, and actually the red line is a



(1) prediction and actual CPU usage (b) the scheduler result

Figure.4. Experiment Result

little higher than the blue one, that means the prediction result is always higher than the real one, but it won't too high to cause waste of the resources.

4.3 Evaluation of Load Balancing

The experiments above have demonstrated the validity of the prediction model, and then we will evaluate the effect of the load balancing algorithm. In this experiment, dozens of transcoding tasks are scheduled on four virtual machines and the threshold value is 20. Figure 4(b) show the scheduling results, the x-axis is the task count and y-axis is the CPU usage percent of virtual machine. We can see that the load of each virtual machine increases evenly and we notice that when there are 20 tasks, the CPU usage of machine 3 is 64% at first which is 21 higher than machine 4 who's CPU usage is only 43%. Then the scheduler migrate a transcoding task on machine 3 which consumes about 13% CPU resource to machine 4 to make the load balance.

5 Conclusions

In this study, we implement a dynamic transcoding system which has a prediction-based load balancing scheduler. This transcoding system can provide real-time transcoding services for mobile clients in video conference, and the prediction-based algorithm can effectively take advantage of the computing resource in cloud computing platform, and ensure that the transcoding load of each virtual machine grows balanced.

Acknowledgments

This work was supported by the State Key Laboratory of Software Development Environment Funding No.SKLSDE-2009ZX-12.

References

1. Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Zhi-Li Zhang, Yafei Dai, Cloud Transcoder : Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices[C], NOSSDAV'12,June 7-8. 2012.
2. Lao Feng, Xinggong Zhang, Zongming Guo. Parallelizing Video Transcoding Using Map-Reduce-Based Cloud Computing[C], IEEE International Symposium on Circuits and Systems, May 20- 30. 2012.
3. Adriana GARCIA, Hari KALVA, Cloud Transcoding for Mobile Video Content Delivery[C], ICCE, 2011
4. Bo Shen, Wai-Tian Tan, Frederic Huve. Dynamic Video Transcoding in Mobile Environments[C], IEEE MULTIMEDIA, 2008.
5. A.D. Bavier, A.B. Montz, and L.L. Peterson, "Predicting Mpeg Execution Times," Proc. Int'l Conf. Measurements and Modeling of Computer Systems (SIGMETRICS), pp. 131-140, 1998.
6. Jiani Guo, Laxmi Narayan Bhuyan. Load Balancing in a Cluster-Based Web Server for Multimedia Applications. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 17, NO. 11, NOVEMBER 2006.
7. Microsoft. Server and Cloud Platform. http://www.microsoft.com/en-us/server-cloud/hyper-v-server/default.aspx