

Improving Disk Efficiency in Continuous-Media Servers with Soft-Scheduling¹

Jack Y.B. Lee and John C.S. Lui²

ABSTRACT

Continuous-media such as audio and video have stringent timing requirements for correct decoding and presentation. Consequently, system designers have commonly resorted to dimensioning the system capacity according to worst-case scenarios. One notable example is the disk scheduler, where actual performance varies considerably depending on the request mixes. Most current continuous-media servers employ the SCAN scheduler or its variants with worst-case dimensioning techniques to guarantee performance, albeit at the expense of disk efficiency. This study investigates a soft-scheduling approach to disk-scheduler design. Specifically, soft-scheduling improves disk efficiency by: (a) relaxing hard performance guarantees to statistical performance guarantees to achieve disk capacity gains; (b) employing random-placement policy in place of sequential-placement policy to achieve better utilization in multi-zone disks; (c) a novel Dual-Round Scheduling algorithm that allows some request overflows in a service round to be absorbed by the previous round; (d) a novel Early-Admission Scheduler that enables the use of large media block without adversely increasing disk response time. In addition, procedures for detecting and recovering from round overflow to minimize data loss are also presented. Results from detailed simulation of five modern disk drives show that soft-scheduling can achieve substantial gain in usable disk capacity over conventional hard-scheduling approaches.

Index Terms: Continuous media server, disk scheduler, SCAN, soft-scheduling, hard-scheduling, dual-round scheduling, early-admission scheduling, first-block replication, overflow detection and control, performance analysis and evaluation.

I. INTRODUCTION

Continuous-media (CM) delivery systems such as audio/video servers and video-on-demand (VoD) systems have been available commercially for some years. It is still the norm today to delegate the duty of delivering CM data to more specialized servers that have dedicated hardware due to the stringent timing requirement in CM data presentation. For example, it is crucial to display video frames at the correct frame rate or else video playback will become jerky. Audio poses an even more stringent timing requirement as any delay jitter in the playback process will result in audible clicks and pops, greatly reducing playback quality. Therefore the majorities of today's CM servers are dedicated, and are dimensioned according to worst-case scenarios for guaranteed performance [1-6].

For CM servers serving stored data (as opposed to data captured in real-time), the disk scheduler plays a vital role in providing glitch-free services to the end-users. Among the many disk schedulers proposed in the literature, the SCAN scheduler [5] and its variants such as Circular-SCAN (CSCAN) [5] and Group Sweeping Scheme (GSS) [4] are the most popular choices. Using worst-case dimensioning techniques, these schedulers are simple to implement and provide guaranteed performance.

However, these *hard-scheduling* approaches have several shortcomings. Firstly, as the usable disk capacity (in terms of maximum number of concurrent streams) is dimensioned according to worst-case scenarios, the disk will be under-utilized during actual operation. Secondly, most modern disk drives employ zoning to improve disk capacity. Zoning divides the disk surface into multiple regions (or zones). Each zone has a number of consecutive cylinders having the same track size (in numbers of sectors per track). As disk rotates with a constant angular velocity, outer zones can be allocated more sectors per track than inner zones. Therefore, one side-effect is that outer zones will have substantially higher transfer rate than inner zones. For hard-scheduling algorithms, one would need to dimension the disk according to characteristics of the innermost zone, and hence sacrificing additional disk bandwidth available in the outer zones.

In this study, we investigate a soft-scheduling approach to disk-scheduler design. Specifically, by designing disk scheduler with statistical performance guarantees instead of deterministic performance guarantees, we can use the disk I/O bandwidth more efficiently and at the same time, still able to satisfy the client QoS with high probability. Additionally, by placing data randomly instead of sequentially across the disk surface, we can avoid performance degradation due to disk zoning and

achieve better disk efficiency. To further increase the usable disk capacity, we propose a new Dual-Round Scheduling technique to schedule disk rounds in pairs so that some overflows in one disk round can be absorbed in the previous disk round, and an Early-Admission Scheduling technique to enable the use of larger media block sizes for better disk efficiency without adversely increasing the system response time. Finally, we present methods for detecting and recovering from disk round overflow to minimize data loss. Results from detailed simulation of five modern disk drives will be presented to quantify performance gains of the proposed techniques and compare with existing hard-scheduling algorithms.

The rest of this paper is organized as follows. Section II reviews some existing works and compares them with this study; Section III presents and formulates a model for the conventional hard-scheduling algorithms for later comparisons; Section IV presents and analyzes the proposed soft-scheduling algorithm; Section V presents and analyzes the Dual-Round Scheduling algorithm; Section VI presents and analyzes the Early-Admission Scheduling technique; Section VII deals with issues in overflow management; Section VIII presents performance results and compares various algorithms quantitatively; and Section IX concludes the paper.

II. PREVIOUS WORKS

The principle of statistical multiplexing is not new and has been applied in many different areas. In this section, we review literatures related to our study and compare them with our approach.

Vin *et al.* [7] proposed a statistical admission control algorithm for multimedia servers. Their admission control algorithm used disk round-length distribution and request size distribution to admit more streams than deterministic admission control algorithms. Their simulation results showed that substantially more streams can be admitted compared to the deterministic case. Instead of using empirical distributions, Chen *et al.* [8] took an analytical approach to derive the disk round length function and then use the Chebychev's Inequality to obtain statistical bounds. By allowing a small probability of overflow, their results also showed performance gains over conventional hard-scheduling algorithms. Note that both studies assumed disks without zoning. A more recent work by Nerjes *et al.* [9] incorporated zoning into their analytical disk model and used the method of Chernoff bounds to obtain tighter statistical bounds for the SCAN scheduling algorithm. These pioneering works on statistical admission control all focused on achieving better usable disk capacity by exploiting the statistical behavior of the disk's service-round length. The key differences are in the

way the disk's service-round length is modeled and in the way statistical bounds are obtained.

Other researchers have studied the disk zoning problem in isolation of the admission control problem. For example, Birk [10] proposed a data layout technique called Track-Pairing for video servers with multi-zone disks. Under Track-Pairing a media stream is placed alternatively between tracks in the outer zones and matching tracks in the inner zones. During retrieval, tracks from both zones will be retrieved in a cycle so that a more uniform disk throughput can be obtained. Another study by Ghandeharizadeh *et al.* [11] proposed a placement algorithm called FIXB where media blocks are striped across all disk zones in a round-robin manner. Media blocks are then retrieved from every zone in a disk cycle so all zones will be utilized to contribute to the average throughput. There are yet other methods such as Logical Track [12] and deadline-driven techniques [13] for tackling this disk zoning problem but none of these studies investigated statistical admission control issues in the context of continuous-media servers.

Our study not only incorporates a disk's statistical behavior and zoning configuration, we also consider the issue of managing service round overflows at runtime to minimize data loss. We show that service round overflows not only lead to minor data loss, but also can induce additional overflows in subsequent rounds. We propose a modified disk scheduler to incorporate overflow detection and recovery to tackle this problem in Section VII.

In addition, we propose two new techniques: Dual-Round Scheduling (Section V) and Early-Admission Scheduling (Section VI), to further improve disk efficiency under soft-scheduling. Dual-Round Scheduling further reduces overflow probability by absorbing some round overflows by the previous round. Early-Admission Scheduling substantially reduces scheduling delay by aggressively admitting new streams into the on-going service round instead of delaying to the next service round.

Another difference is in disk modeling, rather than using analytical models as approximations, we use detailed simulation to obtain the round length distribution. We observe that modern disk drives have rather complex zoning configurations such as variable zone size and irregular transfer rate among zones, that cannot be modeled by the method in Nerjes *et al.* [9]. Moreover, contrary to Nerjes *et al.* [9] we show that the distribution approaches normal even with a small number of requests per round (e.g. 10) for all five disk drives simulated. This observation significantly reduces the complexity in computing numerical results as we can use the normal distribution to approximate the simulated distribution.

III. HARD-SCHEDULING

We review the conventional hard-scheduling approach to disk capacity dimensioning in this section. Fig. 1 depicts a common retrieval and transmission scheduler serving requests from five different CM streams, denoted by numerics 1 to 5. For simplicity, we assume CM streams are homogeneous and have the same average data rate, denoted by R . Therefore in the simplest case, the scheduler will retrieve one fixed-size block of data, say of Q bytes, for each of the active streams in a service round. Requests within a disk service round are served using CSCAN (or its variants) to minimize seek-time overhead.

A. Performance Modeling

To understand how system dimensioning is performed, we first need to establish a performance model for the disk. The service time for retrieving a single request can be broken down into four components, namely fixed overhead (e.g. head-switching time, settling time, etc.), seek time, rotational latency, and transfer time. Mathematically, we have:

$$t_{request} = \alpha + t_{seek} + t_{latency} + \frac{Q}{r} \quad (1)$$

where the constant α denotes fixed overhead; the random variable t_{seek} denotes the seek time; the random variable $t_{latency}$ denotes the rotational latency; the constant Q denotes the amount of data to read; and the random variable r denotes the disk's transfer rate (see Table 1 for a summary of notations).

Using (1), we can then formulate the length of a disk service round. Assuming the use of CSCAN with up to k requests served per round, the service round length, denoted by $t_{round}(k)$, is given by

$$t_{round}(k) = k\alpha + \sum_{i=1}^k \left(t_{seek}^i + t_{latency}^i + \frac{Q}{r_i} \right) + t_{seek}^{end} \quad (2)$$

where i denotes the i^{th} request in a service round and t_{seek}^{end} is the time to position the disk head to the last track to prepare for the next sweep. We use this generic disk model for capacity dimensioning in the next section.

B. Capacity Dimensioning

Under hard-scheduling, the goal of capacity dimensioning is to determine the maximum number of concurrent media streams that can be sustained with deterministic performance guarantee. Consider a system with a homogeneous media bit-rate of R bytes per second and a constant request size of Q

bytes. Using double buffering as shown in Fig. 1, data blocks retrieved in a disk service round will be scheduled for transmission in the next round at the media bit-rate R . As long as disk service rounds are no longer than a transmission round, denoted by T_r , data flow to the clients will be continuous³. This is also known as the continuity condition in the literature. Formally, this condition can be expressed as

$$t_{round}(k) \leq \frac{Q}{R} \quad (3)$$

which must be met for all disk service rounds. In other words, the worst-case disk service round must be no longer than one transmission round.

Now consider (2) again. Worst-case rotational latency can be computed from the disk's rate of rotation. Specifically, assume the disk spins at a rate of W cycles per second, then the worst-case rotational latency is just one complete rotation, i.e. W^{-1} seconds. For seek time, it has been shown that worst-case seek overheads are incurred when requests are evenly spaced across the disk surface, provided that the seek function is concave. We denote this worst-case seek time (including the last head-repositioning time) by $t_{seek}^{\max}(k)$. Finally, the transfer rate depends on the zone at which the request is located in a multi-zone disk. To compute an upper-bound, we can use the minimum transfer rate at the innermost zone, denoted by r_{min} .

Modifying (2) with the previous upper bounds, we then have

$$t_{round}^{\max}(k) = \max[t_{round}(k)] = k\alpha + t_{seek}^{\max}(k) + k\left(W^{-1} + \frac{Q}{r_{min}}\right) \quad (4)$$

which can then be used for capacity dimensioning:

$$C = \max\left\{k \mid t_{round}^{\max}(k) \leq \frac{Q}{R}, k = 1, 2, \dots\right\} \quad (5)$$

where C denotes the dimensioned disk capacity in number of concurrent media streams.

IV. SOFT-SCHEDULING

The worst-case dimensioning technique in hard-scheduling enables the disk to provide deterministic performance guarantee. However, as with any worst-case techniques, the tradeoff would be lower disk utilization in practice as the worst-case scenario occurs very sparingly. For example, ignoring rotational latency for the moment, the worst-case seek time under CSCAN for a disk with a total of N tracks occurs with probability

$$\Pr\left\{n_i = \frac{N-1}{k+1}, i = 1, 2, \dots, k\right\} = \left(\frac{1}{N-1}\right)^k \quad (6)$$

where n_i denotes seek distance for the i^{th} request.

For a disk with $N=5001$ tracks (common nowadays) and $k=10$, this computes into a probability of 1.024^{-37} . This is clearly negligible in practice and this motivates us to investigate soft-scheduling that provide statistical guarantee rather than deterministic guarantee.

A. Statistical Capacity Dimensioning

In statistical capacity dimensioning, the objective is to find an operating point that provides higher usable disk capacity than deterministic capacity dimensioning, subjected to a given overflow probability constraint. Let $F_{round}(t, k)$ be the cumulative distribution function (CDF) for the disk service round length, i.e.

$$F_{round}(t, k) = \Pr\{t_{round}(k) \leq t\} \quad (7)$$

We then define an overflow probability constraint ε that specifies the maximum acceptable occurrence probability for violating the continuity condition in (3). Using this constraint and (7), we can then compute the usable disk capacity, denoted by $C(\varepsilon)$, from

$$C(\varepsilon) = \max\{k \mid (1 - F_{round}(T_r, k)) \leq \varepsilon, k = 0, 1, \dots\} \quad (8)$$

This is the maximum number of requests that can be served in each service round with an overflow probability no greater than ε .

B. Randomized Placement Policy

Note that storage allocation for a media title is randomized under soft-scheduling, i.e., available disk blocks are randomly selected for storing a media title. This randomization is necessary to prevent correlated overflow conditions. To see why this randomization process is needed, let us assume that media blocks from a media stream are stored sequentially onto the disk. Now suppose a new stream joins the system at round i and renders the service round to overflow (i.e. length of service round exceeding T_r). Then due to the sequential data placement, the next round will have requests in similar locations, and hence will likely experience overflow as well. Randomized placement can break-up spatial-correlation between requests from consecutive service rounds and avoid such correlated overflow conditions.

In practice, the entire placement information for a media stream can be stored in an index file. Assume 16 bits are used to store the beginning sector number for a media block, then a 2GB media stream stored in 128KB blocks will consume 32KB for storing the index file, which is negligible compared to the size of the media stream. Hence, the server can simply read the entire index file into memory at the time of stream admission.

V. DUAL-ROUND SCHEDULING

The previous soft-scheduling approach in general can achieve better usable disk capacity than hard-scheduling at the expense of a small probability of service round overflow. In this section, we propose a Dual-Round Scheduling (DRS) technique to further reduce this overflow probability so that even more streams can be admitted.

A. Read-Ahead Algorithm

One observation in the service round lengths is that a majority portion of the service rounds are shorter than T_r . This is necessary to maintaining an overflow probability to within a small threshold ϵ as discussed in Section IV-A. We define *slack time* as the length of idle time in a service round. In particular, the service round preceding an overflowed round may have sufficient slack time to absorb the extra delay in the overflowed round. This motivates us to propose DRS where the system starts the overflowed service round earlier to compensate for the longer service round length. Let the length for round x be t_x and further assumes that this particular round is overflowed, i.e. $t_x > T_r$. If the length of the preceding round ($x-1$) is less than or equal to $(2T_r - t_x)$, then we can compensate for the longer service round x by starting it $(t_x - T_r)$ seconds earlier than normal.

Let B_{early} be the number of extra buffers (each Q bytes) available for storing these early-retrieved media blocks. Then the disk scheduler can be modified as follows: If the current round finishes early (i.e. round length shorter than T_r), then the disk will immediately proceed to serve requests in the next round. This process continues until either the extra buffers are exhausted, or all requests in the next round are completely served. On the other hand, no special operation is needed if the current round does not finish early (either round length equals T_r or round overflows). We analyze the capacity gain in Section B next, and derive an upper bound for B_{early} in Section C.

B. Performance Modeling

To quantify the improvement, re-consider the round lengths for any two consecutive disk service rounds. As data placement is random, the service round lengths for any two disk service rounds are i.i.d. according to $F_{round}(t,k)$. Let $f_{round}(t,k)$ be the density function of $F_{round}(t,k)$ and let $F_{round}^{(2)}(t,k)$ be the distribution of the sum of two service round lengths, which is the auto-convolution of $F_{round}(t,k)$:

$$F_{round}^{(2)}(t,k) = \int_{-\infty}^{\infty} F_{round}(t-x,k) f_{round}(x,k) dx \quad (9)$$

Now consider an arbitrary service round i . With DRS, round i can overflow under two conditions. First, if round $(i-1)$ does not overflow, then round i will overflow only if the combined round lengths are longer than $2T_r$:

$$\begin{aligned} \Pr\{(t_i + t_{i-1}) > 2T_r \mid t_{i-1} \leq T_r\} &\leq \Pr\{(t_i + t_{i-1}) > 2T_r\} \\ &= 1 - F_{round}^{(2)}(2T_r, k) \end{aligned} \quad (10)$$

Second, if round $(i-1)$ does overflow, then it will be truncated to at most T_r (see Section VII-B). In this case, round i overflows if it is longer than T_r :

$$\Pr\{t_i > T_r \mid t_{i-1} > T_r\} = 1 - F_{round}(T_r, k) \quad (11)$$

Hence the overflow probability for round i , denoted by $\Omega(k)$, can be computed from

$$\begin{aligned} \Omega(k) &= \Pr\{t_i > T_r\} \\ &= \Pr\{(t_i + t_{i-1}) > 2T_r \mid t_{i-1} \leq T_r\} \Pr\{t_{i-1} \leq T_r\} \\ &\quad + \Pr\{t_i > T_r \mid t_{i-1} > T_r\} \Pr\{t_{i-1} > T_r\} \\ &\leq \left[(1 - F_{round}^{(2)}(2T_r, k)) F_{round}(T_r, k) \right] + \left[(1 - F_{round}(T_r, k))^2 \right] \end{aligned} \quad (12)$$

Finally, we can compute the usable disk capacity from

$$C_{DRS}(\varepsilon) = \max\{k \mid \Omega(k) \leq \varepsilon, k = 0, 1, \dots\} \quad (13)$$

C. Buffer Requirement

DRS can further increase the usable disk capacity under the same overflow probability constraint. The tradeoff is additional buffers needed to store the early-retrieved media blocks. To obtain an upper bound for the extra buffer requirement, we note that in the worst case, the second disk service round will have a length of $t_{round}^{\max}(k)$ as given in (4). To prevent overflow, the server will have to start the service round earlier by

$$T_{early} = t_{round}^{\max}(k) - T_r \quad (14)$$

seconds. Note that DRS cannot compensate for overflowed rounds with length longer than $2T_r$ as the

slack time for the previous round is bounded by T_r .

Now the time to read a media block of size Q bytes is bounded from below by

$$t_{read}^{\min} = \frac{Q}{r_{\max}} \quad (15)$$

where r_{\max} is the maximum transfer rate (i.e. at the outermost zone). Hence during the time interval T_{early} , we need at most

$$B_{early} = \min \left\{ \left\lceil \frac{T_{early}}{t_{read}^{\min}} \right\rceil, C_{DRS}(\epsilon) \right\} \quad (16)$$

extra buffers to store the early-retrieved media blocks.

It may appear that the extra buffers may also be used to increase the media block size instead of using DRS, which can also increase disk efficiency. However, increasing the media block size also results in longer disk service round time and consequently, increasing the scheduling delay for new streams. In practice, a system is likely to have been dimensioned to use the largest media block size for maximum disk efficiency and hence further increasing the block size will not be feasible. By contrast, DRS does not affect the scheduling delay as the media-block size is unchanged. Hence, one can employ DRS to further increase the usable disk capacity in a system with already optimized media block size.

VI. EARLY-ADMISSION SCHEDULING

In conventional round-based scheduler such as SCAN and CSCAN, the media block size is one of the key parameter in determining the achievable disk utilization. As current memory cost continue to drop due to rapid increases in memory density, it may appear that one can keep increasing disk utilization simply by choosing larger block sizes. However, in addition to memory cost, the usable block size is also limited by another factor – scheduling delay.

Specifically, under conventional round-based scheduler, a request arriving mid-way in round i will start retrieval in the next round ($i+1$). Due to double-buffering, the retrieved block will be transmitted in round ($i+2$). Hence, the worst-case scheduling delay is two rounds and the average scheduling delay is 1.5 rounds (Fig. 2). This scheduling delay not only affects the start-up latency, but also affects the system response time when VCR controls are performed.

A. Admission Algorithm

Soft-Scheduling enables a new way of shortening this scheduling delay. The principle is to try to admit a new stream into the current round rather than delay to the next round. If admission to the current round is successful, then the scheduling delay can then be reduced by one service round as shown in Fig. 3.

Let s_i be the start time for round i . Consider a new stream arriving at time t_a when service round i is in progress. At that instant, the disk head is either moving to the next target track (Case 1) or stationed in a track reading data (Case 2). Let n be the track number for the target track (Case 1) or current track (Case 2). Assuming Poisson arrival and randomized placement, the disk head would be equally probable to be located at any one of the N tracks for non-zoned disks when a new request arrives. For zoned disks, the probability for the disk head to be located at track n when a new stream arrives, denoted by p_n , would be skewed by the track size:

$$p_n = \frac{z_n}{\sum_{j=0}^{N-1} z_j} \quad (17)$$

where z_n is the size of track n .

We consider the case where the disk head is scanning in the forward direction with increasing track number. The case for reverse direction scanning is similar and is not repeated. To simplify notations, let $f_{seek}(k)$, $k=1,2,\dots,N-1$, be the seek function that includes both seek time and fixed overhead. Let there be u existing streams (i.e. serving u requests per round), and the requests are located in track v_1, \dots, v_u . Define $v_0=0$ to represent the initial head position and let v_{u+1} be the track number storing the first block for the new stream. Let t_{new} be the time the new request arrives and t_{due} be the time for the end of the current service round. Suppose the disk is serving request w . Then if $v_{u+1} \geq v_w$ the required media block for the new stream will be located in a down-stream location. For this case, the disk scheduler can compute the residual service time, denoted by \mathfrak{R} , including the new request from

$$\mathfrak{R} = \sum_{j=w}^u \left(f_{seek}(v_{j+1} - v_j) + \frac{Q}{r_{j+1}} + t_{latency}^{j+1} \right) + f_{seek}(N - v_{u+1}) + t_{residual} \quad (18)$$

where the first summation term is the service time for the remaining requests; the second term is the head-repositioning time; and the last term is the residual service time for the current request w .

Now if the computed residual service time does not overflow, then this new request can be admitted into this round as if it arrived before the round starts. We can check for overflow condition by

$$\max\{\mathfrak{R}\} + t_{new} \leq t_{due} \quad (19)$$

where

$$\begin{aligned} \max\{\mathfrak{R}\} = & \sum_{j=w-1}^u \left(f_{seek}(v_{j+1} - v_j) + \frac{Q}{r_{j+1}} + W^{-1} \right) \\ & + f_{seek}(N - v_{u+1}) \end{aligned} \quad (20)$$

If the computed residual service time overflows, the new request will have to be delayed for schedule in the next round.

On the other hand, if $v_{u+1} < v_w$, then the required media block for the new stream will be located in an up-stream location. For this case, the disk scheduler has two choice: it can proceed to serve all existing requests first and then come back to serve the new request – *non-preemptive schedule*; or it can backtrack to serve the new request first before proceeding with the rest of the existing requests – *preemptive schedule*.

We can compute the residual service time for the current round under non-preemptive schedule from

$$\begin{aligned} \mathfrak{R}_{nps} = & \sum_{j=w}^{u-1} \left(f_{seek}(v_{j+1} - v_j) + \frac{Q}{r_{j+1}} + t_{latency}^{j+1} \right) \\ & + f_{seek}(v_u - v_{u+1}) + \frac{Q}{r_{u+1}} + t_{latency}^{u+1} \\ & + f_{seek}(N - v_{u+1}) + t_{residual} \end{aligned} \quad (21)$$

where the first term is the service time for the existing requests; the second term is the service time for serving the new request; the third term is the head-repositioning time; and the last term is the residual service time for request w .

Similarly, we can compute the residual service time for the current round under preemptive schedule from

$$\begin{aligned}
\mathfrak{R}_{ps} = & f_{seek}(v_w - v_{u+1}) + \frac{Q}{r_{u+1}} + t_{latency}^{u+1} \\
& + f_{seek}(v_{w+1} - v_{u+1}) + \frac{Q}{r_{w+1}} + t_{latency}^{w+1} \\
& + \sum_{j=w+1}^{u-1} \left(f_{seek}(v_{j+1} - v_j) + \frac{Q}{r_{j+1}} + t_{latency}^{j+1} \right) \\
& + f_{seek}(N - v_u) + t_{residual}
\end{aligned} \tag{22}$$

where the first term is the service time for the new request; the second term is the service time for request $(w+1)$; the third term is the service time for the remaining requests; the fourth term is the head-repositioning time; and the last term is the residual service time for request w .

By comparing $\max\{\mathfrak{R}_{nps}\}$ with $\max\{\mathfrak{R}_{ps}\}$, the scheduler can choose the method with shorter delay and then check for overflow using similar method to (19) and (20). If the round does not overflow, the new stream can then be admitted into the current round, thereby shortening the scheduling delay by one complete service round of T_r seconds.

B. First-Block Replication

We note that the admission criteria in the previous section is conservative in the sense that the current round is guaranteed to be not overflowed by the new request even under the worst-case scenario. As a result, some new requests may be rejected even though it may not cause overflow.

To further improve the chance of successfully admitting a new stream into the current round and to simplify the disk scheduler, we propose a First-Block Replication (FBR) technique where the first block of a media stream is stored at the innermost track and also replicated at the outermost track. With this technique, we can guarantee that request for a new stream will always be located downstream. Secondly, as the disk head has to be repositioned to the platter edge at the end of a service round, seek time for the new request is eliminated as well. The residual service time for the current round can be computed from

$$\begin{aligned}
\mathfrak{R} = & \sum_{j=w}^{u-1} \left(f_{seek}(v_{j+1} - v_j) + \frac{Q}{r_{j+1}} + t_{latency}^{j+1} \right) \\
& + f_{seek}(N - v_u) + t_{residual} \\
& + \frac{Q}{r_{u+1}} + t_{latency}^{u+1}
\end{aligned} \tag{23}$$

and the scheduler can admit the new stream immediately if the round does not overflow.

The significance of FBR is that it guarantees the new request to be located down-stream, as if the request arrives before the round started. Consequently, rather than using the worst-case to estimate round overflow as in (19), we can approximate the probability of round overflow by the round-length distribution

$$\begin{aligned}\delta &= \Pr\{\text{overflow}\} \\ &= 1 - F_{\text{round}}(T_r, u + 1)\end{aligned}\tag{24}$$

and then the expected scheduling delay can then be obtained from

$$\begin{aligned}D &= 1.5T_r\delta + 0.5T_r(1 - \delta) \\ &\approx 0.5T_r\end{aligned}\tag{25}$$

for small δ .

Therefore, Early-Admission Scheduling with First-Block Replication can reduce the scheduling delay by 2/3 for small δ . The tradeoff is storage overhead for the replicated blocks. For example, the overhead for an one-hour MPEG1 video with bit-rate 150KB/s and block size 128KB is only 0.024%. Note that this FBR technique can be extended to replicate additional blocks such as the first block in the beginning of a chapter to achieve similar scheduling-delay reduction in performing interactive controls (e.g. chapter selection).

To quantify the gain with FBR, we assume that a scheduling delay constraint D_{\max} is given as part of the service requirement. Then the expected scheduling delay must be smaller than D_{\max} :

$$\frac{Q}{R}(0.5 + \delta) \leq D_{\max}\tag{26}$$

Rearranging gives the largest block size that can be used:

$$Q \leq \frac{RD_{\max}}{0.5 + \delta}\tag{27}$$

Note that we also need to round Q computed from (27) to integral multiples of disk sector size.

VII. OVERFLOW MANAGEMENT

The previous sections focus on performance modeling. In this section, we turn the focus to operational issues in implementing soft-scheduling. Specifically, we tackle the issue of overflow detection and control.

A. Deadline-Driven Detection

The analytical models in Section IV and V give the probability of experiencing a round overflow during system operation. In practice, the system must detect any overflow condition and take corrective actions. To illustrate, consider the scenario in Fig. 4 where overflow occurs in round i . Consequently, transmission for the last block retrieved in round i cannot proceed normally as it missed the transmission cycle. Moreover, schedule for the next round is also delayed, further increasing the likelihood of overflow in subsequent rounds. Clearly, we need to contain the problems caused by overflow to prevent error propagation.

To detect round overflow, we need to compare the round length with T_r . It may appear that the exact length of any disk round can be computed from

$$t_{round} = \sum_{j=0}^{u-1} \left(f_{seek}(v_{j+1} - v_j) + \frac{Q}{r_{j+1}} + t_{latency}^{j+1} \right) + f_{seek}(N - v_u) \quad (28)$$

Unfortunately, the exact rotational latency cannot be computed in advance and consequently the round length remains a random variable. Therefore rather than computing the exact round length, we propose a Deadline-Driven Detection algorithm for overflow detection. Specifically, the time to retrieve the i^{th} block in a round, denoted by $t_{request}(i)$, is given by

$$t_{request}(i) = f_{seek}(v_i - v_{i-1}) + \frac{Q}{r_i} + t_{latency}^i \quad (29)$$

Note that except the latency term $t_{latency}^i$, all other terms in (29) are known. Let t_{due} be the time when the current service round ends, and let d_i be the time to start retrieving block i . Then to prevent overflow in reading block i , we must ensure that:

$$t_{due} \geq d_i + t_{request}(i) + f_{seek}(N - v_i) \quad \forall i \quad (30)$$

where the last term is head positioning time under CSCAN. Hence, we can guarantee that overflow will not occur if

$$d_i \leq t_{due} - \left(f_{seek}(v_i - v_{i-1}) + f_{seek}(N - v_i) + \frac{Q}{r_i} + \frac{1}{W} \right) \quad (31)$$

where we replaced the rotational latency by the worst-case of one complete round of rotation.

The retrieval deadlines d_i ($i=1,2,\dots,u$) can be computed at the beginning of a service round. Before retrieving media block i , the system compares the current time t against d_i . If the deadline is exceeded (i.e. $t > d_i$), then it raises an overflow exception and proceeds with corrective actions discussed next.

B. Overflow Control

The goal of overflow control is to truncate the overflowed round to within T_r . One straightforward solution is to drop all requests exceeding their retrieval deadlines. To minimize data loss, we can drop a partial request by reducing the amount of data to read from Q to Q_d such that

$$t + \left(f_{seek}(v_i - v_{i-1}) + f_{seek}(N - v_i) + \frac{Q_d}{r^i} + \frac{1}{W} \right) \leq t_{due} \quad (32)$$

is satisfied. Rearranging we can then obtain the reduced block size:

$$Q_d \leq r^i \left(t_{due} - t - f_{seek}(v_i - v_{i-1}) - f_{seek}(N - v_i) - \frac{1}{W} \right) \quad (33)$$

Depending on the system design, the affected clients can handle data loss in several ways. First, the client could attempt recovery by means of retransmission if that is supported by the system. The effectiveness of this approach will depend on the amount of client buffers as well as the time required in performing the retransmission. Second, the client could minimize the effect of data loss by error-concealment. The effectiveness will depend on the coding algorithm employed as well as the type of data lost. Finally, the client may simply pause media playback for the duration of one service round and wait for the server to transmit the needed media block in the next service round. Although not transparent to the end-user, this approach is simple in implementation and independent of the media-format.

VIII. PERFORMANCE EVALUATION

To compare the performance between soft-scheduling and hard-scheduling, we conducted extensive simulations using detailed disk models obtained from the DiskSim simulator project [14,15]. We simulated five modern disk drives from three manufacturers (Quantum Atlas-III, Quantum Atlas-10K, Seagate Barracuda, Seagate Cheetah, IBM 9ES). The disk models include parameters such as seek

time, rotational latency, number of disk zones, number of cylinders per zone, number of sectors per track in each zone, etc. Block sizes of 64KB, 128KB, 256KB, and 512KB are simulated for each of the five disk models.

A. Service Round Length Distribution

Fig. 5 shows the round length distribution for the Quantum Atlas-10K disk model for four different round sizes. A remarkable observation is that the distributions all resemble the normal distribution. The same observation holds for all disk models simulated. In retrospect, this is expected since the round length is a summation of multiple random variables and hence would approach normal as predicted by the central limit theorem.

We take advantage of this observation and use the normal distribution in place of $F_{round}(t,k)$ for computing results in the following sections. As shown in Fig. 5, the normal approximation curves overlap closely with their simulated counterparts and hence justifies their use for computing numerical results.

B. Soft-Scheduling

Once the round length distribution is known, we can compute the usable disk capacity from (8). The first set of results is obtained from simulation with media bit-rates of 150KB/s (e.g. MPEG-1 video). Fig. 6-a and 6-b shows the normalized gains in disk capacity versus overflow probability constraint for bit-rate of 150KB/s and block sizes of 64KB and 128KB respectively. Fig. 7-a and 7-b shows a similar set of results for bit-rate of 600KB/s (e.g. MPEG-2 video) and block size of 256KB and 512KB respectively. Note that the normalized capacity gains is defined as

$$G = \frac{C(\varepsilon) - C}{C} \quad (34)$$

where $C(\varepsilon)$ is the usable disk capacity under soft-scheduling and C is the usable disk capacity under hard-scheduling. The lowest overflow probability constraint is set to 1×10^{-10} , equivalent to a mean-time-between-overflow of 138.5 years assuming the disk is operated continuously at full capacity 24 hours a day. Depending on the overflow probability constraint, the block size, and the particular disk model, the capacity gains ranges from around 20% to over 40%.

To further investigate the effect of media block size on capacity gains, we plot in Fig. 8 the capacity gains versus media block sizes for media bit-rate of 150KB/s and overflow probability constraint of 10^{-6} . We observe that while the capacity gains vary with the chosen block size, the gains

remain substantial for all block sizes, with all but one case exceeding 25%.

C. Dual-Round Scheduling

To investigate the additional gains by using Dual-Round Scheduling, we compute the normalized additional capacity gain from

$$G = \frac{C_{DRS}(\epsilon) - C(\epsilon)}{C(\epsilon) - C} \quad (35)$$

and plot the results in Fig. 9 for block size of 64KB and bit-rate of 150KB/s. The results clearly show that DRS can further improve capacity gains over single-round scheduling. Note there are ups and downs in the curves due to variations of the factor $C(\epsilon)$ in (35). We also observe that in general DRS is more effective for smaller overflow probability constraints. This is explained by the fact that it is more likely to have sufficient slack time in the previous round to absorb overflow when the overflow probability constraint is small. Given today's low memory cost, DRS is an attractive option to achieving better capacity at the expense of modest increase in buffer requirement (see Section VIII-E).

D. Early-Admission Scheduling

To study the capacity gains from Early-Admission Scheduling, we compute the media block size according to (27), and round it down to multiples of 64KB. The usable disk capacities for various combinations are shown in Fig. 10. The horizontal axis is the scheduling delay constraint used for computing the block size.

We observe that EAS substantially improves the capacity for all cases, including both hard-scheduling and soft-scheduling cases. For example, at a scheduling delay constraint of one second, the usable disk capacity increases from 35 to 81 (131% increase) for hard-scheduling and increases from 47 to 107 (128% increase) for soft-scheduling. The improvement in DRS is similar. These dramatic increases in usable disk capacity are explained by the fact that scheduling delay is reduced by $2/3$ under EAS. Therefore, substantially larger block size can be used to improve disk efficiency.

E. Buffer Requirement

Soft-scheduling does not modify the way in which buffer is managed and hence has the same buffer requirement as hard-scheduling. For disk scheduling algorithms such as SCAN and CSCAN, the buffer requirement will be two buffers per stream, one for disk retrieval and the other one for transmission. If DRS is employed, then additional buffers will be required to stage early-retrieved

media blocks. Using the formulae in Section V-C, we computed the per-stream buffer requirement for various media block sizes and summarized the results in Fig. 11.

Here we have two observations. The first observation is that buffer requirement in all cases increases with larger media block as expected. For single-round scheduling (SRS), buffer requirement is independent of disk models. Differences between disk models in the DRS case is due to differences in round length as discussed in Section V. The second observation is that buffer requirement of DRS is only modestly higher than SRS. This is because additional buffers are only incurred for staging retrievals whereas buffer requirement for transmission is not affected by DRS. As an illustration, a PC server with 512MB available memory will have sufficient buffers for more than 1000 concurrent streams (Quantum Atlas-10K, $Q=128\text{KB}$, $R=150\text{KB/s}$) even with DRS. Hence whether DRS should be employed would simply become a cost-effectiveness issue (i.e. memory cost versus disk costs) to be worked-out by the system designer.

IX. CONCLUSION

In this paper, we presented a new soft-scheduling approach to increase disk efficiency in continuous-media servers. Our results obtained from detailed simulations demonstrated that existing hard-scheduling approaches sacrifice substantial disk efficiency for scheduling simplicity. This is particularly significant for modern disk drives with zoning. Capacity gains in the range of 20% to 40% are achieved by soft-scheduling. With the additional dual-round scheduling technique, usable disk capacity can be further improved by another 10%~20% depending on disk models and system parameters. In addition, Early-Admission Scheduling allows the use of much larger block size without adversely increasing scheduling delay, further increasing disk efficiency substantially. Finally, we also presented complete procedures for detecting and recovering from round overflow, which are often neglected in other studies.

ACKNOWLEDGMENT

We would like to express our gratitude to Ganger *et al.* [14] for making the five disk configurations available to researchers.

REFERENCES

- [1] J. Gemmell and S. Christodoulakis, "Principles of Delay Sensitive Multimedia Data Storage and Retrieval," *ACM Transactions on Information Systems*, vol.10(1), 1992, pp.51-90.
- [2] H.M. Vin and P.V. Rangan, "Designing a Multi-User HDTV Storage Server," *IEEE Journal on Selected Areas in Communications*, vol.11(3), Jan 1993, pp.153-164.
- [3] A.N. Mourad, "Issues In the Design of a Storage Server for Video-on-Demand," *ACM Multimedia Systems*, vol(4), pp.70-86, 1996.
- [4] P.S. Yu, M.S. Chen, and D.D. Kandlur, "Grouped Sweeping Scheduling for DASD-based Multimedia Storage Management," *ACM Multimedia Systems*, vol.1(2), 1993, pp.99-109.
- [5] A.L.N. Reddy and J.C. Wyllie, "I/O Issues in a Multimedia System," *IEEE Computer*, vol.27(3), Mar 1994, pp.69-74.
- [6] D.J. Gemmell, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe, "Multimedia Storage Servers: A Tutorial," *IEEE Computer*, vol.28(5), pp.40-9, May 1995.
- [7] H. Vin, P. Goyal, and A. Goyal, "A Statistical Admission Control Algorithm for Multimedia Servers," *Proc. 2nd ACM International Conference on Multimedia '94*, San Francisco, CA USA, Oct 1994, pp.33-40.
- [8] H.J. Chen and T.D.C. Little, "Storage Allocation Policies for Time-Dependent Multimedia Data," *IEEE Transactions on Knowledge and Data Engineering*, vol.8(5), Oct 1996, pp.855-864.
- [9] G. Nerjes, P. Muth, and G. Weikum, "Stochastic Service Guarantees for Continuous Data on Multi-Zone Disks," *Proc. 16th Symposium on Principles of Database Systems (PODS'97)*, May 1997, pp.154-160.
- [10] Y. Birk, "Track-Pairing: A Novel Data Layout for VoD Servers with Multi-Zone-Recording Disks," *Proc. International Conference on Multimedia Computing and Systems*, May 1995, pp.248-255.
- [11] S. Ghandeharizadeh, S.H. Kim, C. Shahabi, and R. Zimmermann, "Placement of Continuous Media in Multi-Zone Disks," *In Multimedia Information Storage and Management* (Ed. S.M. Chung), Kluwer Academic Publisher, Aug 1996.
- [12] M.F. Mitoma, S.R. Heltzer, J.M. Menon, *Logical Data Tracks Extending Among a Plurality of Zones of Physical Tracks of One or More Disk Devices*, US Patent No. 5,202,799, April 1993.
- [13] S. Ghandeharizadeh, S.H. Kim, "Design of Multi-User Editing Servers for Continuous Media," *Multimedia Tools and Applications*, vol.11(1), May 2000, pp.339-365.
- [14] G.R. Ganger, B.L. Worthington, Y.N. Patt, *The DiskSim Simulation Environment Version 2.0*. <http://www.ece.cmu.edu/~ganger/disksim>.
- [15] B.L. Worthington, G.R. Ganger, Y.N. Patt, and J. Wilkes, "On-Line Extraction of SCSI Disk Drive Parameters," *Proceedings of the ACM Sigmetrics Conference*, May 1995, pp.146-156.

Table 1. Summary of notations.

Description	Symbol	Notes
Average media bit-rate.	R	Parameter in bytes per second (e.g. 150,000 Bps).
Media block size.	Q	Parameter in bytes (e.g. 65336 bytes).
Length of a transmission cycle.	T_r	Computed from Q/R , in seconds.
Disk transfer rate.	r	Random variable in bytes per second.
	r_i	Transfer rate for the i^{th} request.
	r_{\min}	Minimum transfer rate (i.e. of innermost track).
Fixed overhead in disk read.	α	Constant in seconds.
Seek time in disk read.	t_{seek}	Random variable in seconds.
	t_{seek}^i	Seek time for the i^{th} request.
Rotational latency in disk read.	t_{latency}	Random variable in seconds.
	t_{latency}^i	Rotational latency for the i^{th} request.
Head repositioning delay.	$t_{\text{seek}}^{\text{end}}$	Random variable in seconds.
Service time for a request.	t_{request}	Random variable in seconds.
	$t_{\text{request}}(i)$	Service time for request i .
Length of a service round serving k requests.	$t_{\text{round}}(k)$	Random variable in seconds.
Disk platter rotation rate.	W	Constant in cycles per second.
Worst-case seek time for serving k requests.	$t_{\text{seek}}^{\max}(k)$	Computed, in seconds.
An upper bound for service round length serving k requests.	$t_{\text{round}}^{\max}(k)$	Computed, in seconds.
Usable disk capacity under Hard-Scheduling.	C	Computed, in number of requests (served in a service round).
Usable disk capacity under Soft-Scheduling.	$C(\mathcal{E})$	Computed, in number of requests (served in a service round).
Usable disk capacity under Dual-Round Scheduling.	$C_{\text{DRS}}(\mathcal{E})$	Computed, in number of requests (served in a service round).
Number of disk tracks.	N	Parameter in numbers.
Seek distance for request i .	n_i	Variable in number of tracks.
Overflow probability constraint.	\mathcal{E}	Parameter.
Extra number of buffers for Dual-Round Scheduling.	B_{early}	Computed, in number of Q -bytes buffers.
Probability density function for round length serving k requests.	$f_{\text{round}}(t, k)$	Parameter.
Probability distribution function for round length serving k requests.	$F_{\text{round}}(t, k)$	Parameter.
Overflow probability for serving k requests in a round.	$\Omega(k)$	Computed.

Table 1 (continuation). Summary of notations.

Description	Symbol	Notes
Size of track i .	z_i	Parameter in bytes.
Probability of disk head located at track i .	p_i	Computed.
Track number for request i .	v_i	Parameter.
Residual service time of a round.	\mathfrak{R}	Variable in seconds;
	\mathfrak{R}_{nps} \mathfrak{R}_{ps}	for non-preemptive scheduling; for preemptive scheduling.
Arrival time for a new request.	t_{new}	Variable, in seconds.
End time for the current service round.	t_{due}	Variable, in seconds.
Seek function, including both seek time and fixed overhead.	$f_{seek}(k)$	k is seek distance in number of tracks.
Probability of round overflow under First-Block Replication.	δ	Computed.
Expected scheduling delay.	D	Computed, in seconds.
Scheduling delay constraint.	D_{max}	Parameter in seconds.
Retrieval deadline for request i .	d_i	Computed, in seconds.
Size of partial block retrieval during round overflow.	Q_d	Computed, in bytes.

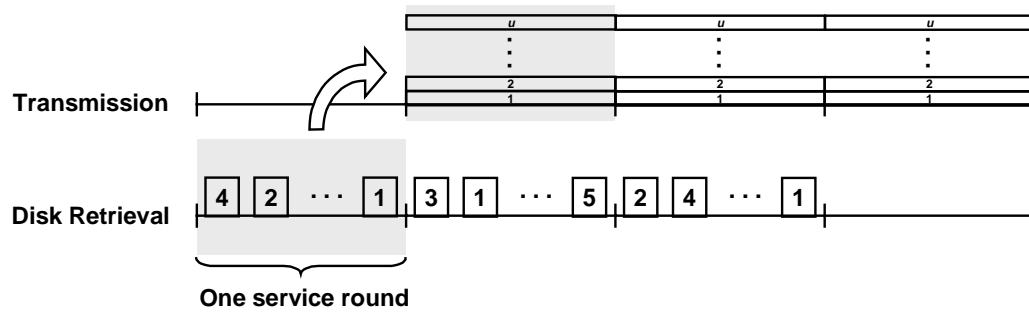


Fig. 1. Retrieval and transmission scheduling.

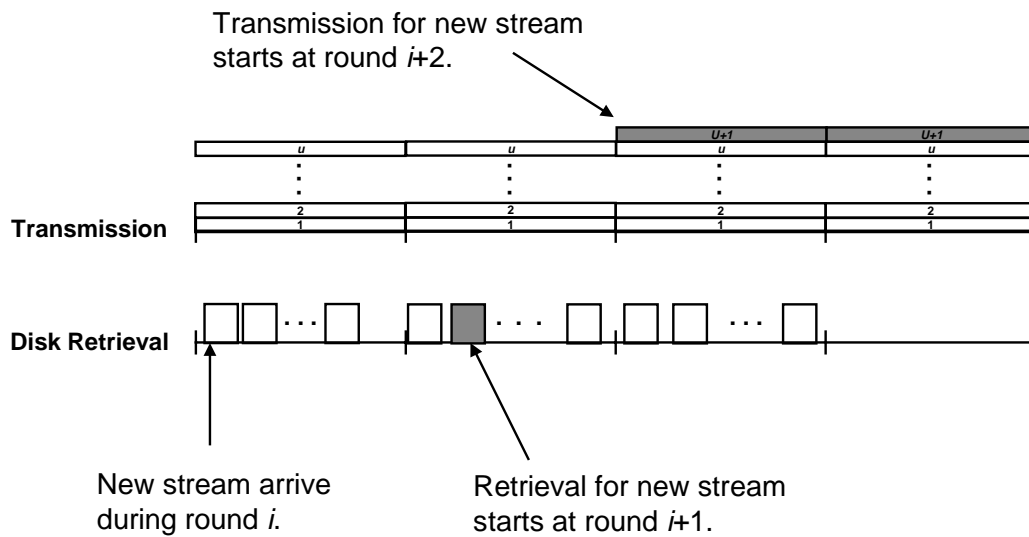


Fig. 2. Admission scheduling in conventional round-based scheduler.

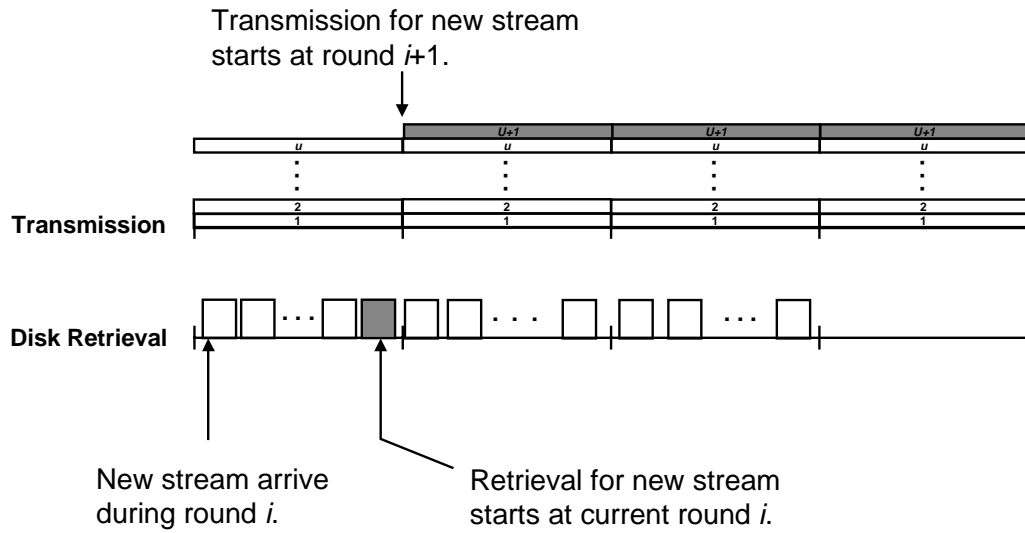


Fig. 3. Early-Admission Scheduling in Soft-Scheduling.

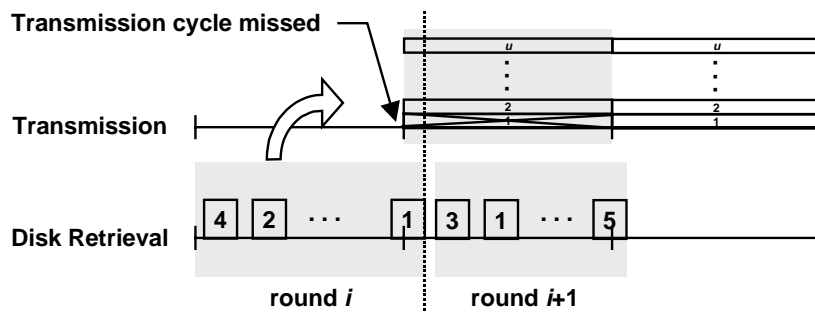


Fig. 4. Service round overflow.

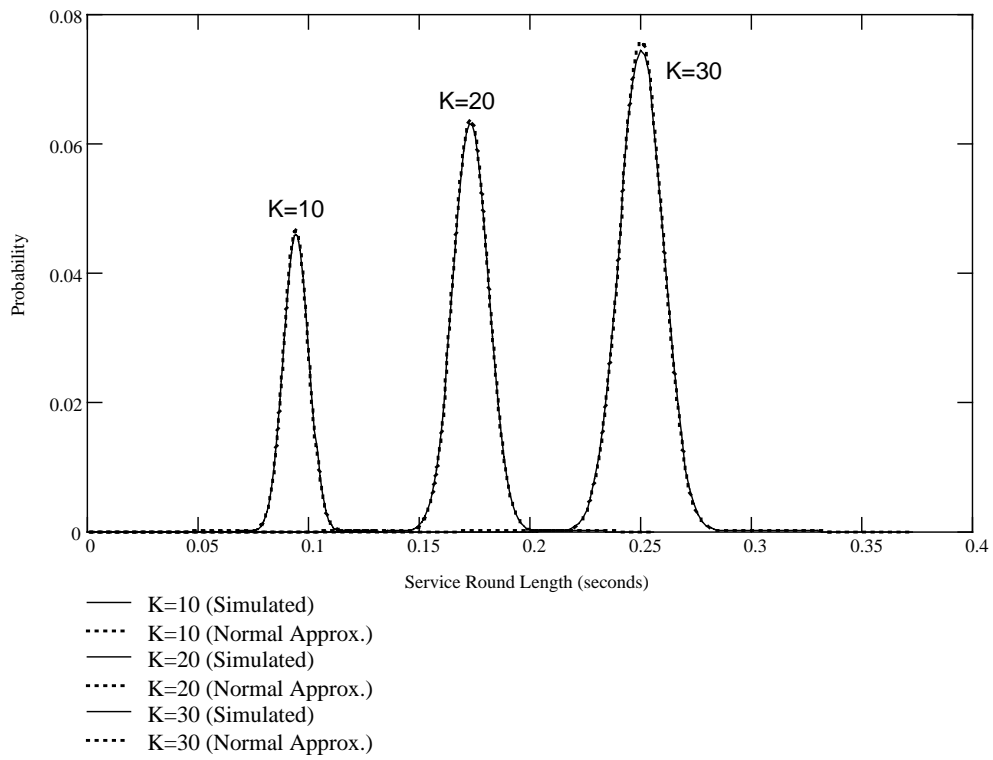


Fig. 5. Service round length distributions and the corresponding normal approximations.

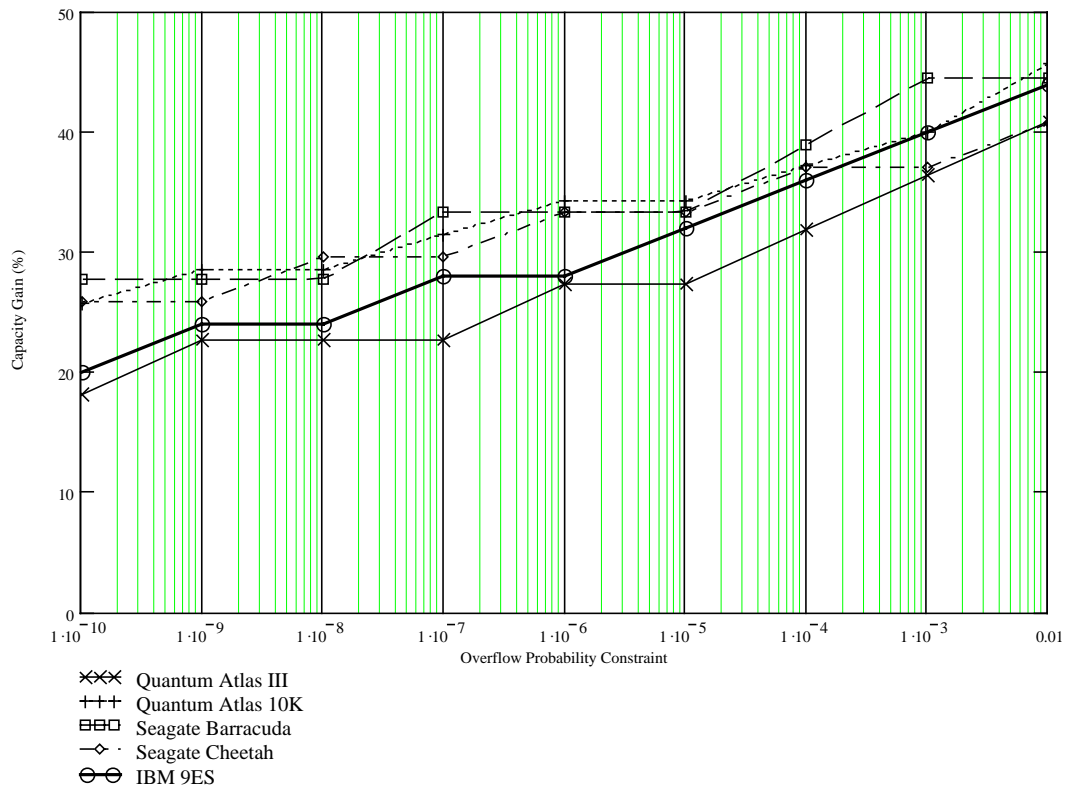


Fig. 6-a. Capacity gain in Soft-Scheduling ($Q=64\text{KB}$, $R=150\text{KB/s}$).

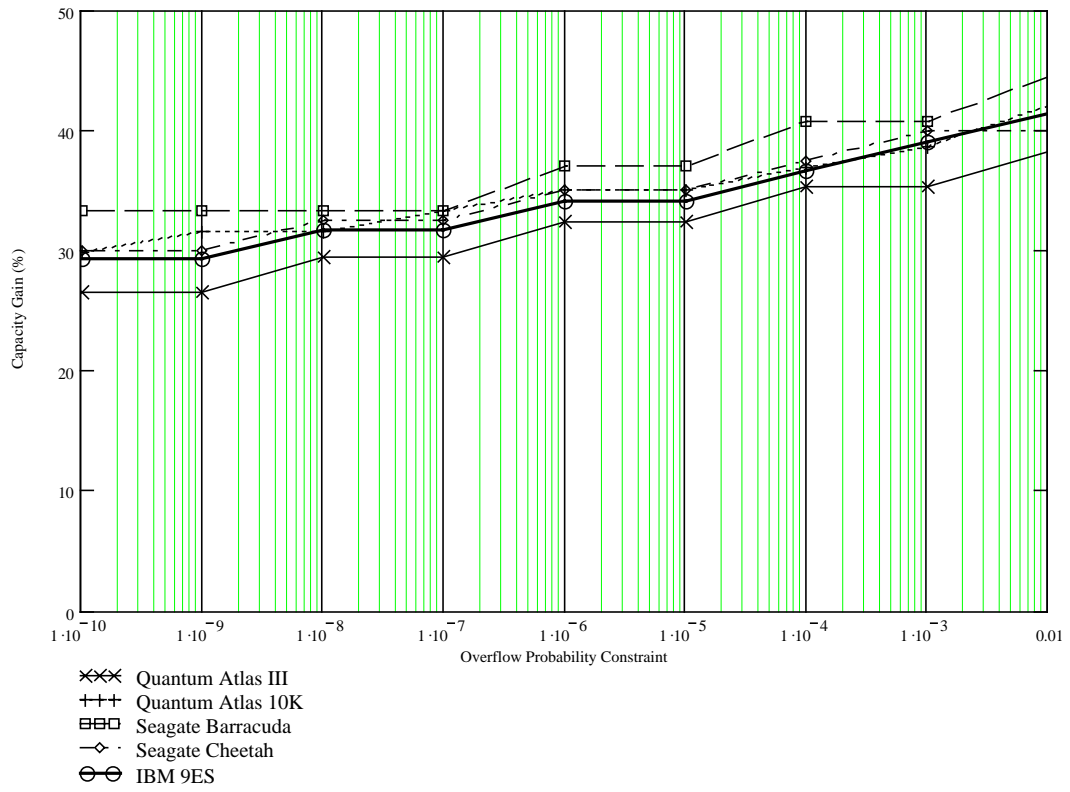


Fig. 6-b. Capacity gain in Soft-Scheduling ($Q=128\text{KB}$, $R=150\text{KB/s}$).

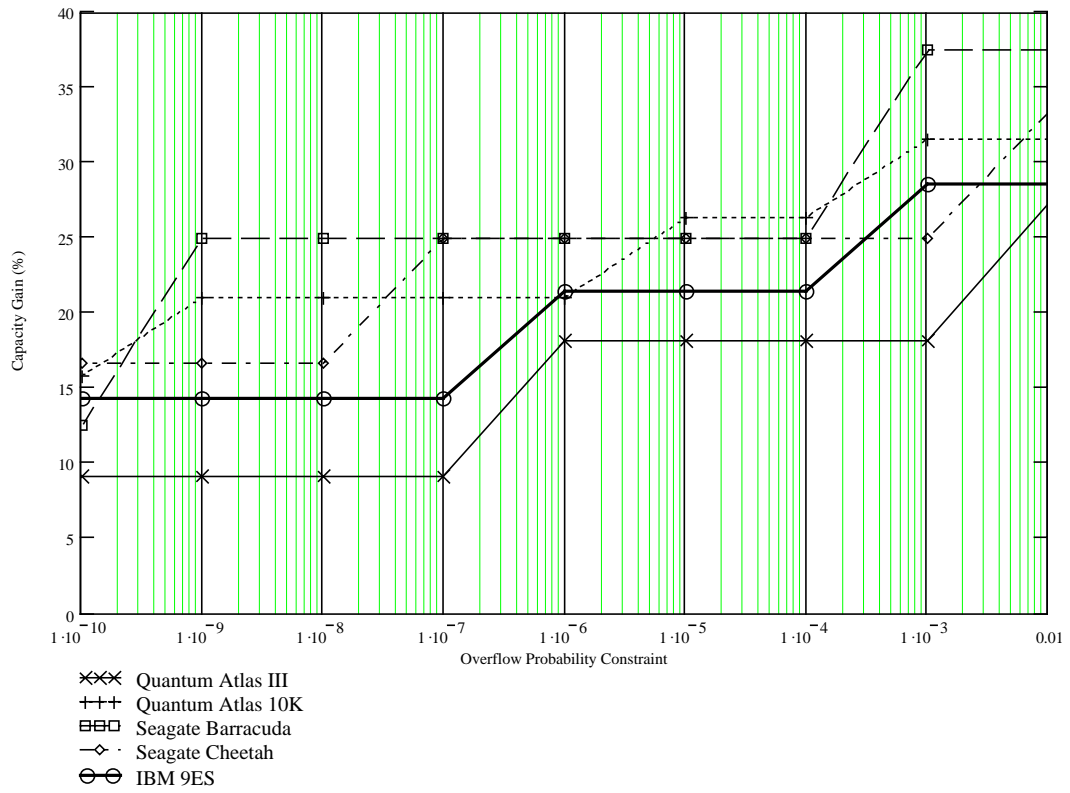


Fig. 7-a. Capacity gain in Soft-Scheduling ($Q=256\text{KB}$, $R=600\text{KB/s}$).

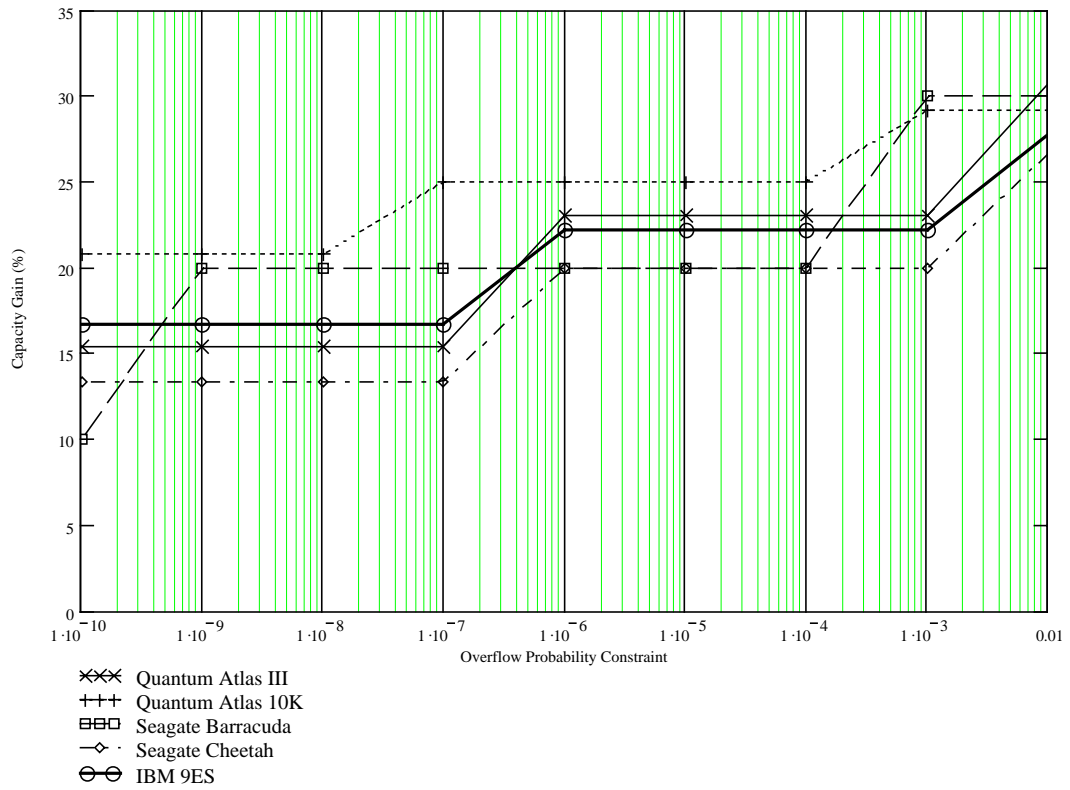


Fig. 7-b. Capacity gain in Soft-Scheduling ($Q=512\text{KB}$, $R=600\text{KB/s}$).

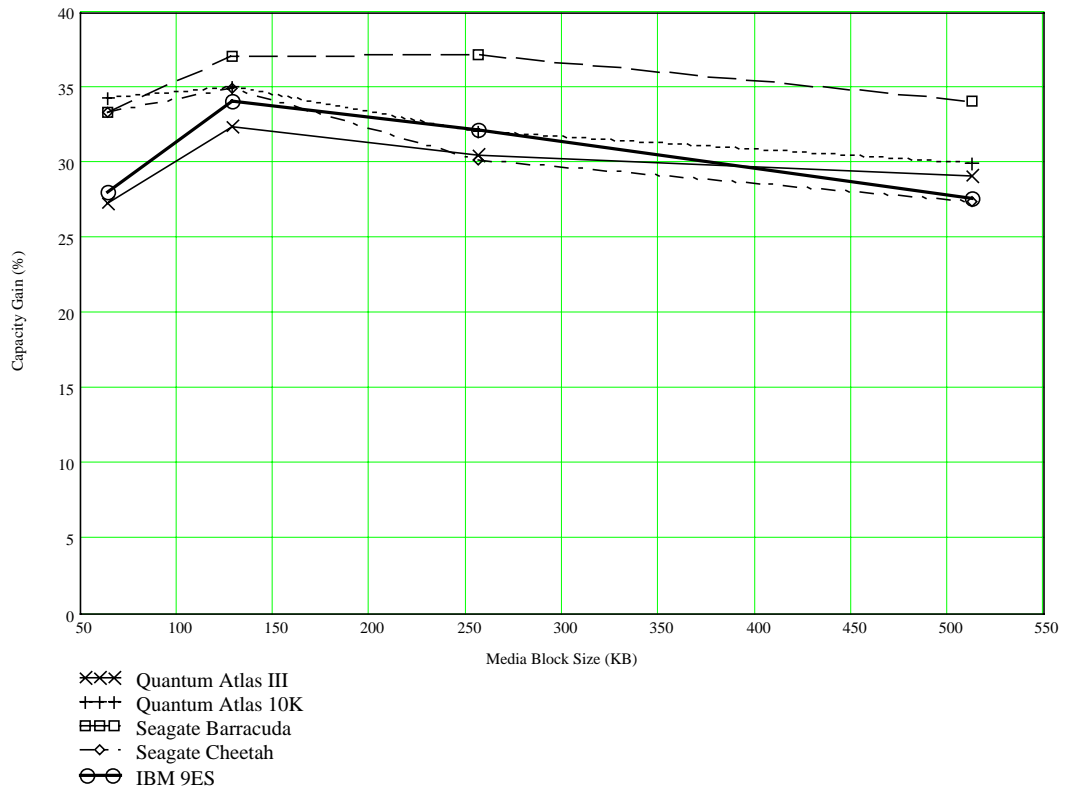


Fig. 8. Capacity gain versus media block size ($R=150\text{KB/s}$, $\epsilon=10^{-6}$).

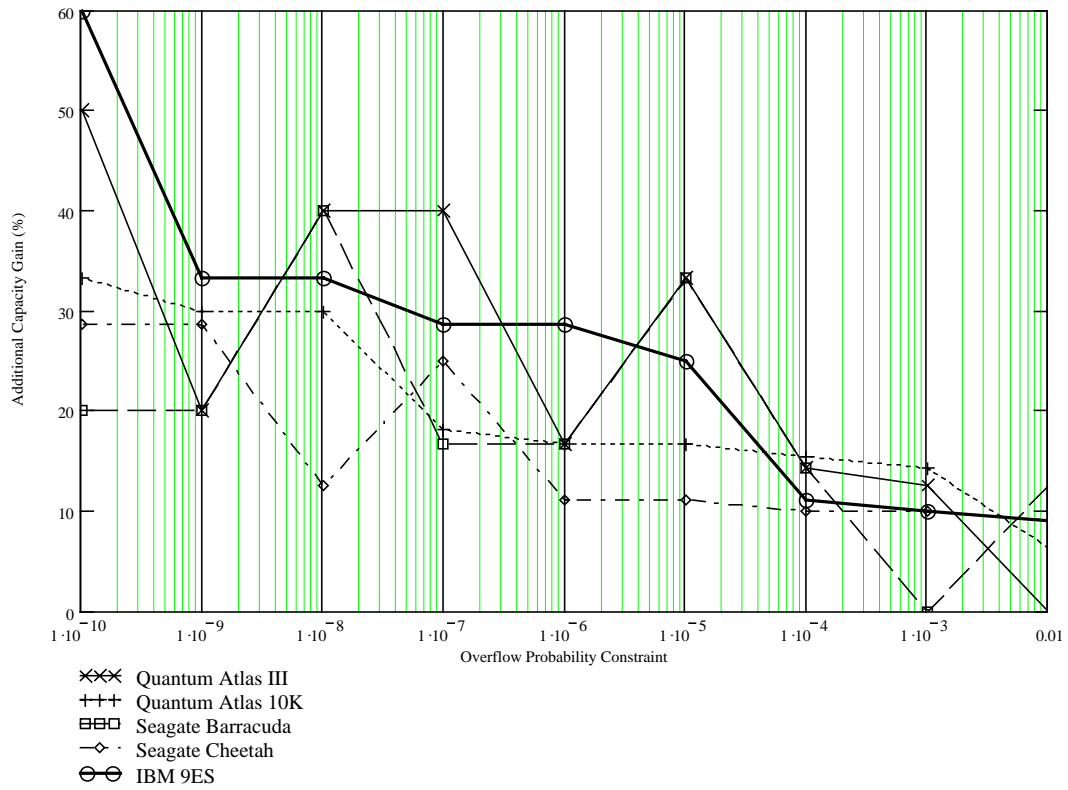


Fig. 9. Additional capacity gain due to Dual-Round Scheduling ($Q=64\text{KB}$, $R=150\text{KB}$).

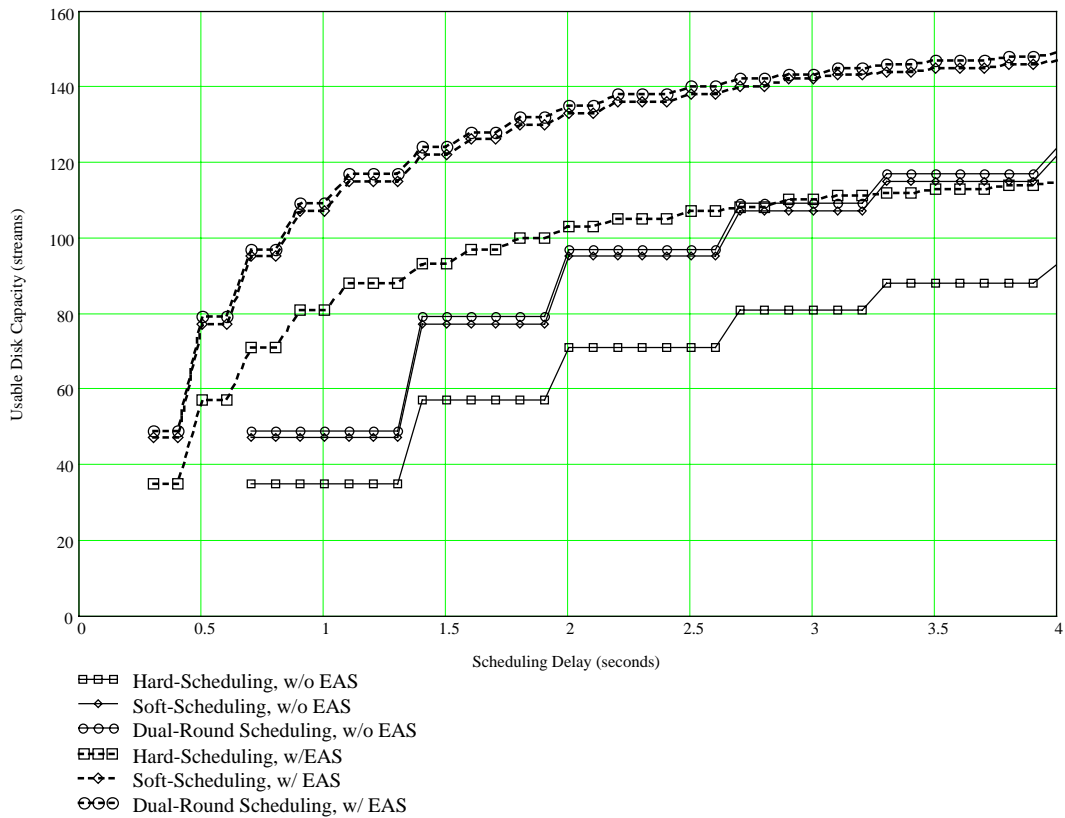


Fig. 10. Usable disk capacity versus scheduling-delay constraint ($R=150\text{KB}$, $\epsilon=10^{-6}$).

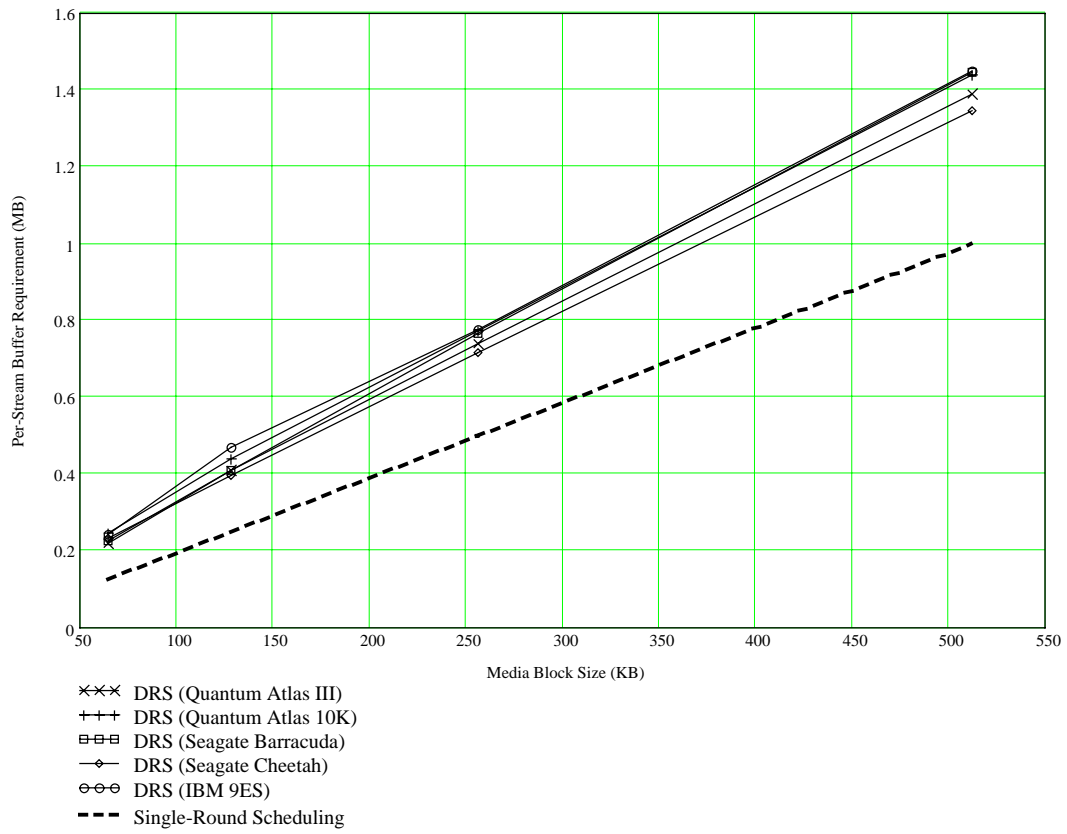


Fig. 11. Per-stream buffer requirement comparison ($R=150\text{KB/s}$, $\varepsilon=10^{-6}$).

¹ Manuscript received on _____

² Jack Y.B. Lee is with the Department of Information Engineering and John C.S. Lui is with the Department of Computer Science and Engineering, both at the Chinese University of Hong Kong, Shatin, N.T., Hongkong. Email: jacklee@computer.org, cslui@cse.cuhk.edu.hk.

³ Note that this continuity condition guarantees continuous data retrieval and transmission only. Playback continuity will also depends on other factors such as network delay jitter, loss, etc. that fall outside the scope of this study.