リアルタイムUNIXのバッファキャッシュ管理方式

5C－2

菅井 尚人　落合 真一
三菱電機(株) 情報技術総合研究所

1　はじめに

標準化やオープン化要求の高まりによって、産業用制御計算機においてもUNIXをベースとしたリアルタイムOSが広く使用されている。これらは、カーネル内プリエンプションや優先度に基づいたスケジューリングを行なうことによりリアルタイム性を確保している。また、ファイルシステムについても、連続領域割り当てファイルシステムなど、リアルタイム用途への適用を考慮したものをサポートしているが、バッファキャッシュ機構については、リアルタイム性の配慮がなされていない。

本稿では、リアルタイムシステムでの利用を考慮し、プロセスの優先度をバッファキャッシュ管理にも拡張するリアルタイムUNIXのバッファキャッシュ管理方式について提案する。

2　既存方式の問題点

UNIXのバッファキャッシュ機構を、リアルタイムシステムで使用する場合の主な問題として、以下のことが考えられる。

1. バッファの新規割り当て時の問題

   低優先度のプロセスによってバッファが枯渇してしまい、高優先度のプロセスがバッファを使用しようとする時に、バッファの割り当てまでに長い時間を要する可能性がある。

2. sync時の問題

   syncは、遅延書き込みの状態にある全てのバッファに対して、ディスクへの書き込みを起動する処理を行なうが、ディスクへの書き込みが完了するまでの間、他のプロセスがそのバッファを使用することはできない。ディスクへの書き込みの完了までの時間は、書き出しの順番によって大きく変わるため、高優先度プロセスが使用しようとしているバッファが、優先度の低いプロセスに使用されているバッファよりも、長時間使用できなくなる可能性がある。

3. 遅延書き込みの問題

   一般に、読み込み処理はI/Oネックとなるが、書き込み処理は遅延書き込みの場合、CPUバウンドの処理となるため、結果として書き込み処理が優先されることになる。このため、バッファは遅延書き込みの状態にあるものが多数となり、ディスクへの書き出しのためバッファの獲得に長い時間を要したり、sync時の問題を引き起こす可能性が高まる。

4. バッファ量の問題

   バッファ量が多ければ、それだけキャッシュとしての効果が期待できるが、その一方、前述の遅延書き込みの問題の影響は大きくなる。

3　優先度を考慮したバッファキャッシュ管理

前節で述べた問題点のうち1および2に対処する機能を実装するため、リアルタイムUNIX "moose"[1]のファイルシステムモジュールとして、「優先度バッファキャッシュファイルシステム」を作成した。(図1)

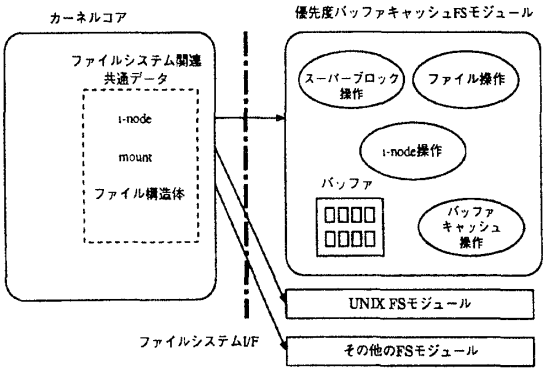

図1: 優先度バッファキャッシュFS

このファイルシステムは、バッファキャッシュ管理の部分を除き、既存のUNIXファイルシステムと同等の機能を持ち、ディスク上のデータも

Buffer Cache Management for Real-Time UNIX System
Naoto SUGAI, Shinichi OCHIAI
Mitsubishi Electric Corporation

互換性が保たれている。以下、本ファイルシステムに実装したバッファキャッシュ管理の機能について述べていく。

3.1 バッファの優先度別分割

低優先度プロセスによるバッファ占有の問題を回避するため、バッファを複数のグループに分割し、それぞれのグループのバッファは、一定優先度以上のプロセスに対してのみ割り当てられるようにする。

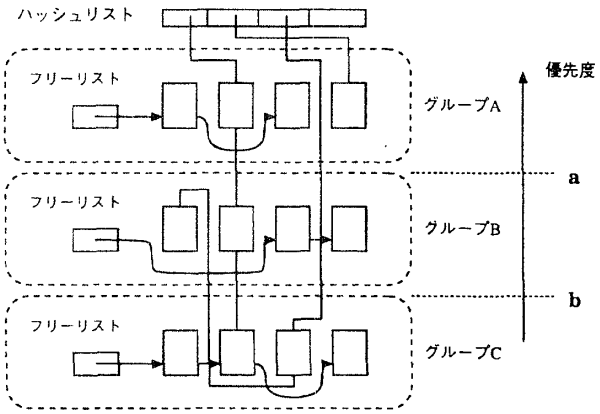

図 2: 優先度別バッファ

図2は3つのグループに分割した場合の例を示している。

バッファは、初期化時に各グループに分けられ、それぞれのグループのフリーリストにつながれる。バッファを獲得しようとする場合、そのプロセスの優先度によってバッファを獲得するフリーリストを選択する。プロセスの優先度がbより低い時は、グループCのフリーリストからバッファを割り当てる。優先度がaより低い時は、グループBのフリーリストから割り当て、もし、グループBのフリーリストが空である場合は、グループCのバッファを割り当てる。優先度がa以上であれば、グループAのフリーリストから順に利用可能なバッファを捜していく。

デバイスのハッシュリストは全バッファで共通である。このため、優先度の異なるプロセスでバッファを共用できるが、この時発生する問題については後述する。

使用が済んだバッファは、そのバッファのグループのフリーリストに戻される。

3.2 sync 時の優先度順書き出し

sync 時の問題は、高い優先度のプロセスによって使用されているバッファを先にディスクへ書き出すことで回避できる。しかし、個別のバッファについてその順番を決定していくのは、処理のオーバーヘッドが大きいため、優先度別のバッファグループを利用し、高い優先度グループのバッファからグループ毎にディスクへの書き出しを行なう。これにより、高優先度プロセスが使用しようとするバッファが、低優先度プロセスのバッファよりも長時間使用できなくなることを防ぐと共に、使用できない時間を短縮する。

4 検討課題

遅延書き込みの問題と、適正なバッファ量の問題については、検討を継続している。

バッファの優先度別分割を行なった場合、グループ間でのバッファ使用の不均衡により、システム全体としてのバッファキャッシュの利用効率が低下するおそれがあるため、対策を検討する必要がある。

また、現状では、バッファを複数のプロセスで共有する場合、そのバッファを最初に獲得したプロセスの優先度によって、バッファのグループが決定してしまう。このため、低優先度プロセスが使用していたバッファを、その後に高優先度プロセスが使用した場合、sync 時の優先度順書き出しが正しく行なわれない。異なる優先度のプロセスで共有されるバッファの扱いについて検討する必要がある。

5 おわりに

リアルタイム UNIX において、プロセス優先度をバッファキャッシュ管理にも拡張した方式について述べた。バッファキャッシュ機構をリアルタイムシステムで利用する場合の問題に対処するための方法として、バッファの優先度別分割、sync 時の優先度順書き出しを提案した。

現在、これらの実装をほぼ完了し、各機能の効果を評価中である。

今後、前節で述べた未解決の課題について検討すると共に、アプリケーションの実行モデルを作成し、本方式の有効性を検証していく。

参考文献

[1] 落合、菅井:「リアルタイム UNIX のモジュラー OS 化検討」、情報処理学会第 51 回全国大会、1997