MANTECH Publications

Journal of Digital Image Processing and Computer Vision
Volume 2 Issue 1

Video Streaming Via Smart Cache

Abhinand R K[1], Afsal A Azeez[2], Nikhin Raj K K[3], Eldo P Elias[4], Jisha P Abraham[5], Joby George[6]

Professor[4,5,6]

Department of Computer Science and Engineering

Mar Athanasius College of Engineering, Kothamangalam, Kerala, India

Corresponding Authors: *abhinandrkvatakara@gmail.com[1], afsalaazeez@gmail.com[2], nikhinrajkk@gmail.com[3], eldope@gmail.com[4], jishaanil@gmail.com[5], jobygeo@hotmail.com[6]*

Abstract

Video streaming has become the leading contributor to the explosive growth of Internet traffic. As a key system parameter of video apps, cache threshold plays an important role in regulating the downloading behavior of a system, and directly affects the efficacy of video streaming. In this paper, we first conduct a series of well designed experiments to understand the mechanism of cache management in the OS and investigate the impact of cache threshold on the performance of video streaming. The experiments show that the current static configuration of cache thresholds in the OS cannot well balance the tradeoff between cost incurred by unconsumed content and user quality of experience. To achieve cost-effective video streaming, this paper further proposes a control-theoretic cache management algorithm called smart cache with adaptive thresholding (SCAT), which can intelligently tune cache thresholds to satisfy user preferences.

The complexity of cache management and optimization can be decreased extensively by the SCAT strategy, facilitating its easy integration with the OS kernel codes. Finally, we implement and evaluate our proposed scheme on the real platform and the experimental results verify that our proposed scheme achieves significant gain over other alternative approaches. Compared to the default scheme, SCAT achieves over40% reduction of unconsumed content cost.

Keywords: *Video streaming, Multimedia broadcasting, Cache management, Cache threshold, Control theory.*

INTRODUCTION

In recent years, the prevalence of devices leads to the rapid growth of the mobile Internet traffic. Cisco reported in [1] that the global mobile data traffic has increased 1.5Exabytes per month, nearly 81 percents higher than that in 2012 [2].The report also shows that 53% of the network traffic is generated by video, and it is expected that over two-thirds of the world's mobile data traffic will be video in 2018.In reality, the quality of mobile video streaming is highly impacted by user mobility and the fluctuation of wireless channels. To mitigate the effect of stochastic channel fluctuation, a commonly adopted approach is to cache a certain amount of video data before playback.

For cache management, two cache thresholds are widely used to regulate the prefetching behavior of a client, namely, high threshold and low threshold. Normally, the high threshold is set as the maximum size of memory space allocated for video cache, and the low threshold is set as the minimum amount of fresh content in the cache to avoid playback freezing. In spite that video caching can help improve user experience, it is at the cost of downloading extra data bytes before the actual video playback.

II. THEORY

Cache thresholds are of great significance to the performance of video streaming. In this section, we conduct a series of measurement studies to understand the mechanism of cache management in the Linux OS and examine the impacts of different cache threshold settings on mobile video streaming.

A. Measurement Methodology

In our measurements, to understand their impacts on video streaming, we adjust the setting of high and low thresholds and evaluate the corresponding quality measures, such as freezing duration, the amount of unconsumed content, and so on.

In the media framework, the underlying media player is called when playing a local or remote video file. Its logs contain all the

information of video playback, including starting time, pause time, freezing duration, etc. In the default settings, video logs generated by the media framework are debug-level system files, which cannot be directly accessed by upper-level applications. To access the log files, we raise the priority level. We write a simple monitor program to automate the task of log collection on the linux platform. To keep track of cache status, we insert extra codes into the media framework. The generated cache logs can be retrieved using the system tools.

B. Evolution of the Cache State

In the following section, we use our testbed to investigate the evolution of cache state, and study the impacts of different configurations of cache thresholds. We first investigate the dynamics of cached contents for video streaming. We define fresh content as the amount of video data downloaded but not viewed in the cache, and stale content as the amount of video content in the cache that has been viewed.

We find that, the client stops fetching more content when the total amount of cached content reaches the high threshold. Once the amount of fresh content drops below the low threshold, the fetching process is started again. In the meanwhile, most of the stale content is cleaned out the cache. To better understand the evolution of cache state, we examine the source code of the media framework. We find that, when the media framework is called to handle a video streaming request, its data source provider follows the FSM (Finite State Machine) shown in Fig. 1. The FSM contains five states, namely, Initial, Fetching, Idle, Keep-Alive and Final.

Initially, when the video playback is started by the viewer, the data source provider enters into the Fetching state and downloads video chunks continuously from the HTTP server. When the total amount of cached data reaches a high threshold, it moves to the Idle state. No data transmission occurs in the Idle state. However, a keep-alive mechanism will be activated in the idle state. The data source provider enters into the Keep-alive state every 15 seconds and downloads 64KB video data to keep the TCP connection alive. After that, it will return back to the Idle state. Once the amount of the fresh cached data falls below a low threshold, the state machine moves to the Fetching state and starts another round of bulk data transmission.

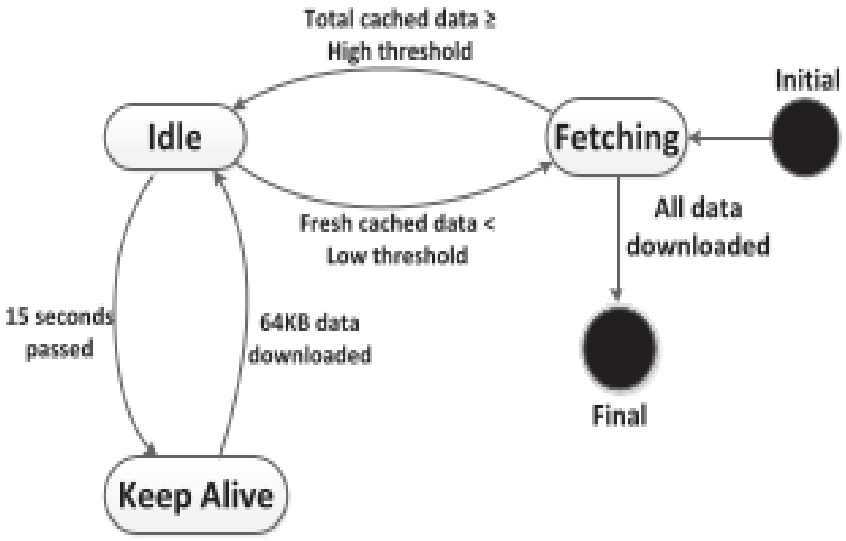

Fig.1 Finite state machine

C. Cost Incurred by Unconsumed Content

The viewing pattern of PC users is quite different from that of mobile users [3], and PC users tend to abort more frequently than PC users during viewing videos, which will result in the sunk cost of unconsumed video data (i.e., fresh content) in the cache.

We define wasted content as the amount of unconsumed content in the cache when a PC user aborts video viewing. To analyze the amount of wasted content under different configurations of cache thresholds, we write a shell script to simulate the abortion behavior of viewers and measure the average amount of wasted content due to user abortion.

D. Impacts on User Quality-of-Experience

In this section, we study how the configuration of cache thresholds impacts the user Quality-of-Experience (QoE). Interns of user QoE, we focus more on freezing duration. In the viewing process, when fresh content in the cache is exhausted, the video player has to enter into a freezing state. We first vary the value of high threshold and plot the impacts on freezing duration in. In the figure, the value of freezing duration is the accumulated freezing duration in the whole video viewing process, instead of the duration of a single freezing period.

In the viewing process, the user may encounter multiple short freezing periods. It is found that, the impact of varying high threshold is different under different bandwidth settings. When the bandwidth is set as 320Kbps, the increase of high threshold can help to reduce freezing duration in a certain degree. For the high-bandwidth environment (i.e.,400Kbps), we cannot gain more reduction of freezing duration by increasing the high threshold. When the bandwidth is insufficient (i.e, 280Kbps), the impact brought by changing high threshold does not show a clear trend.

III. SYSTEM DESIGN

In this section, we propose a control-theoretic cache management scheme to carefully balance the tradeoff between the unconsumed content cost and the user QoE. Instead of changing the entire FSM of the media framework, we consider to optimize cache management by intelligently tuning cache thresholds dynamically. The reconfiguration of cache thresholds is much easier to implement and brings marginal impact to the OS kernel codes.

The unconsumed content cost caused by user abortion is determined by the amount of unconsumed content when aborting the viewing. Assume that the video playback will continue until the end of the current time slot, if a user aborts viewing in the middle of a time slot. Let w k be the amount of wasted content in the cache when a user aborts viewing in time slot k.As the real abortion occurs at the end of time slot k, the value of w k equals b(k), which is the amount of fresh content at the\end of time slot k. For the user QoE, we focus more on the fluency of video playback. Let f k be the duration of freezing period in time slot k. We aim to minimize the value of fk in each time slot to guarantee smooth playback. From the perspective of a PC client, it prefers to keep the amount of fresh content in the cache as low as possible so as to minimize the cost of unconsumed content. In the meanwhile, it is also important to maintain a good user QoE level during video playback. However, these two objectives are conflicting with each other.

A. Algorithm Design

Our problem formulation shows that the cache state and playback state in the previous time slot can be exploited as feedback signals for cache management. The value of w k and fk can be easily obtained at the end of each time slot k.Therefore, we can design an online cache

control algorithm based on control theory to optimize both unconsumed content cost and user QoE and achieve cost-effective PC video streaming..

A user can specify a targeted control objective on the unconsumed content cost and user QoE, ($w*$, $f*$), in which $w*$ is the maximum tolerable cost of unconsumed content and $f*$ is the maximum tolerable duration of freezing time in each time slot. Normally we can set $w*$ as a small constant value and $f*$ as zero for fluent video playback.

The amount of cached fresh content at the end of each time slot (namely b(k)) actually reflects the distance to the targeted control objective. w k and fk in the previous time slot can be the feedback signal for closed-loop control. According to our measurement results, the cost of unconsumed data is more sensitive to the setting of high threshold, while the freezing duration is largely determined by the setting of low threshold. Therefore, by controlling the update of high threshold and low threshold, we can approach the targeted
control objective ($w*$, $f*$) gradually.

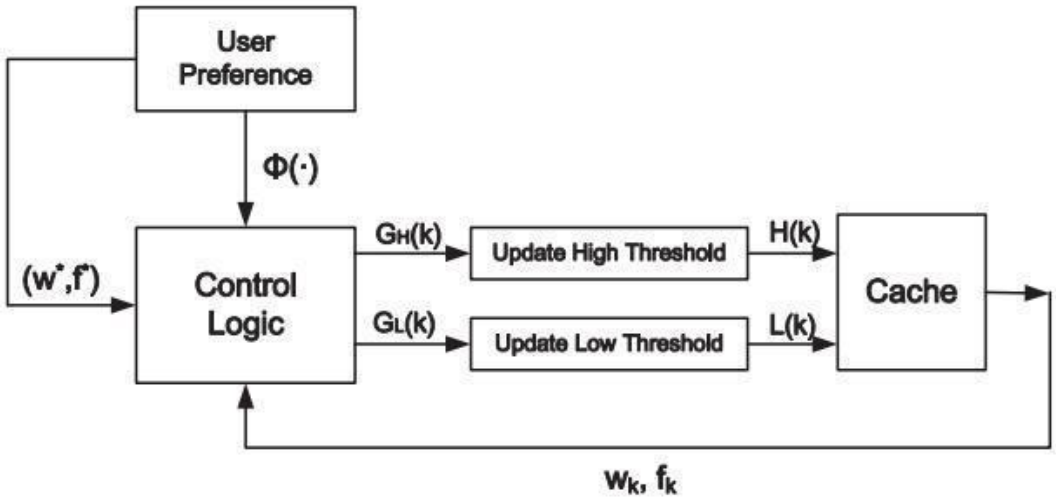

Fig.2 Overall architecture

Let w (k) and f (k) be the marginal utility incurred by the difference between the current state and the targeted state,

which are defined as below:

$$w(k) = (w_k, f_k) - (w^*, f_k) \quad (1)$$

$$f(k) = (w_k, f_k) - (w_k, f^*) \quad (2)$$

We define two separate control factors G H (k) and G L (k),where ψ_h and ψ_l are two constants which determine the smoothness of two control factors. The update of high threshold and low threshold can be governed by the following controllers:

$$H(k+1) = G(H(k) \cdot H(k)) \quad (3)$$

$$L(k+1) = G(L(k) \cdot L(k)) \quad (4)$$

For a better understanding, we plot the operation of our proposed online cache control algorithm SCAT (Smart Cache with Adaptive Thresholding) in Fig. 9. Initially, a user will specify the targeted control objective (w^*, f^*) and the utility function , which serve as the input for the control logic module. Combined with the feedback signal w_k (i.e., b(k)) from the cache and the freezing duration f_k in the past timeslot, the control logic module can determine how to update the control factors G H (k) and G L (k), and then determine high threshold and low threshold accordingly. The new threshold values will be sent to the cache module for updating.

The detailed description of our proposed algorithm is given in Algorithm 1. Note that, SCAT updates thresholds at the beginning of each time slot. The execution of SCAT will not stop until the abortion of video viewing. The time complexity of SCAT is low, as the derivation of control factors (G H (k),G L (k)) is not complicated. More specifically, its time complexity is determined by the derivation of the value of the utility function at each time slot. When updating cache thresholds, it is necessary to calculate the utility values corresponding to four different combinations of inputs. Since the cache size and freezing time can be directly obtained without further computation, the time complexity of SCAT at each time slot equals to four times of the time complexity to evaluate the utility function.

IV. EXPERIMENTAL RESULT

To better evaluate the effectiveness of different schemes, we simulate user viewing abortion behaviors and record the amount of unconsumed content. In order to reduce the variance, we run the experiment multiple times under different bandwidth settings and plot the average amount of wasted content in Fig. 13. The results show the clear advantage of SCAT under the high-bandwidth scenario (i.e., 600 Kbps). SCAT reduces the amount of unconsumed content due to viewing abortion by 87.3% compared with MaxHigh, 42.6% compared with Default and 10.7% compared with Green Tube.

It is because that SCAT will lower the high threshold to reduce the possible unconsumed content once the amount of fresh content is high, while static schemes like Default or Max High will always accumulate too much fresh content in the cache due to the unreasonable high threshold. In addition, SCAT also outperforms Green Tube, as Green Tube only minimizes unconsumed content at its expected abortion time point.

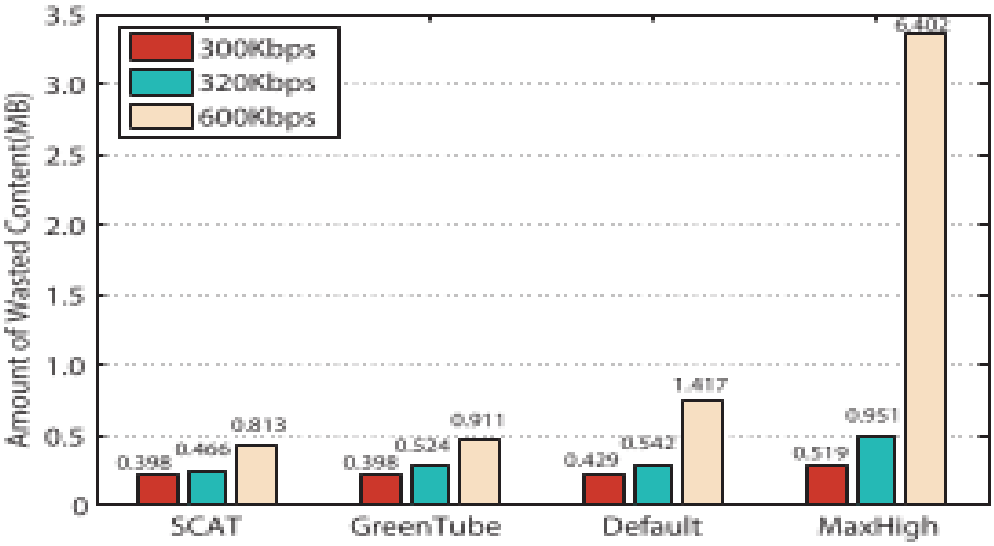

Fig.3 Average amount of wasted content

CONCLUSION

Cache management has significant impacts on the cost and performance of video streaming. we identified an implicit tradeoff between unconsumed content cost and user QoE when varying the threshold. Motivated by our findings, we further proposed a control-theoretic cache management which can achieve cost-effective video streaming by dynamic adjustment of cache thresholds. We also implemented and evaluated our proposed algorithm on the linux OS.

Our results show that, our proposed scheme can save the cost of unconsumed content compared with other schemes and significantly shorten the freezing duration in the low-bandwidth environment. In the future work, we will extend our algorithm to take more metrics related to user experiences into account. The extension is still based on adaptive cache threshold tuning to ensure the implementation be simple and practical.

REFERENCES

1) http://www.cisco.com (2013). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast

2) http://www.netmarketshare.com Operating System Market Share

3) A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in Proc. Internet Meas. Conf. (IMC), Berlin, Germany, 2011, pp. 345–360.

4) J. Huang, D. Wu, and J. He, "Demystifying the magic of cache thresholds in the Android media framework," in Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP), Hefei, China, 2014, pp. 1–6.

5) http://www.wireshark.org/ (2015) Wireshark Homepage. [Online].

6) AM. van der Schaar and S. Shankar, "Cross-layer wireless multimedia transmission: Challenges, principles, and new paradigms," IEEE Wireless Commun., vol. 12, no. 4, pp. 50–58, Aug. 2005.