Adaptive Disk Power Management for Portable Media Players

Jaedoo Go and Minseok Song, Member, IEEE

Abstract — To support the large storage requirements, consumer electronics for video playback are increasingly being equipped with hard disk drives (HDD) that consume a significant amount of energy. A video player may prefetch many frames to give disk an opportunity to go to standby mode, but this may cause playback to be distorted or stopped if timely power mode transitions are not incorporated. We present the design, implementation and evaluation of a data prefetching scheme for energy-aware video data retrieval for portable media players (PMP). We formulate the problem when the prefetching is used for variable-bit-rate (VBR) streams to reduce disk energy consumption and then propose a new energy-aware data retrieval scheme that prefetches video data in a just-in-time way so as to increase the period in which disk stays in standby mode while guaranteeing the real-time service. We implemented our scheme in the legacy video player called Mplayer that is typically used for Linux-based consumer devices. Experimental results show that it saves energy as much as 51% compared with conventional schemes.

Index Terms — Portable Media Player, Disk Power Consumption, Video Playback

I. INTRODUCTION

Recent advances in multimedia and network technologies make it feasible to provide video-on-demand (VOD) services for various application areas, and people can access video service at any time using their consumer electronics such as personal digital assistants (PDA), personal multimedia player (PMP), mobile phone, MP3 player and so on. These devices are increasingly using hard disk drives (HDD) to store multimedia contents which typically require the large storage requirements. The HDD is the most cost-effective storage medium for the PMP, but consumes a significant amount of energy. Therefore, reducing the power consumption of the HDD becomes an important issue for the portable consumer electronics.

To reduce power consumption, modern disks have multiple power modes [1,10]: in active mode the platters are spinning and the head is reading or writing data; in seek mode the head is seeking; in idle mode the disk spins at full speed but there is no disk request; and in standby mode the disk stops spinning completely and consumes much less energy than in any other mode. Therefore, it is important to extend the length of time that disks stay in standby mode. Because returning from standby to active mode involves spinning up the disk, the energy saved by putting the disk into standby mode needs to be greater than the energy needed to spin it up again; we call the shortest idle period which justifies the energy cost of spinning up again the break-even time, and the idle period must be greater than the break-even time.

Video playback can be represented by a producer consumer model in which disk produces data that is consumed at regular interval by a video player [1,7]. Many frames may be prefetched and stored in buffer, and further requests can be handled from the buffer without accessing disk, permitting disk to enter standby mode [1,10]. More data in the buffer increases the idle period, which extend the period during which the disk stays in standby mode. This requires large buffer space because high bit-rate streams empty buffer quickly and additional cushion buffer is needed to store data displayed during power mode transitions, which may take a few seconds. But many embedded systems have small buffer space, so energy saving is limited.

To effectively utilize buffer space, data needs to be prefetched in a just-in-time way. In particular, for the video playback, timeliness of power mode transitions is important because delays may cause the playback to be distorted or stopped. What is more, video data is compressed by variable-bit-rate (VBR) compression techniques, so exhibit a high degree of data variability. For example, the ratio of the maximum and the average bit-rate of some MPEG videos can be as high as a factor of 10 [5]. These characteristics make timely transition of the power modes difficult, which may cause the playback to be distorted or stopped.

We present the design, implementation and evaluation of a data retrieval scheme for energy-aware video playback. We first analyze the relationship between the prefetched data size and energy consumption and formulate the problem when the prefetching is used for VBR data. We next propose an adaptive data prefetching (ADP) scheme that prefetches video data in a just-in-time way into buffer so as to increase the period during which disk stays in standby mode while guaranteeing the real-time video service. We finally present the implementation details of our scheme in the legacy video player Mplayer on a Linux machine.

The rest of this paper is organized as follows: We explain the related work in Section II and motivation in Section III. We propose an ADP scheme in Section IV, and present implementation details in Section V. We validate our scheme in Section VI and conclude the paper in Section VII.
II. RELATED WORK

To reduce disk energy consumption, most existing schemes involve switching disks to standby mode whenever that is possible without affecting performance. Papathanasiou et al. [6] proposed the optimal caching policy when the access patterns are known a priori. Crk et al. [3] presented an algorithm that predicts the point in time for the disk spinup so as to reduce the delays perceived by users due to power mode transitions. Chen et al. [2] proposed an energy-aware caching scheme that reshapes the access streams to make a bursty pattern. All these schemes are targeted to general workloads and may not be used for video workloads in which data needs to be read from disk continuously and sequentially.

Recently a few attempts have been made to reduce the disk energy consumption in storage servers. Pinheiro et al. [8] proposed load concentration techniques to give disks more opportunities to enter low-power mode. Zhu et al. [12] proposed buffer cache management schemes that selectively keep data in memory so that certain disks stay in low-power mode for longer periods. Gurumurthi et al. [4] proposed a dynamic multi-speed model called DPRM which dynamically adjusts the disk rotation speed. But these schemes are targeted to large-scale storage servers, so it is difficult to apply them to mobile disks used in consumer electronics.

Several energy saving techniques have been proposed to reduce disk energy consumption for video playback. Rao et al. [9] investigated the problem of choosing disk speeds for audio and video workloads. Liu et al. [11] investigated I/O speed setting strategies for multimedia workloads. Cai et al. [1] exploited data buffers to create long periods of idleness for MPEG video streams. Pettis et al. [7] analyzed disk energy consumption using analytical models. Won et al. [10] presented prefetching schemes to extend the length of time that disk stay in standby mode. All these schemes, however, do not consider VBR characteristics of video streams. In addition, most of them employ analytical analysis and do not consider the implementation issues. Only the prefetching scheme presented in [10] provides the implementation details, but its analysis assumes constant-bit-rate (CBR) video streams and may cause the playback distortion if it is applied to VBR streams. It also employs a double buffering technique for the prefetching and may not utilize the buffer space effectively.

Some works have been done to improve the performance of storage systems used for portable media players. For example, Kim et al. [13] proposed I/O optimization techniques for hybrid disk-based media players. Jo et al. [14] presented the improvement of flash memory performance for portable media players. But these schemes do not consider energy issues.

III. MOTIVATION

Power mode transitions take time during which disk requests cannot be handled. Let $T_s$ be the spinup time, $T_f$ the spindown time, $E_s$ the spinup energy and $E_f$ the spindown energy. Let $P_d$ be the power to read or write data, $P_i$ the power consumption in idle mode, $P_s$ the power to seek, and $P_l$ the power consumption in standby mode. A video player may prefetch many frames into buffer and display them without reading data from disk; meanwhile, the disk may enter standby mode to save energy. If disk is in standby mode, data cannot be read from disk, which implies that there must be sufficient data in buffer for the real-time video playback. We refer to the maximum possible period during which the prefetched data in buffer can be displayed without data produced by disk as the maximum possible idle (MPI) duration which changes with consumption of playback. To decide to permit disk to enter standby mode, our player considers the MPI duration, which needs to be greater than the break-even time, $T_b$. Because the energy saved by putting the disk into standby mode needs to be greater than the energy needed to spin it up again, the following inequality must be satisfied:

$$P_l \times (T_b - T_s - T_f) + E_f + E_s > P_i \times T_f.$$

Consequently, $T_b$ can be calculated as:

$$T_b = \frac{E_s + E_f - P_i \times (T_s + T_f)}{P_l - P_l}.$$

The data rate of the frames determines the length of the MPI duration; high bit-rate frames empty the buffer quickly, which decreases the MPI duration, whereas low bit-rate frames consume buffer slowly, which increases the duration. Due to the VBR nature of video streams, data rate varies with consumption of playback, so the MPI duration exhibits a high degree of variability. When disk enters the standby mode, the MPI duration may not exceed the break-even time, in which the mode transition is not profitable in terms of energy consumption.

Each frame needs to be read into buffer before being decoded; otherwise, the playback may be distorted or stopped because each frame must be decoded continuously. Suppose that the disk starts to be spun up when the length of the MPI duration is less than $T_s$. Then, after all the frames in the buffer are decoded, there will be no frame in buffer because disk is still spinning up. Therefore, it is important to spin up the disk in a timely manner, and the bit-rate of future frames needs to be considered when deciding the point in time for the spinup.

For prefetching, conventional scheme employs a double buffering technique in which there are two equal-sized buffers and a single lock, and retrieval and display threads access two buffers concurrently in a producer-consumer relationship [10]. Because the retrieval thread fills one buffer up quickly, disk may go to standby mode until the display thread consumes another buffer completely. This method, however, utilizes only the half of the total memory space for the prefetching and may miss a number of viable opportunities to permit disk to enter standby mode. In addition, it does not consider the VBR characteristics of streams when deciding the point in time for disk spinup and spindown, so may incur unnecessary disk spindown or cause the playback to be distorted or stopped.
IV. ADAPTIVE DATA PREFETCHING

A. Determining the point in time for disk spinup and spindown

To address the aforementioned issues, we now propose an adaptive data prefetching (ADP) scheme. To efficiently utilize the buffer space, ADP uses a buffer sharing technique in which there is a single circular buffer and display and retrieval threads concurrently access it as shown in Fig. 1. The buffer is composed of segments, each of which is filled by a single disk request. Because the disk bandwidth is higher than the playback rate of a video, the buffer becomes full even though both threads access the buffer simultaneously.

Each video \( V_i \) has \( N_i \) frames, and a video player periodically decodes video \( V_i \) at the rate of \( r_i \) frames per second (fps), so each frame is decoded once every \( \frac{1}{r_i} \) seconds. Let \( S_i(m) \) (bits) be the size of \( m^{\text{th}} \) frame \((m = 1, \ldots, N_i)\) of video \( V_i \) and \( S_i^f \) the size of the largest frame of video \( V_i \). When a decoder consumes a segment or a disk produces a segment, MPI duration is recalculated. Let us assume that the buffer contains data from the \( O_j \)th bytes of the \( f_s \)th frame to the \( O_f \)th bytes of the \( f_f \)th frame of video \( V_i \) as shown in Fig. 1. Since the frame size varies with playback, the MPI duration depends on the bit-rate of frames in buffer. Let \( I_i \) be the length of the MPI duration when playing video \( V_i \), which can be calculated as follows:

\[
I_i = \frac{1}{r_i} \times (f_f - f_s + \lfloor \frac{O_f}{S_i(f_f)} \rfloor).
\]

If the exact amount of data for the \( f_f \)th frame has been stored in buffer (that is, \( \lfloor \frac{O_f}{S_i(f_f)} \rfloor = 1 \)), then \( f_f - f_s + 1 \) frames can be decoded during \( I_i \) without data being produced from disk. Otherwise, \( f_f - f_s \) frames can be decoded. As the bit-rate of the stream increases, then \( f_f - f_s \) decreases because the number of frames retained in the buffer decreases.

Therefore, we can easily see that \( I_i \) decreases as higher bit-rate frames are in the buffer.

Whenever a segment is produced or consumed, ADP updates values of \( O_s \), \( f_f \), \( O_f \) and \( f_f \) and calculates \( I_i \). When the buffer becomes full, ADP decides whether to permit disk to go into standby mode and checks whether the MPI duration is greater than the break-even time. If \( I_i \geq T_b \), then entering the standby mode will not lead to the power reduction, which implies that additional idle time \( \alpha \) is needed to save energy. To reflect this, if \( I_i \geq T_b + \alpha \), then ADP permits disk to enter standby mode; otherwise, disk still stays in idle mode to handle disk requests.

After disk enters standby mode, ADP must decide a point in time for the disk spinup to resume reading data. When disk starts to be spun up, sufficient data must be in buffer to avoid playback distortion. In standby mode, the platter is not spinning and the head is parked, so disk seek is required when disk spins up. To prepare for the situation in which a head moves to the innermost track after spinup and full rotation after seek, we use the worst-case seek time \( T_{as} \) for seek time and the worst-case rotational latency \( T_r \) for rotational latency.

Because disk cannot produce data for \( T_r + T_{as} + T_r \), data consumed for \( T_r + T_{as} + T_r \) must be in buffer. In addition, each frame needs to be read into buffer before decoding it, so data of the next one frame consumed after spinup must be read ahead. Let \( r_t \) be the transfer rate of disk, and to prepare for the situation in which the largest frame may be decoded after spinup, data consumed for \( \frac{S_f}{r_t} \) may be in buffer. Therefore, for the timely spinup, the length of the MPI duration must satisfy the following condition;

\[
I_i \geq T_r + T_{as} + T_r + \frac{S_f}{r_t}.
\]

When disk stays in standby mode and after a segment is consumed, ADP calculates the MPI duration assuming that the next segment would be consumed and we denote it as \( Q_i \). For example, in Fig. 1, \( Q_i \) represents the maximum possible period during which data from segment 2 to segment 9 can be displayed without data produced by disk. Then ADP spins the disk up when \( Q_i \) becomes less than or equal to \( T_r + T_{as} + T_r + \frac{S_f}{r_t} \) to avoid the playback distortion.

B. Power modeling

During the entire playback, disk repeats the power cycles as shown in Fig. 2. We assume that there are \( T_i \) cycles to play video \( V_i \). Let \( T_A(m) \) be the duration that disk stays in active mode, \( T_I(m) \) the duration in idle mode, and \( T_{as}(m) \) the duration in standby mode during cycle \( m \ (m = 1, \ldots, C_i) \). We use the average seek time \( T_{as} \) for power modeling and every spinup needs one disk seek. Therefore, total energy...
consumption needed to play video \( V_i \), \( E_i \) can be calculated as follows:

\[
E_i = \sum_{m=1}^{m=C_i} (T_A(m) \times P_2 + T_I(m) \times P_1)
\]

\[
+ T_S(m) \times P_3 + T_{as} + E_s + E_f
\]

(1)

**Fig. 2.** An example of the power cycles.

**V. IMPLEMENTATION**

We have augmented a video player MPlayer on a Linux kernel 2.6 that uses an EXT2 file system. We used an ioctl system call to enable disk to enter standby mode. The ioctl system call takes three parameters; a file descriptor, a request code and a command code [10]. To spin the disk down, the player issues HDIO_DRIVE_CMD as the request code and 0xE0 as the command code. Our player analyzes the trailer information in MPEG and extracts the size of every frame before playback, and based on this, decides whether disk enters standby mode or not. The segment size corresponds to 2048 bytes. We temporarily disabled the pdflush daemon which periodically writes data to disk by making its disk access period exceed the playback duration.

**Fig. 3** shows our thread architecture in which retrieval and decoder threads run concurrently with producer-consumer relationship. The retrieval thread reads segments from disk and fills the buffer in, whereas the decoder thread reads segments from buffer and decodes them. We added the calculation module that decides the point in time for the spinup in the retrieval thread and for the spindown in the decoder thread. The two threads are synchronized; for example, when the buffer becomes full, the retrieval thread only fills the buffer after the decoder consumes a segment.

**VI. EXPERIMENTAL RESULTS**

We used a 2.5-inch hard disk drive from FUJITSU, whose parameters are shown in Table 1. We measured the energy consumption when three different videos with different bit rates are played. The details of video characteristics are shown in Table 2. The playback time is 10 minutes, and the value of \( \alpha \) is 0. We have measured the time during which disk stays in standby mode using a gettimeofday function and calculated the energy consumption using Equation (1). We compared our scheme with two conventional methods;

- **No prefetching (NP):** Prefetching is not used so disk cannot enter standby mode.
- **Double buffering (DB):** Double buffering is used for prefetching, in which there are two equal-sized buffers and retrieval and decoder threads access two buffers alternately. For example, suppose that there are two buffers, namely \( B_1 \) and \( B_2 \). While disk fills the buffer \( B_1 \) up and enters standby mode, a decoder thread decodes the frames in the buffer \( B_2 \). When the buffer \( B_2 \) becomes empty, the disk is spun up and starts to fill the buffer \( B_2 \) in; meanwhile the decoder thread decodes the frames in the buffer \( B_1 \).

**TABLE I**

<table>
<thead>
<tr>
<th>Disk Characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spinup time (( T_s ))</td>
<td>2s</td>
</tr>
<tr>
<td>Spindown time (( T_f ))</td>
<td>1.5s</td>
</tr>
<tr>
<td>Worst case seek time (( T_{ws} ))</td>
<td>22ms</td>
</tr>
<tr>
<td>Average seek time (( T_{as} ))</td>
<td>12ms</td>
</tr>
<tr>
<td>Worst case rotational delay (( T_r ))</td>
<td>11ms</td>
</tr>
<tr>
<td>Break-even time (( T_{b} ))</td>
<td>13.9s</td>
</tr>
<tr>
<td>Transfer rate (( r ))</td>
<td>33MB/s</td>
</tr>
<tr>
<td>Active power (( P_a ))</td>
<td>1.9W</td>
</tr>
<tr>
<td>Seek power (( P_s ))</td>
<td>1.9W</td>
</tr>
<tr>
<td>Idle power (( P_i ))</td>
<td>0.6W</td>
</tr>
<tr>
<td>Standby power (( P_l ))</td>
<td>0.13W</td>
</tr>
<tr>
<td>Spinup energy (( E_s ))</td>
<td>5J</td>
</tr>
<tr>
<td>Spindown energy (( E_f ))</td>
<td>2J</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>Video Characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie Name</td>
<td>Resolution</td>
</tr>
<tr>
<td>The kingdom</td>
<td>640 X 270</td>
</tr>
<tr>
<td>The happening</td>
<td>720 X 384</td>
</tr>
<tr>
<td>A clip from a drama</td>
<td>1280 X 720</td>
</tr>
</tbody>
</table>

**Fig. 4** shows how the energy consumption of the three schemes depends on the four buffer sizes (i.e. 2, 4, 8 and 12 MB) when a low bit-rate movie called “The Kingdom” is played. The ADP scheme exhibits the best performance under all workloads with an energy saving of between 10% and 47% compared with the NP scheme and between 6% and 47% compared with the DB scheme. This is because the
J. Go and M. Song: Adaptive Disk Power Management for Portable Media Players 1759

ADP scheme adaptively allows disk to go to standby mode based on the frame rate of the video. The DB scheme shows the worst performance when the buffer size is 2 MB. This can be explained as follows: first, the DB scheme utilizes only half of the total memory. Second, it does not consider the MPI duration when permitting disk to enter standby mode, which may not be profitable in terms of energy requirement because the MPI duration may not exceed the break-even time.

Fig. 4. Energy comparison when a clip from “The kingdom” is played

Fig. 5 shows how the energy consumption of the three schemes depends on four different buffer sizes when a video clip from “The happening” is played. The ADP scheme exhibits the best performance under all workloads with an energy saving of up to 48% compared with the DB scheme and up to 40% compared with the NP scheme. It is noteworthy that when the buffer size is 2 MB and the DB scheme is applied, the energy requirements cannot be measured, because disk spins up too late, which brings the playback to a halt. This fact implies that timely power-mode transitions are very important for the video playback and VBR characteristics must be considered for the decision. The DB scheme exhibits the worst performance under all workloads because the mode transition may not lead to energy reduction. From the figure, we also observe that the performance gap between ADP and NP increases as the buffer size increases. This is because increasing the buffer size increases the period in which disk stays in standby mode.

Fig. 5. Energy comparison when a clip from “The happening” is played

Fig. 6 shows the energy requirements when a high bit-rate clip from a drama is played. The ADP scheme exhibits the best performance under all workloads with an energy saving of up to 51% compared with the DB scheme and up to 22% compared with the NP scheme. When the buffer size is 2 or 4 MB, the energy requirements of the DB scheme cannot be measured because the playback has been stopped. From the figure, we also observe that the energy saving is relatively small compared with the situation in which low bit-rate videos are played, because high bit-rate videos empty the buffer quickly.

Fig. 6. Energy comparison when a clip from a drama is played

VII. CONCLUSIONS

We have proposed a new energy-aware data prefetching scheme to reduce disk energy consumption for portable media players. We analyzed the relationship between data rates of the frames and energy consumption. We then went on to propose an adaptive data prefetching scheme that spins the disk up or down based on the future idle time so as to provide real-time playback in an energy-efficient way. We have implemented our scheme in the legacy video player Mplayer on a Linux machine. Experimental results show that our scheme achieves appreciable energy savings under a range of workloads.

REFERENCES


Jaedoo Go received his B.S. degree in Computer Engineering from Korea Polytechnic University, Korea, in 2007. He is currently a M.S. candidate in the School of Computer Science and Engineering at Inha University, Korea. His research interests include embedded systems, low-power storage systems, real-time operating systems and multimedia systems.

Minseok Song received his B.S. and M.S. degrees in Computer Engineering from Seoul National University, Korea, in 1996 and 1998, respectively. He received Ph.D. degree in Electrical Engineering and Computer Science from Seoul National University, Korea, in 2004. Since September 2005, he has been with the School of Computer Science and Engineering at Inha University, Incheon, Korea, where he is currently an assistant professor. His research interests include embedded systems and multimedia systems.

