Layered Stream Scheduling Scheme for Cluster Video Servers

Limin Zhu
Information Management Center, College of Management
Huazhong University of Science and Technology

Abstract: - People have put more emphasis on stream-scheduling policies to build high performance video servers. But the most important aim of these traditional stream-scheduling schemes is to improve the resource utilization, such as memory, network bandwidth and CPU, to maximize the number of concurrent streams. These schemes did not consider the QoS needs of video streaming or described the QoS parameter weakly, so the media streams average qualities have no guarantees when the number of media streams is very large. Traditional schemes put little consideration on cluster video servers. In this paper, we will propose a new two-layer stream-scheduling scheme with QoS control for cluster video servers. We first proposed new definitions for video streams with the consideration of QoS and then described the scheme of how to schedule video streams from two layers. The target of this scheme is to improve the average QoS of all streaming tasks and to obtain large number of scheduled video streams.

Key-Words: - Stream; Scheduling; Multimedia; Cluster; Qos; VOD

1. Introduction and Related Works

Design towards process to structure is a trend of systems engineering. The characteristics of continuous media are different from traditional text-based or image-based files. Typically, video-on-demand technology imposes high bandwidth and real-time requirements. This technology should be capable of delivering concurrent video and audio streams to numerous client users. There are several challenges for designing the servers, such as the high storage-capacity and throughput in the video server and the high bandwidth in the network to deliver large number of video streams. In order to solve these problems, the video server must be a high performance computing system. While most systems are built on a single high performance computer, more and more people tend to run this service on a cluster system composed of off-the-shelf commodities for their cost-effectiveness and fault-tolerant capability, and this area has attracted increasing attention of the researchers.

Researchers have many achievements in media stream-scheduling field [1], but few people have put their emphasis on stream-scheduling policy for cluster video servers, which are featured with distributed stream control [2]. Another reason is that traditional researchers do not consider the QoS of streaming tasks and the features of media streaming when they designed the stream-scheduling scheme.

Traditional schemes do not consider these characteristics and they are very effective to non-media tasks. For example, EDF scheme and time-sharing scheme [4] are very adaptive for real-time tasks and some non-real-time tasks. We should first take the characteristics of media into account. Cluster video servers distribute all tasks onto many data servers or processing nodes. We should also design a stream-scheduling scheme to satisfy the distributed architecture.

In this paper, we propose a new-layered stream scheme for cluster video servers. With the help of the new stream-scheduling scheme, the cluster video servers can accept more streaming tasks and schedule them with good average QoS compared with traditional stream-scheduling schemes, such as EDF scheme and time-sharing scheme.

The paper is organized as follows. In section 2, we discuss the architecture of cluster video server. Then we present our stream-scheduling scheme with QoS control in Section 3. In Section 4, we give some performance analysis through simulations. Finally, section 5 closes with conclusions.

2. Typical Video Server Architecture

Architecture of the video server is illustrated in Figure 1. The video server system consists of three major components: a virtual server, several control servers and some data servers. In this system, all media files are cut into many clips and all clips should be stored on all distributed data servers.

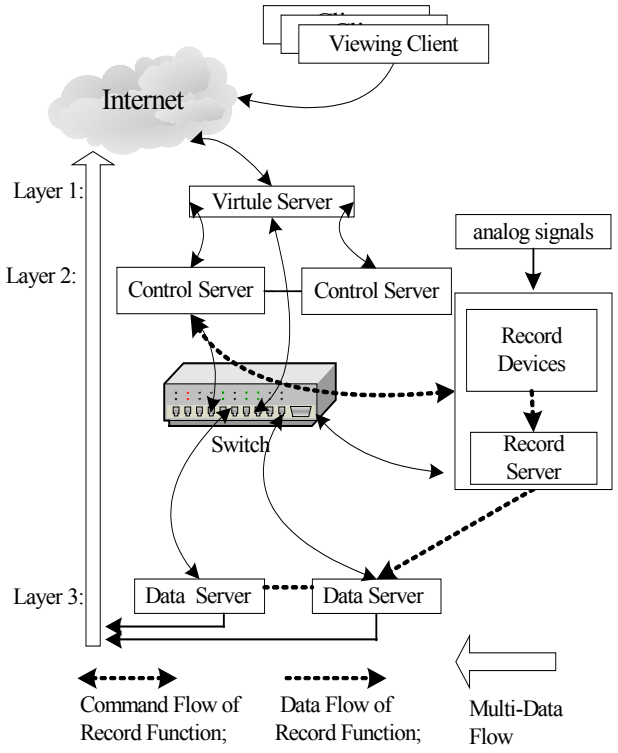

Figure 1 Scalable Cluster Video Server

A control server has several tasks. One is to make admission control for requests from the clients. Second task is to parse requests from clients. According to the RTSP request types, the control server executes the "describe", "setup". When it receives "play" command, it translates the "play" command into a sub-"play" task list according to clip files information of the requested movie. The last task of a control server is to notify corresponding data servers to send media data to clients directly according to sub-task lists.

A storage node works in a semi-independent way. It performs orders from control servers, and decides its own sub-stream scheduling scheme to manage media data and sends them out.

From above, we find out that one integrated media stream should get a permission from a control server and be parsed into several sub-streams according to its clips information. All sub-streams are scheduled by corresponding data servers.

3. Layered Stream Scheduling Scheme

An integrated stream-scheduling scheme for cluster video server has two layers, showed as Fig.2. The first layer is located on control servers and its main mission is to schedule streams with the coarse granularity. In this layer, the system selects permitted streams, creates all sub-tasks and transmits the sub-tasks to the second layer with some specified parameters. The second layer is located on data servers. Considering the QoS necessaries, this layer will adapt a new scheme to schedule the sub-streams and send media data to clients directly.

In the following of the section, we first describe the two stream-scheduling layers and the QoS parameters in detail.

3.1. Layer 1: Stream-Scheduling

There are many requests from clients to the control servers. When they pass admission control check-up, they become new tasks, waiting for being scheduled. One task is divided into several sub-tasks according to its clips information. We can define the task I as below:

$$Task(I) = \begin{pmatrix} TLength(I), ClipNum(I), ClipTable(I), \\ FirstStart(I), Base\,\Pr iority(I) \end{pmatrix} \quad (1)$$

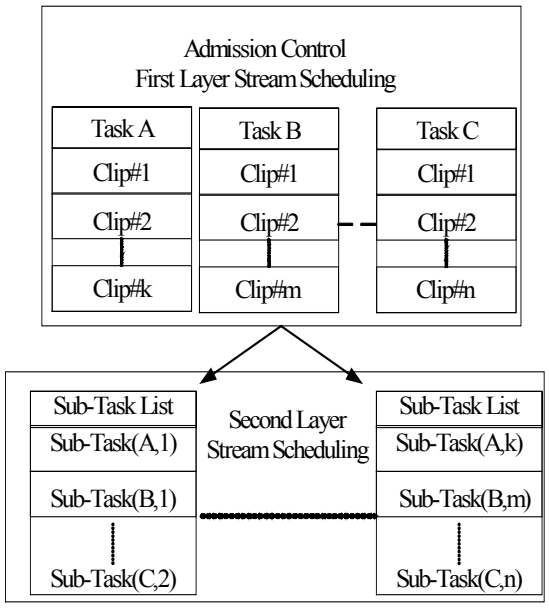

Figure 2 Double-Layer Stream Scheduling Graph

In formula (1), *TLength(I)* defines the time length of the task *I*, representing the corresponding playing time length in task *I*. *ClipNum(I)* defines the number of the clips, belonging to the requested media file in task *I*. *FirstStart(I)* defines the first start time of the task scheduled. *BasePriority(I)* defines the basic priority of media file in task *I*. In this system, we give high priorities to media files with hot spot. With this scheme, hot media files get high operation priorities while being scheduled. In our system, we define three level priorities: high, middle and low with values 10, 20, 30, respectively. *MaxPriority* defines the maximal priority with value 100. *ClipTable(I)* is a very important parameters in the task definitions, defined below:

$$ClipTable(I)=\{j=1,ClipNum(I) \mid (SubTask(I, j), SubTaskAddr(I, j)) \} \quad (2)$$

Where *j* represents the sequence number of the media file in task *I* with values from 1 to *ClipNum(I)*. (*SubTask(I, j), SubTaskAddr(I, j)*) defines a clip description item in clip tables. In this component, *SubTask(I, j)* represents a sub-task *j* in task *I*. *SubTaskAddr(I, j)* defines the data server which should receive and schedule this sub-task.

3.2. Layer 2: Stream-Scheduling

For task I, the first scheduling layer dispatches all sub-tasks of this task to predefined data servers according to *SubTaskAddr(I, j)*. For the second scheduling layer, data servers start up threads to deal with the received sub-tasks.

In continuous media applications, the operations on multimedia objects can be described as periodic tasks. Every periodic task has its own start time, execution time and deadline. Multimedia files are made up of large numbers of frames, such as P frame, B frame and I frame. Most of media files are compressed with constant frame rates. Media files with variable frame rates also have stable average frame rates. For example, in a media file with average frame rate 25, we use 40ms to deal with a frame. The definition of *SubTask(I, j)* is listed below:

$$SubTask(I, j)=\begin{pmatrix} Start(I, j), MaxT(I, j), \\ Cycle(I, j), Num(I, j), \\ MF(I, j), CF(I, j), \Pr iority(I, j) \end{pmatrix} \quad (3)$$

There are seven parameters in this definition *SubTask(I, j)*. *Start(I, j)* defines the time when the sub-task j in task I is scheduled. *MaxT(I, j)* represents the execution time length of the sub-task *j* in task *I*. *Cycle(I, j)* defines the time length of internal cycle.

In the second scheduling layer, we divide execution of a sub-task into many cycles. A sub-task is also partitioned into many routines with maximal execution time *Cycle(I, j)*. A routine represents a frame in multimedia file. *Num(I, j)* defines the number of routines in this *SubTask(I, j)*. *MF(I, j)* represents the maximal number of routines failed. *CF(I, j)* represents the maximal number of routines failed continuously. *Priority(I, j)* defines the priority of a sub-task. The sub-task of task *I SubTask(I,0)* has the same priority of *BasePriority(I)* of *Task(I)*. The following sub-tasks have dynamic priorities according to the previous sub-task execution results.

From the definition in Eq.(3), we have the following conclusions:

First, the deadline of the sub-task *j* in task *I* can be calculated as:

$$Deadline(I, j)= Num(I, j)*Cycle(I, j)+Start(I, j) \quad (4)$$

Second, we can define every routine first start point and deadline below:

$$RoutineStart(I, j, m)=Start(I, j)+ (m-1)*Cycle(I, j) \quad (5)$$

$$RoutineDeadline(I ,j, m)= Start(I, j)+ m*Cycle(I, j) \quad (6)$$

If one routine starts later than *RoutineStart(I, j, m)* or finishes later than *RoutineDeadline(I, j, m)*, we can say that this routine has failed.

At last, if we define the number of all failed routines in task *j* of task *I* as *FailedNum(I, j)* and the number of continuous failed routines as *ContinousFailedNum(I,j)*, we have following claim:

$$\text{If } FailedNum(I, j) > MF(I, j) \text{ or } ContinousFailedNum(I, j) > CF(I, j)$$

$$\text{Then } SubTask(I, j) \ Failed \quad (7)$$

In our stream-scheduling scheme, we assume that every sub-task of one task would succeed in operations. One sub-task succeeds completely means that all routines of this sub-task succeed. But in real environments, some media data can not be sent out

normally, such as resources of servers, including bandwidth, CPU and disks, are very busy, and there are some media frames lost. Since we cannot reach this target, we assume every sub-task can succeed when we take into account the requirements of QoS. Therefore, we have:

$$FailedNum(I,j) \le MF(I,j) \quad \text{and} \quad ContinousFailedNum(I,j) \le CF(I,j) \tag{8}$$

From above, QoS changes only in a little scope. We cannot make sure that there are no media lost and we can ensure that it is not very obvious for streaming qualities to lose media data in a controlled range.

3.3. Sub-Task Scheduling

The transmitting probability *P(I, j)* of a sub-task is given in Eq.(9) below. Data servers choose a sub-task whose probability is the largest from the scheduling list.

The priority of every sub-task is dynamic and depends on its former sub-task transmitting probability. $P(I,j)'$ represents the average probability of all sub-tasks, waiting in the scheduling list on one data server at the time t.

$$P(I,j) = a1 * \frac{FailedNum(I,j)}{MF(I,j)} + a2 * \frac{ContinousFailedNum(I,j)}{CF(I,j)} + a3 * \frac{Priority(I,j)}{Max\ Priority} \tag{9}$$

$$a1 + a2 + a3 = 1 \tag{10}$$

$$Priority(I,0) = Base\ Priority(I) \tag{11}$$

$$Priority(I,j+1) = Priority(I,j) + \left(P(I,j) - P(I,j)'\right) * Priority(I,j) \tag{12}$$

a1, a2, and a3 are coefficients and their values should be adjusted during experiments. In Eq.(9), we consider the influence of QoS, such as all failed routines number and continuous failed routines number when the data servers make decisions about how to select a sub-task to schedule.

When two sub-tasks have the same transmitting probabilities, we compare their routines priorities. In general, I frame have a high priority, B frame middle and P frame low. If current routines of sub-tasks all have the same frames, we choose one sub-task using the traditional EDF arithmetic.

4. Performance Evaluation

4.1. Simulation Model

When system load on one data server is very low, there are almost no failed routines and failed sub-tasks, and there are no need to consider QoS. At this time, our scheme has the same effect as traditional scheme, such as EDF and time-sharing scheme. But when the video server provides massive concurrent stream services and workload of every data server is very high, our scheme is more effective than other schemes.

Without loss of generality, we make some assumptions and parameterize the simulation model. First, if any sub-task *SubTask(I, j)* fails, its parent *Task(I)* can be regarded as failed task and the other sub-tasks will be cut off and the stream request will be discarded at last. We define the ratio of failed tasks amount to the number of all tasks *FailedRatio*. At the same time, we also have the ratio of succeed tasks amount to the number of all tasks *SucceedRatio.*

Second, the admission control procedure adopts a network bandwidth threshold-based policy. The threshold for one autonomous storage node is 90Mbps, $A(T_{new})$ <= 90Mbps.

Third, we have 100 movies compressed in MPEG-1 and every movie has 2 clips, which are distributed onto two different data servers. Every clip file with 30fps has the average bandwidth 1.5Mbps and its time length is about 600 seconds. The movie selection pattern is conform to Zipf distribution (α = 1) [8] and 100 clients requests arrive as a steady Poisson stream with the arrival rate λ=0.25.

We also assume that EDF scheme and time-sharing scheme obey the same rules: *MF(I,j)*=500 and *CF(I,j)*=10. Because very movie is about 600 seconds and has about 18000 frames or routines, the maximal ratio of failed routines to total routines is about 2.7%. That is, when *FailedNum* >= 500 or ContinousFailedNum >=10, EDF scheme, time-sharing scheme or our scheme all regard that current sub-task fails. Here, a1=0.5, a2=0.4, a3=0.1 and the time slot in time-sharing scheme is 1 second.

At last, in the following cluster simulation environments, there are 2 storage nodes and one control server node with hardware configurations PIII800, SDRAM 256M.

4.2. Simulation Results

We first reveal relationship between the timeline and FailedRatio on one data server when using our scheme, EDF scheme, and time-sharing scheme, shown in Figure 3 and Figure 4. In this environment, we find out that maximal number of admitted streams is no more than 60 because of the admission control policy and average bandwidth of every MPEG-1 stream. From these two figures, we find out that double-layer stream control scheme with QoS can schedule more streams than the traditional EDF scheme and time-sharing scheme.

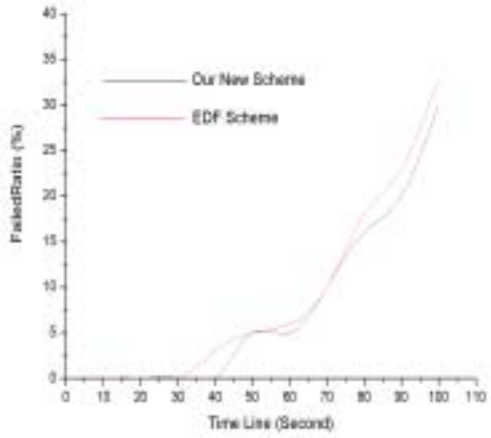

Figure 3 Failed Ratios between Our Scheme and EDF Scheme

According to our simulation environment, there is a feature that system workload is increasing with the elapsing of the time before all clients request have arrived, which has the reason that the clients requests are increasing according to Poisson stream policy. Because the data server workload is harder, it cannot schedule all sub-tasks on this server. Because we have considered the streams QoS, double-layer stream control scheme with QoS has better average QoS than EDF scheme and time-sharing scheme.

We then study the relationship between total failed routines (frames) and timeline in our scheme and traditional schemes, shown in Figure 5 and Figure 6. We find out that failed frames in traditional schemes are more than those in our scheme. There are several reasons to have the results. One is that our new scheme has taken account of the priorities of different frames and different sub-tasks, the other reason is that we stop the decreasing trend of some sub-tasks QoS and make a reasonable adjustment.

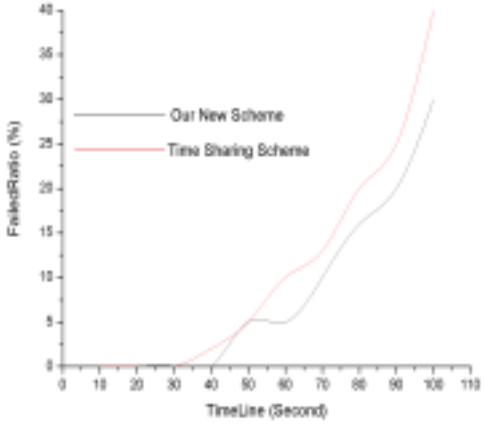

Figure 4 Failed Ratios between Our Scheme and Time-Sharing Scheme

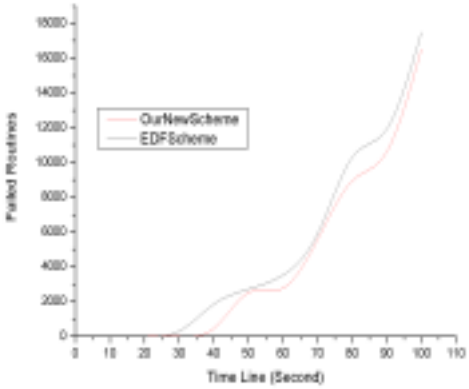

Figure 5 Failed Routines between Our Scheme and EDF Scheme

5. Conclusions

This paper presents an application with layered streaming control scheme to cluster video servers. It is testified by simulation that this scheme improves the average QoS for all streaming tasks on one video server compared with traditional ones. This scheme will also be generalize to be a pattern of random optimization of multi-layered time series in the fields of soft science on the base of systems theory, which will be applied to managements engineering and evaluation engineering such as evaluation of enterprise technology innovation ability etc.

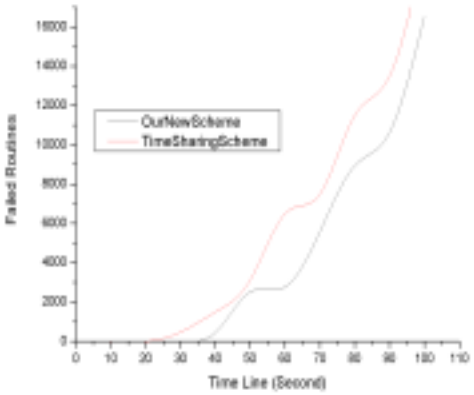

Figure 6 Failed Routines between Our Scheme and Time-Sharing Scheme

6. References

[1]. C.-W. Hsueh, K.-J. Lin, and N. Fan, "Distributed pinwheel scheduling with end-to-end timing constraints", Proc IEEE Real-Time System Symposium, Pisa,1995, pp.172-181.

[2]. L. Amini, J. Lepre, and M. Kienzle, "Distributed stream control for self-managing media processing graphs", Proceedings of the seventh ACM international conference on Multimedia (Part 2), October 1999.

[3]. Tin- Yu Wu, Kun-Chang Chen, Han-Chieh Chao and Tak-Goa Tsuei, "IP Home Network Multimedia Application over IEEE 1394", *4th WSEAS Internantional Conference on Information Science, Communication and Applications (ISA 2004)*, 2004.

[4]. Eung-Nam Ko, "Performance Analysis of Error Detection System for Distributed Multimedia Environment", *4th WSEAS Internantional Conference on Information Science, Communication and Applications (ISA 2004)*, 2004.

[5]. L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping", IEEE/ACM Trans. Networking, 4(4): 482-501, August. 1996.

[6]. J. Y. B. Lee and P. C. Wong, "A Server Array Approach for Video-on-Demand Service on Local Area Networks", Proc. IEEE INFOCOM '96, Mar. 1996.

[7]. E. W. Knightly and N. B. Shroff. "Admission Control for Statistical QoS: Theory and Practice", IEEE Network, March 1999.

[8]. R. L. Axtell, "Zipf Distribution of U.S. Firm Sizes", Science. Sept. 7, 2001, Vol. 293.