システムソフトウェアと
オペレーティング・システム 88－13
（２００１．７．２７）

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

帆波 幸二† 毛利 公一†† 吉澤 康文††
†東京農工大学大学院工学研究科
††東京農工大学工学部

連続メディアを処理するプロセスのリアルタイム性を保証するためには，プロセスに対して，CPU資源だけでなくI/O資源もリアルタイムに割り当てる必要がある．しかし，従来のI/Oのスケジューリング方式では，他のプロセスによってリアルタイム性を保証しなければならないプロセスをブロックしてしまう場合がある．これを解決するために，本論文では，優先度の高いプロセスが，優先度の低いプロセスによってI/Oをブロックされる時間を短くするための機構を提案する．本論文では，特に，ハードディスクを対象とした機構について述べる．ブロック時間を短くするためには，プロセスの優先度に基づいてI/O操作のタイミングを調整する必要がある．本機構では，リアルタイムプロセスがデバイスを利用する間は優先度に基づいてアクセスを制限する手法を用いた．本機構の有効性を示すために，Linux上での動画再生を例として，I/O要求を発行した場合のブロック時間を測定した．その結果，優先度順にI/O要求を処理した場合よりもさらに29.5%が本機構によって短縮できるという試算を得た．

Disk Access Control Mechanism for Real-Time Scheduling

Koji HONAMI† Koichi MOURI†† Yasufumi YOSHIZAWA††
†Graduate School of Engineering, Tokyo University of Agriculture and Technology
††Faculty of Engineering, Tokyo University of Agriculture and Technology

In order to satisfy timing constraints of continuous media, Operating system must allocate timely not only CPU resources but I/O resources. In existing scheduling systems of I/O operation, it cannot avoid blocking problems. In order to solve this, we present a new mechanism that can reduce blocking time. Especially we describe a mechanism for a hard disk in this paper. Setting access restrictions each device while real-time processes use it, we control timing of I/O operations, and solve blocking problems. Finally, we carry out estimates of got effect by proposed mechanism on the Linux. In this estimates, blocking time decrease 29.5% rather than Existing priority based I/O scheduling system.

1. はじめに

リアルタイム処理においては，その時間制約を満たすため，プロセスの要求する計算機資源を適時的に割り当てることが必要不可欠である．リアルタイムスケジューリングは，その重要な役割を担っている．しかし，スケジューラによってCPUが割り当てられた場合においても，データの読み書きをリアルタイムに行うことが

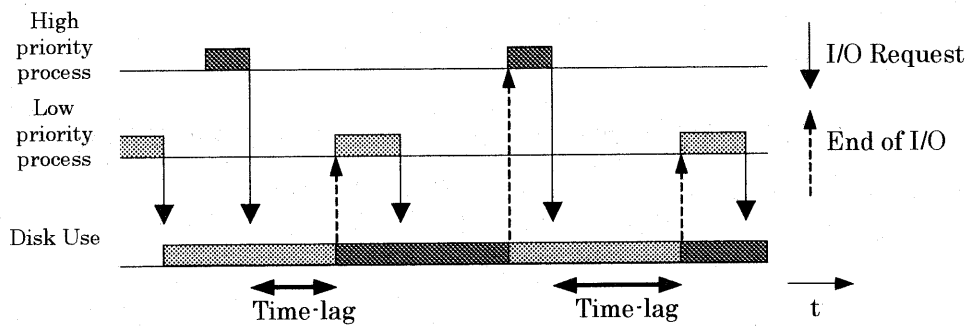

図 1 I/O スケジューリングでは防げないタイムラグ

できなければ，プロセスは，その時間制約を満たすことができない．特に，ディスク I/O においては，必ず I/O 要求を発行すれば即座にデバイスを利用できるというわけではない．他のプロセスがデバイスを使用している場合は，それが終了するまで待たなければならない．この要求発行から，実際のデバイスアクセス開始までのタイムラグは，リアルタイムプロセスの処理時間の予測を困難にし，デッドラインミスを誘発する要因になる．

タイムラグの長さは，採用している I/O スケジューリング方式で大きく変わる．汎用的な OS では，デバイスキューに繋がれた I/O 要求は，セクタ順に整列され処理される．これは，シーク時間を減らして，全体としての効率を上げるという効果がある．しかし，この方式の場合，優先度の高いプロセスでも I/O 操作が後回しにされる可能性があり．要求が増えるに従って後回しにされる確率が高くなる．よって，この方法では，I/O 操作にかかる時間をバウンドすることができず，リアルタイム性を得ることができない．

一方，プロセスの優先度順に I/O を処理した場合は，要求が多くなった場合でも，優先度の高いプロセスの I/O は先に処理され，上記のように後回しにされる心配はない．しかし，I/O 要求が発行された時点でデバイスが利用中であれば，その完了を待つ必要がある．デバイスアクセスが開始された直後に発行された I/O 要求において，この待ち時間は最長になり，I/O 操作 1 回分に相当する時間を待たねばならない．このように，I/O スケジューリングだけでは I/O 要求発行からアクセス開始までにタイムラグが生じることを防ぐことはできない．また，アクセスの度にこのタイムラグは生じる可能性がある(図 1 参照)．

本論文では，上記のように，I/O スケジューリングだけでは回避できないタイムラグを防ぐため，デバイスにアクセス制限を設けるという新しい機構を提案する．本機構は，連続メディア処理のような周期的プロセスを対象とし，かつ，スケジューリングアルゴリズムが固定優先度割付である場合に適用可能である．このようなシステムにおいては，タイムラグの発生を周期実行全体で高々1 回に抑えることができる．

以下，本論文では，2 章でデバイスを即時利用するための理論を示し，3 章でアクセス制御機構の構成，および処理内容について述べる．4 章では本機構による効果の見積もりを示し，5 章でまとめについて述べる．

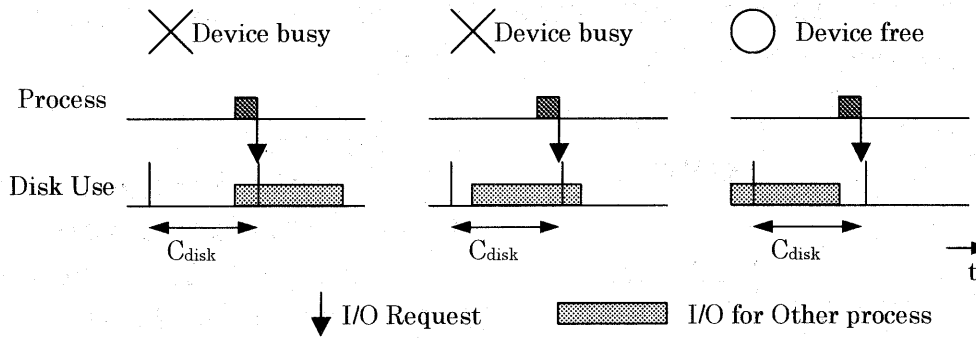

図 2　I/O 要求発行時のデバイス利用状態の違い

2.　アクセス制御の理論

I/O 要求発行時に即座にデバイスにアクセス可能とするためには，要求発行前からそのデバイスが利用されるのを防がなければならない．具体的には，優先度順に I/O スケジューリングがなされている場合，I/O 要求発行から 1 回分のディスク利用時間(以下 C_{disk} と示す)だけ前までの範囲で発行されるディスクアクセスは，即時利用を妨げる危険性があるので(図 2 参照)，そのアクセスを制限する必要がある．事前にどのデバイスがいつ利用されるかということを予測することは非常に困難である．そこで，連続メディア処理プロセスの周期性を利用する．

プロセスの周期駆動を OS がサポートするのであれば，次周期の処理の開始時刻を OS は把握している．処理開始時刻よりも C_{disk} だけ前からアクセスを制限しておけば，プロセスのどのタイミングにおいても，デバイスは未使用で即座に利用可能な状態となる(図 3 参照)．

また，プロセスは毎周期に必ずデバイスを利用するとは限らない．たとえば動画再生なら，何フレーム分かまとめてディスクからユーザバッファに読み込んでおき，バッファの中から 1 周期に 1 フレーム分ずつデータをとりだして処理を行うということも考えられる．このようにデバイスを利用しない周期も存在する可能性があるため，毎回アクセス制限をかけるのではなく，プロセスから要望があったときにのみアクセス制限をかけるということにする．このときプロセスは少なくともディスクを利用する 1 周期以上前にこの要望を出す必用がある．一方，デバイスの解放は周期内ですべての I/O 操作が完了した後に行う．こちらもそのことをプロセスの側から通知する．このようにプロセスの側からの通知と，周期性を利用することで，必要なときに必要なだけデバイスにアクセス制限がかかるようにする．

ちなみに本方式では，最初の周期でデバイスを利用する場合，その利用を前もって宣言することが不可能である．このため，完全にブロッキングを防ぐことはできず，周期的プロセスの処理全体として，高々一回のブロッキングは発生することになる．

デバイスをロックするのではなくアクセスを制限するとしたのは，より高優先度のプロセスによるデバイスアクセスを妨げてはならないからである．また，制限する区間が重なり合う状況発生した場合は，重なり合った期間は優先度の高い方のアクセス制限を採用し，優先度の高いプロセスの I/O を妨げないようにする．

以上に示したようなデバイスのアクセスの制限を行うことで，デバイスはそれを利用するプ

ロセスのうち最高優先度のものが常に即座に使用できる状態を作り出すことが可能になる．

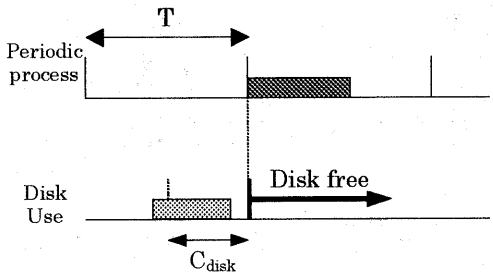

図 3 次周期開始の C_{disk} 前から保護した様子

3. アクセス制御機構の構成

3.1. アクセス制御機構の概要

本機構は，2 章の理論に基づき，プロセスのデバイス利用宣言をもとに，周期開始時刻の Cdisk だけ前からアクセス制限をかける機能を提供するためのものである．図 4 にその全体構成を示す．本機構は次に示す各要素から構成される．

・デバイスキュー
・デバイス予約リスト
・I/O スケジューラ
・システムコールインタフェース

以下，それぞれの構成要素について述べる．なお，プロセスの周期駆動は実現されており，OS はプロセスの処理開始時刻を把握しているとする．

3.2. デバイスキュー

デバイスキューは，read, write システムコールなどで発行された I/O 要求を実際にデバイスで処理するまでの間，一時的に保持しておく領域である．キューは優先度順にソートされ，常に先頭(最高優先度)の I/O 要求から処理される．優先度が同じである場合は，プロセススケジューラで採用しているアルゴリズムによって，FIFO，セクタ順など都合のよう方法で順番を決定する．

3.3. 予約リスト

予約リストはデバイスの利用制限を求めているプロセスの情報がつながれる領域である．予約リストはアクティブ予約リスト，レディ予約リストの二つのキューから構成される．アクティブ予約リストは，現在，デバイスの利用を制限している最中のプロセスのリストであり，デバイスの利用を制限するために使われる．このリストは優先度順に整列され，先頭には常に制限をかけているプロセスの中で最高優先度のものが位置するようにする．レディ予約リストはアクセスを制限する予定であるが，まだその時刻には至っていないもののリストである．こちらのリストにはアクセス制限をかける時刻の情報が付加され，制限をかける時刻の早い順に整列される．アクセス制限をかける時刻となったものはアクティブ予約リストに移される．

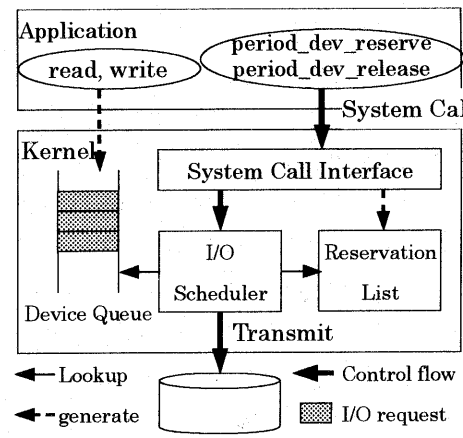

図 4 アクセス制御機構の構成

3.4. I/O スケジューラ

I/O スケジューラは，デバイスキューにペンディング中の I/O 要求から，次に処理するもの

を選択し，デバイスに処理を割り当てる役目を担う．予約リストによりアクセスの制限をかける必要があるため，予約キューの更新作業も行う．処理の流れは次のようになる．

(1) 予約キューの更新
レディ予約リストからアクセス制限をかける時刻となったものをアクティブ予約リストに移動

(2) I/O 要求の選択
次に処理する I/O 要求を選択する．つまり，デバイスキューの先頭を選択する．

(3) アクセス許可のチェック
(選択した I/O 要求の優先度) ≧
(アクティブ予約リストの先頭の優先度)
となればアクセスを許可する

(4) I/O 処理開始
許可された場合その I/O 要求の処理を開始する．そうでなければ何もしない．

I/O 処理を開始したか否かで，次に I/O スケジューラが起動するタイミングが異なる．I/O が開始された場合は，次の起動ポイントはその I/O 操作の完了した時点である．開始されなかった場合は，新しい I/O 要求が発行された，もしくは，予約リストから予約が解消された時点で起動される．後者は，新しくアクセスが許可された I/O 要求ができる可能性がある時点で I/O スケジューラを起動するということである．

3.5. システムコールインタフェース

本機構では，次に示す二つのシステムコールインタフェースを提供する．

(a) デバイス利用の宣言
アクセス制限をかける時刻を算出し，プロセス情報と併せてエントリを作成，予約リストに登録を行う．

(b) デバイス利用宣言の解除
予約リストから当該プロセスのエントリを削除し，必要があれば I/O スケジューラの起動を行う．

これらのシステムコールの書式を以下に示す．また，本機構はプロセスの周期駆動も OS がサポートするため，周期的プロセスモデル[1]を利用している．そのための関数も併せて示す．

(1) 周期実行開始
period_start(T, L)
プロセスを周期的プロセスに移行させる．引数には，周期長と周期内の最悪実行時間を ms 単位で指定する．成功すれば 0，エラーであれば負数を返す．

(2) 周期実行終了
period_end()
周期的プロセスとしての動作を終了し，通常の状態へ戻す．引数はない．0 ならば成功，負数であれば周期実行中でないことを示す．

(3) 次周期待ち
period_skip()
次の周期の開始時刻までスリープする．プロセスは次周期開始時刻にスケジューラによりウェイクアップされる．当システムコールは周期内の処理が終わった時点で発行する．0 ならば成功，負数ならば周期実行中でないことを示す．

(4) デバイス予約

period_reserve_device(DEV, Cdisk)

次周期でデバイスを利用することを宣言する．当システムコールを発行することにより，次周期開始時刻のCdiskだけ前からデバイスにアクセス制限がかかる．これにより次周期でのディスクアクセスでブロッキングが発生しないことが保証される．引数には，デバイス識別子と，1回のディスクアクセスにかかる時間(ms)を指定する．成功すれば0，失敗すれば負数を返す．

(5) デバイス解放

period_release_device(DEV)

period_reserve_device()で確保したデバイスを解放する．引数としてデバイス識別子を指定する．当システムコールを発行しなくてもプロセスが非アクティブ状態になった時点でデバイスは解放される．成功すれば0，失敗すれば負数を返す．

3.6. プログラミングモデル

上記システムコールを用いて周期的プロセスを記述する場合，メインループ付近のコードは図5のようになる．

4. 性能評価

4.1. 評価方法

本機構は現在実装途中でまだ完成はしていない．そこで，連続メディア処理におけるI/O操作時間の測定実験をLinux上で行い，その結果から今回提案した方式でどれだけの効果が得られるのか見積もりを行った．測定実験はLinuxのX-Window上で行い，カリフォルニア大学で開発されたMPEG1再生ソフトmpeg_playによる動画再生を連続メディア処理の例として使用する．測定環境は表1の通りである．

```
need_disk_read = TRUE;          /* ディスク読み込みフラグをたてる */
period_start(T, L);             /* 周期実行開始 */
while ( !end ) {
  if ( need_disk_read ) {
    read( ... );                /* ディスクからバッファにデータを読み込む */
    period_rerease_device( DEV );       /* デバイスの保護要求を解消 */
    need_disk_read = FALSE;
  }

  ┌──────────────┐
  │ 1フレーム分の処理 │
  └──────────────┘

  if ( 次フレームのデータがバッファにない ) {
    need_disk_read = TRUE;
    period_reserve_device( DEV, Cdisk );   /* 次周期に利用するデバイスの保護要求 */
  }
  period_skip();            /* 次周期開始時刻までスリープ */
}
period_end();               /* 周期実行終了 */
```

図5 周期的プロセスのメインループ記述例

表 1　測定環境

Machine	PC/AT 互換機
CPU	Celeron 500MHz
Memory	128Mbyte
HDD	FUJITSU MPE3084AE
OS	Linux Kernel 2.2.13
Driver	Ide-disk version 1.0.9

ちなみに，Linux カーネルは次に示す実験のため，優先度順の I/O スケジューリングも可能なように変更を施してある．

測定内容は，動画再生プロセスがディスク読み込みを行った際にどれだけの時間 I/O 完了待ち状態になっているかである．これを次に示す二つの I/O スケジューリングの方式で測定する．

(a) 通常の Linux の I/O 操作(キューにたまった I/O 要求はセクタ順に処理)による場合

(b) I/O 要求をプロセスの優先度順に処理する場合

それぞれのケースにおいて，動画データの存在するディスク上の別ファイルの順次読み込みを行う負荷プロセスを並行動作させ，その負荷プロセス数によりどのように I/O 時間が変化するかも測定する．負荷プロセスの優先度は動画再生プロセスよりも低く設定する．バッファにデータがありディスクアクセスが行われない場合については測定対象から除外する．

4.2. 計測結果

1 分間で発生したディスク読み込みにおいて発生した I/O 完了待ち時間の平均時間を測定したところ，図 6 のような結果を得た．グラフでは先に述べた(a)，(b)における実験結果に加え，ディスクアクセス自体にかかる処理時間を(c)として示している．この値は負荷をかけている場合でのディスクアクセスのみにかかった平均の時間である．(c)を示したのは，この値が本機構をこの実験に適用したときに得られる理想の値であるからである．

(a)では，負荷プロセスの増加にともない，I/O 時間も増加している．これは，動画再生プロセスをリアルタイムスケジューリングしたとしても，I/O 操作においてプロセスの優先度を考慮しなければ，時間制約を保証することはできないことを示している．一方(b)では，若干負荷プロセス数の増加によって待ち状態の時間も長くなっているようだが，(a)のように顕著ではない．これは I/O 要求が増えても，再生プロセスの I/O が優先されるからである．時間の増加はオーバヘッド増加と考えて良いだろう．(c)本機構を適用したときの理想値で，(b)よりも平均して 29.5%ほど短い．

本実験により，提案した機構を用いることで(b)の優先度順に I/O スケジューリングを行った場合よりもディスク読み込み時の待ち時間は 29.5%削減できる見込みがあることが判明した．以上より，本機構によるデバイスのアクセス制御がどの程度有効であるのかが示された．

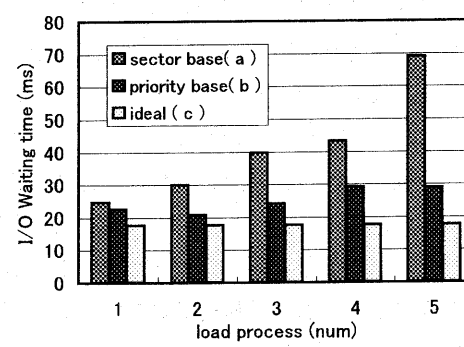

図 6 負荷プロセス数と I/O 時間

5. おわりに

本論文では，連続メディア処理に有効となるディスクアクセス制御機構を提案し，その特徴，

構成および評価について述べた．本機構は，周期的プロセスが前もってディスクアクセスをすることを宣言することで，ディスクアクセスがより優先度の低いプロセスのディスク利用によって妨げられることを防ぐことを可能とするものである．これにより，優先度の高いプロセスほど，入出力操作の要求から開始まで待ち時間を短くすることが可能になる．また，これに伴いリアルタイムスケジューリングにおけるスケジューラビリティの向上も見込むことができる．

さらに，Linux 上での動画再生を例として，入出力操作にかかる時間を測定した．その結果，デバイスキューの I/O 要求を優先度順に処理する I/O スケジューリングを採用している場合に比べて，動画再生プロセスの入出力操作待ち時間を 29.5%削減可能であるという試算を得て，提案する機構の有効性が確認された．

今回提案した機構は，連続メディア処理のような周期的プロセスを対象としており，かつスケジューリングアルゴリズムが固定優先度割り当てによるものである場合に適用可能である．この条件の下では，周期的プロセス全体でブロッキングを高々1 回に抑えることを可能にする．これはリアルタイムプロセスのスケジューラビリティ向上につながる．

今後の展望としては，本機構を Earliest Deadline First [2] アルゴリズムにまで適用できるように拡張することが考えられる．EDF アルゴリズムは，動的優先度割り当てであるが，時間の経過に伴う優先度の変化は全プロセスに一律で，デバイス保護の途中でプロセス間の優先関係が逆転するというようなことがない．このため本機構を適用できる可能性が高い．

参考文献

[1] 鈴木 貴志, 吉澤 康文, “周期駆動機能を持つリアルタイムスケジューラの開発”, 情報処理学会第 58 回全国大会論文集, pp. 61-62, May 9 1999.

[2] C. L. Liu and J. W. Layland: Scheduling Algorithm for multiprogramming a Hard Real Time Environment, J. of ACM, Vol. 20, No. 1, pp. 46-61 (1973)