システムソフトウェアと
オペレーティング・システム 92−11
（２００３．２．２８）

リアルタイム環境に適用可能な Ultra DMA 転送機構の設計と実装

勝部 弘嗣 †　毛利 公一 ††　大久保 英嗣 ††

† 立命館大学大学院理工学研究科　†† 立命館大学理工学部

動画や音声を再生するアプリケーションに代表されるマルチメディアアプリケーションは，大容量の連続メディアデータを周期的に扱うという特性を持つ．そのため，オペレーティングシステムは，高速なファイルの読出しに加え，アプリケーションに対して安定したデータの供給を行うことが重要となる．これらの要求に対応するため，本稿では，Ultra DMA 転送方式によるディスク I/O に対してリアルタイムスケジューリングを行う手法を提案する．本手法では，ディスク I/O 処理に対してプリエンプションポイントを設定し，横取り可能なリアルタイムスケジューリング方式によるディスク I/O スケジューリングを行う．これにより，動画再生処理のような優先度の高いプロセスからのディスク I/O 要求の最大遅延時間を，アプリケーション設計時に考慮することが可能となる．さらに，本機構が提供するデバイス操作インタフェースにより，アプリケーション側でデータ読出し用のスレッドを作成することなく，周期的なデータの供給が可能となる．本手法の評価では，ディスクがアクセス中である割合が 98%という高負荷の状況下においても，それぞれのディスク I/O 要求の時間的制約を満たすことが可能であるという結果を得た．

Design and Implementation of Ultra DMA Transfer Mechanism applicable to Real-time Environment

Hirotsugu Katsube †　Koichi Mouri ††　Eiji Okubo ††

†Graduate School of Science and Engineering, Ritsumeikan University
††Faculty of Science and Engineering, Ritsumeikan University

Multimedia applications have a characteristic that they must manipulate huge continuous data cyclically. Therefor, it is required for operating systems not only to read data from hard disk at high speed but also to provide stable data transfer with applications. In this paper, we propose the real-time disk I/O scheduling method with Ultra DMA. A preemption point is assigned between disk I/O requests so that disk I/O requests can be scheduled according to preemptive real-time scheduling algorithms in this method. Programmers can estimate maximum delay time of processing disk I/O request from high priority applications in advance. Moreover, applications do not have to execute threads or processes for reading data, and can receive data periodically and stably from operating system. We confirmed that this method achieves that time constraints of all disk I/O requests are satisfied even under the condition that disk accessing ratio is 98%.

1 はじめに

マルチメディアアプリケーションの分野において，連続メディアデータの再生や転送などの処理を行う際には，実時間処理要求が発生する．オペレーティングシステム (以下，OS) は，プロセスの持つ時間的制約を満たすために，リアルタイムスケジューリングを行い，このような要求に対応している．しかし，スケジューラによってCPUが割り当てられても，そのプロセスの利用するCPU以外の資源も同時に割り当てられなければ処理を続行することができず，実時間制約を満たすことはできない．特に，動画や音声を再生するアプリケーションに代表されるマルチメディアアプリケーションでは，データの読出し，データの解読 (デコード)，画面やスピーカへの出力といった処理をパイプライン処理し，動画や音声の再生を実現している．パイプライン処理の第一工程にあたるデータの読出し処理は，処理するデータの枯渇を防ぐための最も重要な位置にある．しかし，従来の汎用的なOSで利用されているセクタの物理的な位置順によるディスクI/Oスケジューリングや，プロセスの優先度順によるI/Oスケジューリングでは，ディスクI/O処理に対してリアルタイム応答性を保証することが困難である．

さらに，このようなマルチメディアアプリケーションが扱う連続メディアデータは，サイズが非常に大きい．例えば，120分の動画の再生を考えた場合，MPEG2方式による圧縮では，4.7GByteの容量が必要となる．そのため，上記で述べたような時間的正当性に加え，高速なディスクアクセスを行う必要がある．このような要求に対し，ATA/ATAPI規格のUltra DMA転送では，データ転送速度は100MByte/sec以上の高速な転送速度を実現している [1]．さらに，データの転送はDMAコントローラにより行われるため，データ転送中においてもCPUを利用可能という利点を持つ．しかし，DMA転送では，転送を開始してしまうと転送が完了するまでその転送を中断することができない．そのため，ディスクI/O処理に要する時間を特定することができず，リアルタイム環境には向いていない．

これらの問題を解決するため，本論文では，ディスクI/O処理に対してプリエンプションポイントを設定し，横取り禁止時間の短縮および，その時間の予測を可能とする手法を提案する．この手法により，ディスクI/O処理に対して横取り可能なリアルタイムスケジューリングを適用することが可能となる．本論文では，その一例として優先度とプロセスの持つ時間的制約に基づいたリアルタイムスケジューリング方式を採用し，リアルタイム応答性の向上を図る．さらに，動画や音声のデコード処理のような周期的にデータを必要とするスレッドは，読出し専用のリアルタイムスレッドを作成する必要はなく，読出し要求にリアルタイムパラメータを設定するだけで，周期的なデータの供給を受けることが可能となる．

以下, 本論文では, 2章でスケジューリング手法, 3章で本機構の構成と処理手順，4章で評価，5章で関連研究について述べる．最後に6章で本論文のまとめを述べる．

2 ディスクI/O処理のリアルタイムスケジューリング手法

DMA転送方式では，転送を開始してしまうと，転送が完了するまでその転送を中断することができない．そのため，DMA転送方式を用いた場合，ディスクI/O処理は，横取り不可能な処理として考えられてきた．このような考え方では，優先度の高いスレッドからの要求がどれだけ遅延するかは，要求を発行したタイミングにより変化する．そのため，その遅延時間をアプリケーション設計時に予測するのは非常に困難であり，リアルタイム性を実現するのも困難となる (図1)．

本手法では，1回のATAコマンドにおけるデータ転送量を制限することにより，DMA転送方式によるディスクI/O処理に対してプリエンプションポイントを設定する．例えば，プリエンプションポイントを256セクタアクセスごとに設定した場合，1セクタが512Byteとすると，そのデータ容量は，128KByteとなる．このとき1MByteの読出し要求が発生した場合，システムは，IDEコントローラに対して8回の読出し要求を発行することになる．これは，1MByteの読出しというディスクI/O要求に

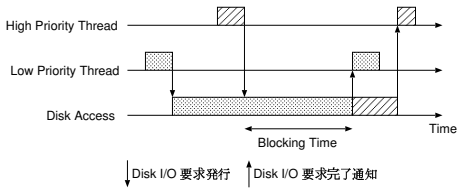

図 1　横取り不可能なディスクI/O処理

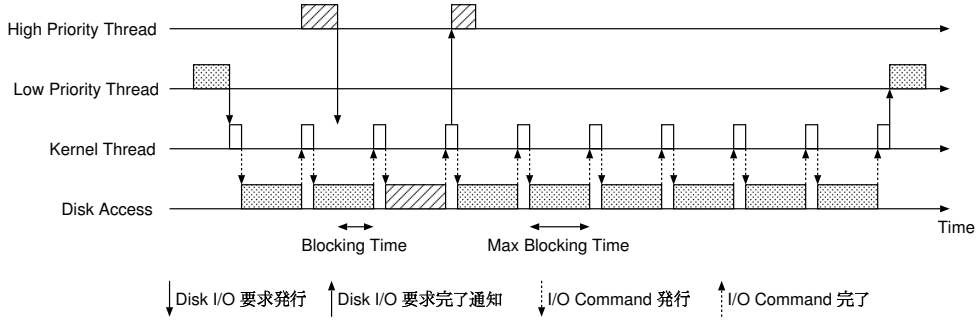

図 2　プリエンプションポイントを設定したディスク I/O 処理

対して，8 回のプリエンプションポイントが存在するということである．すなわち，1MByte の読出し処理中に，それよりも高優先度のディスク I/O 要求が到着した場合，1MByte すべての転送が完了するのを待つ必要はなく，最大 256 セクタをアクセスする時間だけ待てばよいことになる．そのため，プリエンプションポイントを考えた場合，高優先度のスレッドからの要求は，要求発行時から，256 セクタをアクセスする時間以内に必ずサービスが開始される (図 2)．256 セクタをアクセスする時間は，ディスクや IDE コントローラの性能 (位置決め時間，回転待ち時間，転送速度) から算出可能である．これにより，アプリケーション設計時に，ディスク I/O 要求に関する遅延時間を考慮することが可能となる．

さらに，ディスクデバイスを横取り可能であると考えることにより，ディスク I/O 処理に関して，優先度スケジューリングだけでなく，EDF(Earliest Deadline First) スケジューリング方式 [2] や，RM(Rate Monotonic) スケジューリング方式 [2] などのリアルタイムスケジューリングを適用することが可能となる．これにより，ディスク I/O 処理に対して時間的制約を考慮したリアルタイムスケジューリングを行うことが可能となる．

3　リアルタイム環境に適用可能な Ultra DMA 転送機構

3.1　全体構成

本機構の全体構成を図 3 に示す．ユーザスレッド (アプリケーション) は，本機構を直接呼び出すことはなく，本機構の提供するサービスは，ファイルシステムにより抽象化されユーザスレッドに提供される．本機構は，ファイルシステムに対しデバイス操作インタフェースを提供し，ファイルシステムからのディスク I/O 要求を，2 章の手法に基づきリアルタイムスケジューリングを行う．動画や音声のデコード処理のような周期的にデータを必要とするスレッドは，ディスク I/O 要求に周期やデッドラインを設定することにより，1 回のシステムコールで，周期的にデータの供給を受けることが可能となる．以下，ファイルシステム，デバイス操作インタフェース，ディスク I/O スケジューラ，ディスク I/O 要求処理スレッドについて詳しく述べる．

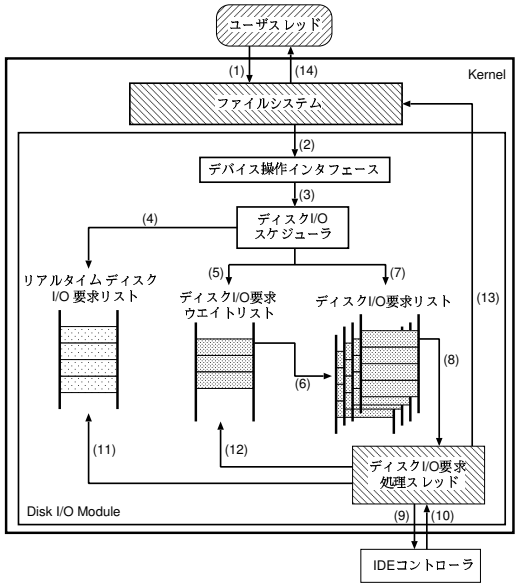

図 3　DMA 転送機構の構成

3.2　ファイルシステム

ファイルシステムは，ディスクデバイスを抽象化し，ディスクを利用するためのインタフェースをユーザスレッドに提供する．ユーザスレッドは，このインタフェースを利用し，ファイル名によるディスクの利用を要求する．ファイルシステムは，ユーザスレッドの要求するファイル名からディスクの物理的な位置の解決を行い，本機構の提供するデバイス操作インタフェースを呼び出すことにより，実際の

ディスク操作を行う．このとき，ファイルシステムから本機構へ，要求を発行したユーザスレッドのスレッド ID，優先度，転送先アドレス，開始セクタ番号，転送バイト数が，最低限必要なパラメータとして渡される．優先度は，ユーザスレッドの CPU 割り付けの優先度を継承する．連続メディアデータの再生など，ユーザスレッドがデータを周期的に必要とする場合は，周期，1 周期に転送するバイト数，相対デッドラインを設定する．さらに，必要に応じて，リリースタイムを設定することを可能とする．リリースタイムは，相対時刻，絶対時刻のどちらでも指定することが可能である．また，ファイルシステムは，連続メディアデータをディスクの物理的に連続した領域に配置する．これにより，転送に要する時間の予測性の向上を図る．

3.3 デバイス操作インタフェース

ディスク I/O スケジューラは，データの読出し処理のインタフェースとして，ファイルシステムに対し，次に示すデバイス操作インタフェースを提供する．

- readdma(IDENo, StartSector, TransferSize, BufferAddress)
- rtreaddma(IDENo, StartSector, TransferSize, TotalTransferSize, RTParameter, BufferAddress, flag)

ファイルシステムは，時間的制約のないディスク I/O を要求するときには readdma を使用し，実時間制約のあるディスク I/O に対しては rtreaddma を使用する．これらのデバイス操作インタフェースのパラメータを表 1 に示す．本機構の扱うディスク I/O 要求は，要求を発行したユーザスレッドのスレッド ID，優先度，転送先アドレス，開始セクタ番号，全転送バイト数，1 周期に転送するバイト数，周期，デッドライン，リリースタイム，ATA コマンドリストから構成され，デバイス操作インタフェースによってファイルシステムから受け取ったパラメータを基に作成される．本機構では，ATA の 1 コマンドにおける最大のセクタアクセス値である 256 セクタアクセスごとにプリエンプションポイントを設定する[1]．そのため，ユーザスレッドの要求する転送バイト数が 128KByte を越える場合，複数の ATA コマンドを作成する必要がある．作成された ATA

[1]但し，この値はハードウェア仕様によって異なってくる．

表 1　デバイス操作インタフェースのパラメータ

パラメータ	内容
IDENo	IDE のドライブ識別子
StartSector	開始セクタ番号
TransferSize	転送バイト数 (Byte) (1 周期に転送するバイト数)
TotalTransferSize	全転送バイト数 (Byte)
RTParameter	リアルタイムパラメータ
Cycle	周期
Deadline	相対デッドライン
ReleaseTime	相対リリースタイム
AbsReleaseTime	絶対リリースタイム
BufferAddress	転送先バッファアドレス
flag	同期フラグアドレス

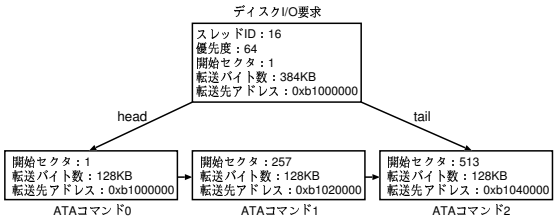

図 4　ディスク I/O 要求と ATA コマンド

コマンドは，ディスク I/O 要求の ATA コマンドリストにより管理される．スレッド ID が 16，優先度が 64 のユーザスレッドから，開始セクタ 1，転送バイト数 384KByte，転送先アドレス 0xb1000000 のディスク I/O 要求が発行された場合，本機構は，これらのパラメータを図 4 のように管理する．

3.4 ディスク I/O スケジューラ

ディスク I/O スケジューラは，3.3 節のデバイス操作インタフェースにより作成されたディスク I/O 要求を，2 章の手法に基づきディスク I/O 処理に関して横取りを可能とするリアルタイムスケジューリング方式によりスケジューリングを行う．以下，本機構におけるディスク I/O スケジューリング方式について述べる．

3.4.1 ディスク I/O スケジューリング方式

横取りを可能とするリアルタイムスケジューリングは，さまざまなものが提案されている．本機構では，その一例として，優先度付き EDF スケジューリング方式を採用する．これは，本機構の実装対象であるリアルタイム OS Easel[3] が，スレッドスケジューリング方式として優先度付き EDF スケジューリングを採用していることによるものである．本スケジューラでは，ディスク I/O 要求を，デッドラインの設定されたディスク I/O 要求とデッドラインの設定されていないディスク I/O 要求の 2 つに分類す

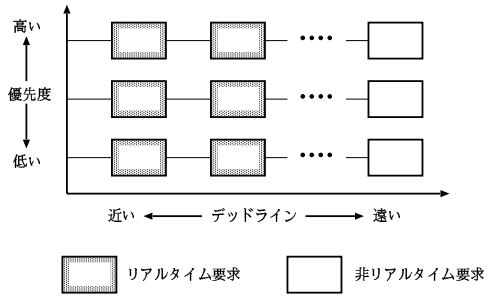

図 5　優先度付き EDF スケジューリング方式

る．前者をリアルタイムディスク I/O 要求と呼び，後者を非リアルタイムディスク I/O 要求と呼ぶ．優先度付き EDF スケジューリング方式では，図 5 に示すように，ディスク I/O 要求を優先度別のリストに繋ぎ，重要度別に分類する．さらに，同レベルの重要度を持つディスク I/O 要求については，デッドラインの近い順にソートする．非リアルタイムディスク I/O 要求は，無限遠のデッドラインを設定したディスク I/O 要求とみなし，該当する優先度のリストの最後尾に繋ぐ．これらは，ディスク I/O 要求リストにより管理される．ディスク I/O 要求リストは，CPU スケジューラのレディキューに相当し，このリストの中に存在している時間が，ディスク I/O サービスまでの遅延時間となる．

リリースタイムが設定されたディスク I/O 要求や周期的なアクセスのためのディスク I/O 要求を実現するために，ディスク I/O スケジューラは，ディスク I/O 要求ウエイトリストを利用する．ユーザスレッドからの要求にリリースタイムが設定されており，システムタイムがディスク I/O 要求のリリースタイムに達していないとき，そのディスク I/O 要求をディスク I/O 要求ウエイトリストに挿入し，リスト内のディスク I/O 要求をリリースタイム順に管理する．また，ディスク I/O 要求に周期が設定されており，次の周期まで待つ間も，このリストにより管理する．ディスク I/O 要求ウエイトリストは，タイマ割込み時にチェックされ，システムタイムがリリースタイムを満たしたとき，そのディスク I/O 要求をディスク I/O 要求リストに挿入する．

3.4.2　ディスク I/O 要求のデッドライン検出方式

ディスク I/O 要求にリアルタイムパラメータが設定されているとき，そのリアルタイムパラメータをリアルタイムディスク I/O 要求リストによりデッドライン順に管理する．リアルタイムディスク I/O 要求リストは，タイマ割込み時にチェックされ，システムタイムがデッドラインに達したとき，そのディスク I/O 要求をデッドラインミスとして検出する．

3.4.3　ディスク I/O スケジューラの処理手順

ユーザスレッドからディスク I/O 要求が発行され，本機構がその要求をスケジューリングするときの処理手順を以下に示す．

(1) ユーザスレッドが，ファイルシステムに対しデータを要求する (図 3-(1))．

(2) ファイルシステムは，ユーザスレッドの要求するファイル名から，データが格納されているセクタを算出し，本機構の提供するデバイス操作インタフェースを呼び出す (図 3-(2))．

(3) 本機構は，ユーザスレッドのスレッド ID，優先度，転送先アドレス，開始セクタ番号，転送バイト数，さらに，必要に応じて周期，デッドラインを基にディスク I/O 要求を作成する (図 3-(3))．

(4) リアルタイムパラメータが設定されている場合，そのパラメータをリアルタイムディスク I/O 要求リストへ登録する (図 3-(4))．

(5) ディスク I/O 要求にリリースタイムが設定されている場合．

  (a) リリースタイムを基にディスク I/O 要求ウェイトリストに登録する (図 3-(5))．

  (b) システムタイムがリリースタイムに達したとき，そのディスク I/O 要求をディスク I/O 要求リストに遷移させる (図 3-(6))．

(6) リリースタイムが設定されていない場合は，リリースタイムに現在のシステムタイムを設定し，優先度に応じたディスク I/O 要求リストを選択する．周期，デッドラインが設定されている場合は，選択したリストの中で，デッドラインに応じた場所に挿入する．周期，デッドラインが設定されていない場合は，選択したリストの末尾に挿入する．(図 3-(7))．

3.5　ディスク I/O 要求処理スレッド

実際に IDE コントローラに対してディスク I/O 要求を発行する処理は，カーネルスレッドであるディスク I/O 要求処理スレッドにより行われる．ディスク I/O 要求処理スレッドは，ディスク I/O 要求リ

ストが空の間もしくはデータ転送中は，待ち状態となっている．以下に，ディスク I/O 要求処理スレッドの処理手順を示す．

(1) ディスク I/O 要求の到着をトリガとして，I/O 要求処理スレッドは，待ち状態から実行可能状態に遷移する．

(2) ディスク I/O 要求処理スレッドに CPU が割り当てられると，ディスク I/O 要求リストから最も優先すべきディスク I/O 要求をチェックし，そのディスク I/O 要求に繋がれている ATA コマンドのうち先頭のものを取り出す．(図 3-(8))．

(3) 取り出した ATA コマンドを基に，IDE コントローラに対してコマンドを発行する (図 3-(9))．

(4) ディスク I/O 要求処理スレッドは，待ち状態に遷移する．

(5) IDE コントローラからの DMA 転送完了の割込みをトリガとして，待ち状態から実行可能状態に遷移する (図 3-(10))．

(6) (3) で発行した ATA コマンドが，そのディスク I/O 要求の最後のコマンドであった場合，以下を実行する．

  (a) 周期が設定されていればリリースタイム，デッドラインを再設定し (図 3-(11))，ディスク I/O 要求ウエイトリストに登録する (図 3-(12))．

  (b) ファイルシステムにデータ転送完了の通知を行い (図 3-(13))，ファイルシステムから，ユーザスレッドへデータ転送完了の通知を行う (図 3-(14))．

(7) 次のディスク I/O 要求を処理する．ディスク I/O 要求がない場合は待ち状態へ遷移する．

4 評価

4.1 評価環境

我々が開発を行っているリアルタイム OS Easel に本機構を実装し，以下の評価を行う．ただし，Easel では，ファイルシステムが未実装であるため，3.3 節で示したデバイス操作インタフェースを，システムコールとして実装し，ユーザスレッドに提供している．実験に用いた計算機，ディスクの性能をそれぞれ表 2 と表 3 に示す．また，プリエンプションポイントは，256 セクタアクセスごとに設定し，転送モードとして UltraDMA mode 6(100MByte/sec)，セクタアドレッシング方式として LBA(Logical Block Addressing) 方式を用いている．

表 2　実験に用いた計算機の性能

Machine	PC/AT 互換機
CPU	Intel PentiumIII 900MHz
Memory	128MB
IDE Controller	Intel 82801BA(ICH2) ATA/ATAPI-6

表 3　実験に用いたディスクの性能

Model name	IBM IC35L060AVV207-0
Interface	ATA/ATAPI-6
Capacity	60GByte
Sector size	512Byte
Rotation speed	7200rpm
Average seek time	8.8msec
Cache	2MByte

4.2 最大横取り禁止時間の計測

4.1 節に示した評価環境における最大横取り禁止時間の計測を行う．ディスクアームの位置を最外周にセットして，転送速度の最も遅い最内周の 256 セクタの読出しを 100 回行なう．このとき，最もディスクの利用時間の長いものが最大横取り禁止時間となる．実験の結果，30msec であった．すなわち，この実験環境での最大横取り禁止時間は，30msec となる．

4.3 デッドラインミスの回数

4.2 節の最大横取り禁止時間をふまえて，ディスク I/O 要求セットを作成する．作成したディスク I/O 要求セットのパラメータを表 4 に示す．これらのディスク I/O 要求セットを用いて，FCFS(First Come First Service) スケジューリング方式および，優先度付き EDF スケジューリング方式を用いて実際に動作させた．開始から 1.5 秒間のタイムチャートを図 6 に示す．さらに，開始から 30 秒間のディ

表 4　ディスク I/O 要求セット

	Request 1	Request 2	Request 3
Cycle	60msec	500msec	1400msec
Worst-case access time	30msec	100msec	400msec
Release time	100msec	100msec	100msec
Priority	64	64	64

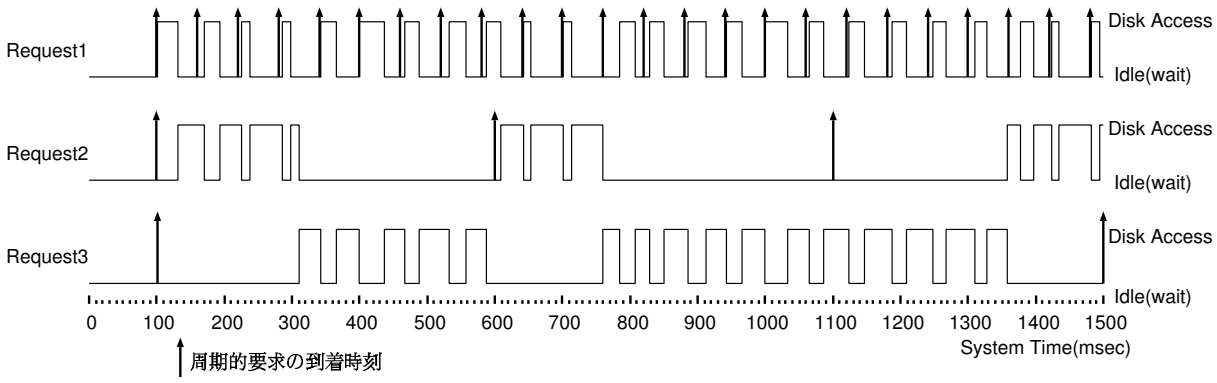

図 6　本機構によるディスク I/O 要求処理のタイムチャート

表 5　デッドラインミスの回数

	Request 1	Request 2	Request 3
FCFS	225	1	0
本機構	0	0	0

スク I/O 要求のデッドラインミスの回数を表 5 に示す.

FCFS スケジューリングでは, 要求の到着順に処理を行い, 横取りが発生しないため, 当該要求のディスク利用時間が他の要求の周期よりも長い場合, 確実にデッドラインミスが発生する. 実験では, Request2, Request3 のディスク利用時間が Request1 の周期よりも長いため, Request1 に関して大量のデッドラインミスが発生している. また, このディスク I/O 要求セットは, 優先度がすべて同じであるため, ディスク I/O 要求を優先度順に処理しても結果は同じである. これに対し本機構, すなわち優先度付き EDF スケジューリング方式では, ディスク I/O 要求の持つ時間的制約を考慮しているため, ディスクがアクセス中である割合が 98%という高負荷の状況下においてもデッドラインミスが発生していない.

4.4　横取り禁止時間の短縮

本評価環境と全く同じハードウェア構成で, 最大横取り禁止時間を短縮する手法として, 短い間隔でプリエンプションポイントを設定する手法が挙げられる. 本機構では, ATA の規格で定められている 256 セクタアクセスをひとつの区切りとし, プリエンプションポイントを設定した. この場合, 最大横取り禁止時間は, 4.2 節の計測結果より 30msec となる. プリエンプションポイントを設定するセクタ数と最大横取り禁止時間および, 256 セクタの読出しにかかる時間を示したグラフをそれぞれ図 7 と図 8 に示す. プリエンプションポイントを 16 セクタアクセスごとに設定した場合, 最大横取り禁止時間は

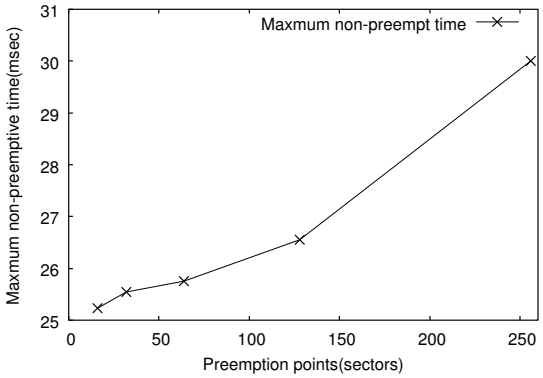

図 7　最大横取り禁止時間

25.2msec となり, 4.8msec の短縮が可能となる (図 7). しかし, プリエンプションポイントを 16 セクタアクセスごとに設定した場合, 256 セクタアクセスにかかる時間は 167.6msec となる (図 8). すなわち, 256 セクタアクセスごとに設定した場合に比べ 5 倍以上の時間が必要となり, ディスクの利用効率が著しく低下する. 図 8 より, プリエンプションポイントを設定する間隔を短くすると, ディスクの利用効率が指数関数的に低下することがわかる. これらの結果から, 性能向上のための手法として, 優先度に応じてプリエンプションポイントの設定数を変更することが考えられる. 優先度の高いマルチメディアアプリケーションからの要求には最大の 256 セクタアクセスを割り当て, 通常のアプリケーションからの要求にはそれよりも短い間隔を割り当てる. これにより, さらにスケジューラビリティの向上を図ることが可能となる.

5　関連研究

従来のシステムでは, ディスクへの I/O 要求は, デバイスキューに繋がれ, ディスクアーム動作が最小になるように整列され処理される. Linux[4] など

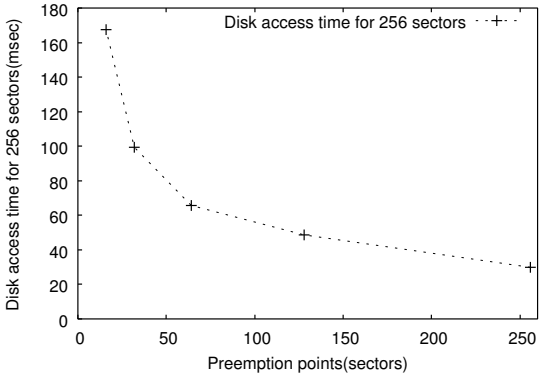

図 8　ディスクの転送効率

の汎用的な OS では，このような手法が用いられ，システム全体としてディスクの利用効率を高めている．しかし，プロセスの優先度を考えた場合，優先度の高いプロセスからの I/O 要求が後回しにされる可能性が発生し，I/O 要求が増えるにしたがって，その可能性は高くなる．すなわち，この手法を用いる場合，I/O 要求を処理する時間を特定することができず，十分なリアルタイム性を得ることは難しい．

また，リアルタイムスケジューリングのための I/O アクセス制御手法として，さまざまな研究が行われている [5]．これらの手法では，高優先度の周期プロセスが，ディスクを利用する 1 周期以上前にディスクを利用することを宣言し，そのディスク利用要求の即時利用を妨げるディスクアクセスを制限している．この手法により，ディスクを利用する最高優先度のプロセスが発行するディスク I/O 要求を，高々 1 回のブロッキングのみに抑えることが可能となる．しかし，ディスクの利用前に必ずその宣言が必要であり，突発的なアクセスを制御することは難しい．さらに，アクセスを制限することから，ディスクの利用率が低下する．

本手法では，ブロッキングの抑止は実現していないが，最大横取り禁止時間をアプリケーション設計時に予測することが可能であるため，同等の利点を得ることが可能である．さらに，横取り可能なリアルタイムスケジューリングにより，動的なディスクアクセスの対応が可能となり，本手法が，より広い範囲へ適用可能であるといえる．また，4.3 節の評価により，ディスクがアクセス中である割合が 98%という高負荷の状況下においても，全ての要求のデッドラインを満たすことが可能であることを示した．

6　おわりに

本論文では，リアルタイム環境に適用可能な Ultra DMA 転送機構を提案し，その特徴，構成および評価について述べた．従来のシステムでは，横取り不可能な処理として考えられていた DMA 転送によるディスク I/O 処理に対して，プリエンプションポイントを設定して横取り可能であると考えることにより，横取りを可能とするリアルタイムスケジューリングを適用することが可能となる．本機構では，その一例として，各プロセスの I/O 要求に対し，プロセスの持つ優先度および時間的制約をスケジューリングパラメータとして優先度付き EDF スケジューリング方式を実装し，評価を行った．その結果，最大横取り禁止時間を計測し，それをふまえたディスク I/O 要求セットであれば，ディスクがアクセス中である割合が 98%という高負荷の状況下においても，それらの時間的制約を満たすことが可能であることを示した．さらに，動画や音声のデコード処理のような周期的にデータを必要とするスレッドは，読出し専用のリアルタイムスレッドを作成する必要はなく，読出し要求にリアルタイムパラメータを設定するだけで，周期的なデータの供給を受けることが可能となる．

参考文献

[1] Peter T. McLean: AT Attachment with Packet Interface-6 (ATA/ATAPI-6), http://www.t13.org/ (2002).

[2] C. L. Liu and J. W. Layland: Scheduling algorithms for multiprogramming in a hard real time environment, J. of ACM, Vol.20, No.1, pp.46-61(1973).

[3] 谷出 新，芝 公仁，大久保 英嗣: リアルタイムオペレーティングシステム Easel におけるメモリ管理機構, 情報処理学会研究報告 2001-OS-86, Vol.2001, No.21, pp. 91-98 (2001).

[4] Daniel P. Bovet, Marco Cesati: 詳解 Linux カーネル, 株式会社オライリー・ジャパン (2001).

[5] 帆波 幸二，毛利 公一，吉澤 康文: リアルタイムスケジューリングのための I/O アクセス制御, 情報処理学会研究報告 2001-OS-88, Vol.2001, No.78, pp. 91-98 (2001).