A Case for Utility: Study on Real-Time Scheduling a Real-World Problem

Michael Roitzsch
Operating Systems Group
Technische Universität Dresden
mroi@os.inf.tu-dresden.de

Abstract

Within the real-time community, research on scheduling algorithms is often motivated from a theoretical perspective. The practitioners on the other hand are repeatedly unaware of the results obtained from such research and thus revert to ad-hoc and application-specific solutions. This gap impedes progress for both groups, so to help bridge it, this work intends to provide a study of an interesting commodity application — video playback — and its scheduling implications. Having a background in both real-time scheduling theory and video processing and decoding, we can approach the problem from both ends.

1. Introduction

Video playback is a use case of increasing ubiquity. Users watch video on a diverse set of devices and in different contexts. As video shows a rendition of things we are used to seeing daily in the real world, users demand a high standard of playback quality. The framerate should be high and jitter-free, turning video playback into a real-time problem with obvious deadlines. If the video content is decoded on a resource-limited device like a portable player or a phone, users desire the highest possible quality given the constraints of the device. Thus, the scheduling needs to consider an adaptive decoding process.

Scheduling theory however is often difficult for practitioners to understand and apply to real-world problems. Task models and scheduling algorithms are described purely mathematically and the typical evaluation approach is to create and analyze random task sets. This is not a flaw in the real-time research. It just means that extra care end experience are required, when an application or system developer with a problem at hand is looking for the right scheduling algorithm. But these practitioners often lack this experience and rather use ad-hoc, application specific solutions, thus partly reinventing the wheel, ignoring existing and sometimes more powerful solutions from researchers. Practitioners without an education in real-time theory also shy away from directly contributing to their system's choice of scheduling principles. They often grudgingly accept the system substrate they are given, even though it is inadequate for their needs.

There is a clear benefit in bringing these worlds closer together. With this work, we want to support this process by studying video playback as one very common use case of today's computers. With this, we hope to demonstrate to the practitioners, how different scheduling algorithms can be applied to the problem and what the results are compared to current application specific solutions. Although those specific solutions solve the problem in many cases, they have the downside of not being tied in with the system-wide scheduler and thus being disconnected from a global view on the available resources. When multiple applications consume those resources, such a global view is the only way to negotiate between competing demands.

System architects for commodity operating systems on the other hand must implement a global scheduling regime that can accommodate many different applications. For those, our work strives to provide feedback on the suitability of different scheduling models for the video use case. We do not want to start from scratch and develop a scheduler custom-tuned to video playback, as this would possibly discriminate other applications. Our goal here is to point out strengths and weaknesses of existing schedulers towards a quantification of utility we will present in more detail in Section 5.

Our contribution is therefore a study of the suitability of various scheduling algorithms for the problem of video playback. We want to bridge the lack of insight from either camp: We want to help the practitioners understand the consequences of the choice of scheduler and we want to help the theorists understand the specific properties of the given use case and their impact on the scheduling.

Before we get to the core part of the paper, the next Section will introduce video as a real-time load in more detail.

2. Video as a Real-Time Task

Video playback is employed as a demo workload by system architects, but often as a black-box application. Here, we want to provide some insight into the real-time characteristics and the inherent problems for scheduling.

2.1. Task Characteristics

Video is typically consumed in conjunction with an accompanying audio track. Compared to video, the resource demand of audio playback is considerably lower, thus we do not consider it directly in this work. But it still has an impact on the video scheduling: The audio track forces the video to strictly adhere to its specified frame rate. If a video without audio experiences resource problems, we can choose to run it 10 % slower and few people will notice. If audio is present, such a choice would require one of the following three degradations of audio playback:

- The video is allowed to fall behind the audio, thus playing out-of-sync with the sound.
- Gaps are inserted into the audio stream.
- The audio is resampled to play 10 % slower, causing a drop of almost two halftones.

Neither of those options is particularly attractive. Gaps and out-of-sync sound are extremely disruptive to the experience and an intonation drop of two halftones is recognizable even to musically untrained ears. Therefore, the video must play by its specified frame rate as soon as audio is present. We assume such a setup.

Due to that constant frame rate, the frames bear natural deadlines. Between those deadlines, the frames must be decoded, which is the major computational load for video playback. The playback process can therefore be modeled as a series of strictly periodic jobs. For some types of video, the frames are divided into slices, which can be decoded separately, thus implying multiple sub-jobs within one period.

Looking at release times of jobs, we can observe a peculiarity. If we assume pre-recorded video that is played from local storage, conceptually all frames of the entire video are released as soon as the user hits the play button. There are dependencies within the frames as they typically must be decoded with some ordering constraints, but nothing is preventing the player from decoding a frame long before it is due for display. The release times are therefore not aligned to the deadlines of previous frames. We can model them as such, if a scheduling algorithm's task model dictates that, but this clearly adds artificial constraints limiting the admission. This will be one of the aspects we plan to evaluate. Obviously, decoding frames ahead of time requires buffering, whose limits should be part of the model. In the extreme case of only one in-flight frame, the release times fall into the traditional pattern of matching the previous deadline.

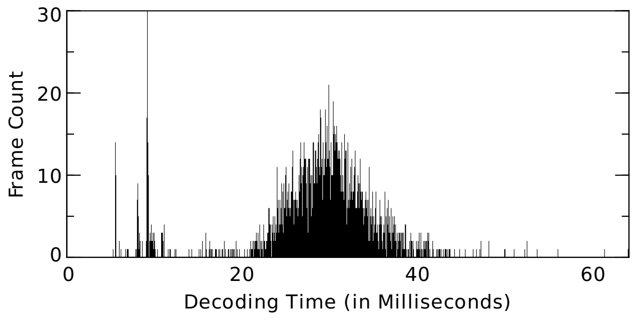

Figure 1. Long-Tail Distribution of Frame Decoding Times

In practice, the decoding pipeline of a video player incorporates multiple stages, from demultiplexing over decoding down to the output to the GPU. These stages can execute within dedicated threads, so the actual threading inside the player may differ from the conceptual task model drafted above. Decoding is by far the dominating step, so our simplified view is justified. One can also consider this as scheduling parts of the video data stream being processed rather than the actual threads who execute the processing.

2.2. Problems with Video

Regarding scheduling, video playback brings along two inherent problems:

- The decoding time of individual frames and thus the execution times of our periodic jobs can vary significantly.
- Dropping jobs by skipping or preempting them means that some frames will not be decoded. This has a non-trivially predictable and sometimes severe impact on video quality and can therefore not be done arbitrarily.

The first problem is illustrated by Figure 1. Within the distribution of frame decoding times, the mean and maximum value exhibit a ratio of 1 : 2.3. With such behavior, scheduling with hard deadlines induces heavy over-provisioning. Thus, video decoding seems to be better suited to firm and soft real-time. Some algorithms of those paradigms however require in-advance knowledge of the execution times of upcoming jobs. These are not easy to obtain for video. We developed a prediction technology which we will detail in Section 3.

The second problem of not being able to drop arbitrary frames is a property of modern video codecs. We work with H.264 [8], a widely deployed modern video coding standard. It is used in the majority of modern media applications

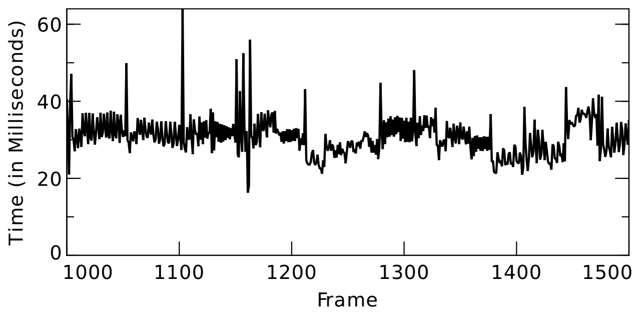

Figure 2. Decoding Time Fluctuations for a Section of an Example Video

ranging from iPod mobile videos over online movie trailers to Blu-Ray discs. This standard poses unique challenges, because previous video standards like MPEG-2 featured a class of frames (B-frames) that could be discarded without further consequences to the decoding process. A lot of earlier research results on resource adaptation and real-time scheduling relies on this fact [4]. With H.264, any frame of the video is potentially used as a reference to decode future frames, so no frames can be easily discarded. This is one of the reasons for H.264's greater coding efficiency, so it is an indispensable characteristic. Dropping a frame anyway results in quality degradations of future frames. We have devised a method, which will be discussed in Section 4, to make this quality impact manageable.

3. Video Decoding Time

Decoding time represents the machine's stake in the video playback task. CPU time is the prominent resource for scheduling, other demands of the decoding process like memory allocations are fairly static. Decoding modern video shows properties of interpreting descriptive code. Very simplified, the application of certain feature blocks of the standard is decided during decoding based on bits in the video stream. Therefore, a video codec's CPU load depends highly on the video stream being processed. As shown in Figure 2, execution times exhibit both short-term and mid-term fluctuations. The short-term variation is caused by different frame types and other mode decisions of the encoder. The mid-term behavior is due to the bit budgeting the encoder performs to mitigate differences in frame complexity.

3.1. Decoding Time Prediction

To provide advance knowledge of the execution time of upcoming decoding jobs, we developed a prediction technology [5]. The overall idea is to find a vector of metrics extractable from the bitstream for each frame. This vector's dot product with a vector of fixed coefficients gives an estimate of the decoding time. The coefficients are determined by the predictor automatically in a training phase and are then stored and used for subsequent predictions. We achieve prediction accuracies, where over 90 % of the predictions have a relative error of 20 % or less.

3.2. Scheduling Implications

Scheduling with hard deadlines means always planning for the worst case. Because of the long-tail distribution of execution times, the worst case for video decoding exceeds the average case. Such a scheduling would result in heavy overprovisioning of resources. When decoded frames are buffered and the decoder can thus work ahead, the system can at least accumulate the slack time and models like jitter-constrained streams [2] can be employed.

Scheduling under a firm deadline regime can mitigate the overprovisioning at the cost of a tolerated reduction in quality by potentially discarding some frames. If some tardiness of frames can be tolerated, even soft deadlines can be applied. The advance knowledge of upcoming execution times is used by some algorithms like SRMS [1]. They will not release jobs in the first place, if they will overrun their deadlines. Other algorithms like QRMS [3] will assign an execution time budget to jobs, release them optimistically and then preempt them as soon as the budget is depleted. Those partial results may be useful or may be thrown away. In the latter case, a job was released and consumed CPU time, but did not contribute to the visible result. We consider this gap between *effective* and *apparent* utilization an interesting property of some scheduling algorithms with firm deadlines.

4. Visual Quality

Perceived visual quality represents the user's stake in the video playback process. As noted earlier, the internals of H.264 cause quality to decrease whenever a decoding job is discarded or preempted prematurely. Current player implementations will wait until the next decoder reset whenever a frame does not meet its deadline. This avoids displaying degraded frames, but is typically even more disruptive, as full decoder resets are seconds apart in common H.264 streams. Another way to trade quality for decoding time is required.

4.1. Fallback Decoding

We developed an alternative, degraded decoding mode for H.264 frames that is faster than full decoding [6]. It is based on some additional data embedded into the video stream that allows the decoder to quickly stitch together a fallback frame using image content available in the decoder's reference frames. In addition to providing this fast fallback mode, we can also quantify the impact on visual quality for the affected frame and the quality loss propagated to future frames via referencing.

4.2. Scheduling Implications

The fallback decoding implies a mandatory minimum decoding time. The full decoding can then be modeled as an optional step, which plays into the hands of imprecise computation approaches. However, the fallback part can be skipped if the full decoding part succeeds, which is different from the traditional imprecise method. It is advantageous that the fallback decoding differs from full decoding in that it exhibits a narrower execution time distribution. Scheduling it with hard deadlines therefore induces less overprovisioning. If even the fallback is scheduled firm and a job does not meet the deadline, the decoder can still revert to waiting for the next decoder reset.

The quantification of quality loss caused by fallback decoding a frame implies priorities amongst the individual jobs. As the user's goal is maximized quality, scheduling models that respect such inter-job priorities are favorable. This leads us to the question of evaluation. We need to compare scheduling algorithms according to how well they solves the given problem.

5. Evaluation Plan

As this is very much work-in-progress, we can only present an evaluation plan here. We want to select a set of scheduling algorithms and explore different options on how the various constraints and options discussed above can be mapped to the algorithms' respective task models. Following that, we will simulate the scheduling using data from real videos. We provide the schedulers with a description of the decoding cost in terms of execution time, either aggregated as a distribution or in the form of a trace, depending on the algorithm's needs. To those schedulers whose task model can handle it, we also provide the utility of each job in terms of contributed visual quality.

Since we are decoding the videos for the user, the benchmark metric to compare the schedulers will be the achieved visual quality. We objectively measure quality with the SSIM metric [7], which matches human perception better than the simplistic PSNR. To represent the current practitioner's approach, we also have an application-specific scheduler for video playback [6]. This can serve as a baseline. We want to see, how various other schedulers perform, we intend to point out weaknesses, suggest improvements. Schedulers we are considering include QRMS [3], SRMS [1], (m,k)-firm approaches, reward-based, and multiversion scheduling. Time-utility approaches appear related, but actually, our notions of utility is completely orthogonal to the execution time. There is no function mapping execution time to utility, because the video quality is a function of the video itself, not the decoding effort.

6. Conclusion

We want to study the suitability of different schedulers for the problem of video decoding. We compare the schedulers based on the utility provided to the user: visual quality. The goal of the study is to help bridge the gap between scheduling theorists, who gain an insight in the video workload and a real-world contest amongst scheduling models, and the practitioners, who become motivated to look at real-time research results and can make a better educated choice on the scheduler to use when designing a system.

References

[1] A. K. Atlas and A. Bestavros. Statistical Rate Monotonic Scheduling. In *Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS 98)*, Madrid, Spain, December 1998. IEEE Computer Society.

[2] C.-J. Hamann. On the Quantitative Specification of Jitter Constrained Periodic Streams. In *Proceedings of the 5th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 97)*, Haifa, Israel, January 1997. IEEE Computer Society.

[3] C.-J. Hamann, M. Roitzsch, L. Reuther, J. Wolter, and H. Härtig. Probabilistic Admission Control to Govern Real-Time Systems under Overload. In *Proceedings of the 19th Euromicro Conference on Real-Time Systems (ECRTS 07)*, Pisa, Italy, July 2007.

[4] D. Isović and G. Fohler. Quality aware MPEG-2 Stream Adaptation in Resource Constrained Systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2004.

[5] M. Roitzsch and M. Pohlack. Principles for the Prediction of Video Decoding Times applied to MPEG-1/2 and MPEG-4 Part 2 Video. In *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS 06)*, Rio de Janeiro, Brazil, December 2006. IEEE Computer Society.

[6] M. Roitzsch and M. Pohlack. Video Quality and System Resources: Scheduling two Opponents. *Journal of Visual Communication and Image Representation*, 19(8):473–488, December 2008. Special issue: Resource-Aware Adaptive Video Streaming.

[7] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[8] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.