システムソフトウェアと
オペレーティング・システム 69－9
（１９９５．６．２３）

リソース予約システムを用いた適応的な連続メディア処理

中島 達夫

北陸先端科学技術大学院大学

連続メディアアプリケーションのサポートは今後のオペレーティングシステムにとって必要不可欠なものとなると考えられる。本稿では、リソースの予約システムとメディアスケーリング可能な連続メディアオブジェクトの組合せた、マシンの性能に非依存な処理を実現する適応的な連続メディア処理を実現するための方式を提案する。

Adaptive Continuous Media Processing based on Resource Reservation

Tatsuo Nakajima

Japan Advanced Institute of Science and Technology

Supporting continuous media applications will be important in future operating systems. In this paper, we propose a methodology which integrate resource reservations and scalable continuous media objects to process continuous media flexiblly.

1 はじめに

ビデオやオーディオなどの連続メディアオブジェクトの扱いは、高速CPU、大容量メモリ、データの圧縮技術などの進歩にともない、従来のワークステーションやパーソナルコンピュータ上でも実行が可能となってきているが、UNIX[1]等の従来のオペレーティング・システム上で、分散マルチメディアアプリケーションを実行すると、タイムシェアリングの概念に基づいたラウンドロビンスケジュールやFIFOを用いたリソース管理による処理や通信の予測不可能な遅れにより、ビデオやオーディオの処理を正しく行なうことが困難となることが生じていた。例えば、マルチメディアアプリケーションと同時に、ファイル転送を行なうと、ファイル転送のためのパケット処理のため、連続メディアオブジェクトの転送が遅らされたり、バッファがあふれるため、連続メディアオブジェクトのパケットが破棄されてしまうことが起きる。

従来のオペレーティングシステムの提供する資源管理ポリシは各アプリケーションにCPUやメモリなどの資源を公平に分割するため、連続メディア処理のように時間に依存した処理を行なうために適していない。また、ファイルシステムやネットワークシステムのようなオペレーティングシステムサービスを変更したり、新しく追加することは困難であった。マイクロカーネル技術では、マイクロカーネルと呼ばれる部分は様々なオペレーティングシステムの共通項のみ提供し、その他の部分はユーザレベルのサーバとして実現される。そのため、複数のオペレーティングシステムインタフェースを提供したり、アプリケーションのために最適化したオペレーティングシステムを提供することが可能になる。

また、遠隔会議システムやマルチメディアプレゼンテーションなどの連続メディアアプリケーションは、ネットワークに結合された様々な性能のワークステーションにより実行される。そのようなシステムでは、ワークステーションの負荷や能力をはじめから予測することが困難である。そのため、アプリケーションを実行するために必要なマシンの性能や負荷がそのアプリケーションを実行するための必要条件となってしまい、連続メディアアプリケーションの実行をワークステーションの性能に合わせてスケーラブルに調整できない。この問題を解決するためには、システムの負荷や能力に合わせてスケーラブルに連続メディア処理の品質を変化することを可能とする連続メディアの適応的な処理が必要となってくる。つまり、システムの負荷が重い場合は、表示するメディアの品質を落して、連続メディアオブジェクトの時間制約を満足するようにする必要がある。

現在、北陸先端科学技術大学院大学の我々のグループでは、慶應義塾大学、カーネギメロン大学との協力で、このような連続メディアを提供するための基盤となるマイクロカーネル技術に基づいたオペレーティングシステムReal-Time Mach[3]を開発中である。我々のグループでは、Real-Time Machを連続メディア処理を可能とするための資源予約システム、Real-Time Mach上の連続メディア処理に適したネットワークシステム、ストレージシステム、連続メディア用ツールキット、連続メディア・アプリケーション[2]を構築している。

以下の節では、適応的な連続メディア処理について述べた後、Real-Time Machが提供している連続メディアのためのOSサポートに関して述べる。最後に、適応的な連続メディア処理のReal-Time Mach上の実現に関して述べる。

2 サービスの質とメディアスケーリング

本節では、連続メディアオブジェクトの品質を変更可能にするメディアスケーリングに関して述べる。最後に、メディアスケーリング可能な連続メディアオブジェクトと適応的な連続メディア処理の関係に関して述べる。

2.1 メディアスケーリング

従来の連続メディアオブジェクトの表現では、メディアの品質は連続メディアオブジェクトが静的にもつ属性であり、動的に変化することができないものと考えられていた。この場合、オペレーティングシステムに渡されるQOSは、処理する連続メディアオブジェクトにより決定され、ユーザが指定できるものではなかった。そのため、メディアの品質を変更したい場合には、複数の連続メディアオブジェクトを用意する必要があった。しかし、多くのビデオやオーディオなどのデジタル表現を考えると必ずしも連続メディアオブジェクトの品質は静的なものでないと考えることができる。

このような品質を動的に変更することが可能な連続メディアオブジェクトを、メディアスケーリング（Media Scaling）可能な連続メディアオブジェクトと呼ぶ。連続メディアオブジェクトの品質を変化させるためのパラメータとして、いくつかの手段を考えることができる。例えば、ビデオのフレームレートや圧縮率、ピクセル毎のビット数、オーディオのサンプルあたりのビット数、ステレオ／モノラルの変換などである[1]。

簡単な例としては、ビデオストリームの適当なフレームを間引く（例えば、2フレームのうちの1フレーム）ことによりメディアの品質を変化できる。また、各フレームを4ビット毎のブロックに分割し、そのうちの左上の1ビットと残りの3ビットを分割することにより、すべてのブロックの1ビットだけ用いれば品質の低いフレームを構成できる。あるいは、フレームの白黒部分と、カラー部分を分けたり、オーディオの左右を合成した音と、左右の差分の音に分割することにより同一の連続メディアオブジェクトの品質を様々に変化することが可能となる。

特に、最近のビデオの圧縮法などでは、サブバンドコー

[1]UNIXは、UNIXシステム・ラボラトリーズ社の開発・許諾製品である。

ディングを用いたものやJPEGのプログレッシブモードやハイアラキカルモードやMPEG2のスケーリング機能などを用いることにより容易にメディアスケーリングを実現することが可能となる。

2.2 適応的な連続メディア処理

実際のアプリケーションでは、各メディアのサービスの質は固定した1つのQOSを用いて指定するのではなく、動的に変更可能なQOSとして表現するとシステムの柔軟性が格段に向上する。システムが混雑している時は、重要度の低いアプリケーションが処理する連続メディアオブジェクトの品質を自動的に低下する。一方、システムの負荷が軽くなった時に連続メディアオブジェクトの品質を向上する。また、新しくアプリケーションを実行する時、アプリケーションが必要とする資源が現在使用可能な資源より多いときは、重要度が低くメディアの品質を下げることが可能なアプリケーションを選択し、そのアプリケーションが処理する連続メディアオブジェクトの品質を下げることにより資源使用量を減少させ、新しいアプリケーションの実行を可能とする。

このように、メディアスケーリング可能な連続メディアオブジェクトを用いてサービスの質を動的に変化させる手法を動的QOS制御と呼ぶ。メディアスケーリング可能な連続メディアオブジェクトを用いて動的QOS制御を行なうことにより[2]、適応的な連続メディア処理が可能となる。メディアスケーリング可能な場合は、ユーザは実際に好ましいメディアの品質を決定する必要がある。なぜなら、ユーザは、連続メディアオブジェクトが提供可能な最低の品質に満足することができないかも知れないし、最高の品質を必要としないかも知れない。このように、ユーザが各自の好みをQOSのレンジとして指定することが可能となるので、連続メディアオブジェクトの品質とユーザの好みを分けることができるため、アプリケーションの柔軟性が格段に向上する。

マルチメディアマニュアルの再生を考えてみると、高品質のオーディオデータを再生するだけのCPU能力をワークステーションが持っていない場合は、オーディオデータのビット数を減らしたり、モノラルにしたり、オーディオデータを再生せず、そのオーディオデータと同じ内容のテキストデータを画面に表示することも可能である。ビデオデータに関しても同様にシステムの能力や負荷に応じて容易に連続メディアオブジェクトの品質を変更することができる。また、複合オブジェクトとして連続メディアオブジェクトを考えると、各連続メディアオブジェクトのそれぞれの品質を変化させるのではなく、あまり重要でない情報の連続メディアオブジェクトをユーザからの指定があるまで表示せず、ユーザからの要求があった時に、別な連続メディアオブジェクトの表示を止めて、指定された連続メディアオブジェクトを表示することにより、プレゼンテーション全体としての品質を動的に変化することも可能であると考えられる。

[2]遅延が大きくなってもかまわない場合は、いくつかのパケットをまとめることにより、全体の処理を減らすことでスループットを変化させるようなメディアスケーリングを伴わない動的QOS制御を考えることもできるが、本稿では考慮しない。

3 リソースの予約管理

実時間リソース管理はアプリケーションが指定したQOSを保証するためには有効であるが、これだけでは完全にQOSを保証することはできない。つまり、はじめに、QOSを保証することができるかどうかをチェックするアドミッション制御(Admission Control)と指定したQOS以上のリソースを用いて他のアプリケーションの実行を妨げないことをを保証するためのリソースの予約(Bandwidth Enforcement)が必要である。

我々の予約システムの特徴は従来の実時間システムなどで考えられていたものと異なり、リソースを最悪使用量に基づき悲観的にアロケーションするのではなく、平均使用量を用いて楽観的にアロケーションを行なう。悲観的なアロケーションを用いた場合は、時間制約を満足することは可能となるが、普段は滅多に使わない多くのリソースをアロケーションする必要があるため、連続メディア処理のように従来のアプリケーションと共存して実行する場合は不適当である。我々の予約システムでは、平均使用量に基づき楽観的にリソースの予約を行なうことによりリソースを有効に利用することを可能とする。そのため、各アプリケーションがリソースの平均使用量を越えた時に発生するリソースのオーバロード時の制御が重要となってくる。また、オーバロード時には一時的なメディアの質の低下が生じる可能性があるが、連続メディアのように一時的なメディアの品質の低下に耐えることが可能なアプリケーションでは、我々の提案する楽観的なアプローチの方がより好ましいと考えられる。

マイクロカーネルが予約すべき資源として、CPU、メモリ、I/Oが考えられる。ここでは、以下の節では、それぞれの予約システムに関して述べる。

3.1 CPUリソースの予約

あるアプリケーションが利用すると予測したCPU使用量より多くのCPUリソースを用いようとする場合は、他の連続メディアアプリケーションが用いるCPUリソースと衝突しないようにする必要がある。そのため、CPUリソースを予約する必要があるが、アプリケーションが必要なリソースの最悪値を用いて予約を行なうと資源が無駄になってしまう可能性がある。我々のCPU予約システムは、資源の平均使用量で予約を行なう。そして、システムの負荷が重くなった場合には、動的QOS制御を用いて、重要度の低いアプリケーションのQOSを低下させることで各アプリケーションのQOSを満足する。

CPUの予約は、各アプリケーションの実行の周期とその周期内に利用するCPUの利用時間により指定される.

アプリケーションは実行前に、アドミッションサーバにどの程度のCPUが必要かを通知する。要求が受け付けられた場合、アプリケーションは予約されたリソースが与えられ実行を開始する。予約されたCPUリソースを用いている場合は予約がアクティブであるといい、予約がアクティブである時の高い優先度を用いてアプリケーションを実行する。予約したCPUの使用量を使い切った時は予約がパッシブになったといい、アプリケーションがもともと指定した低い優先度で実行する。その時、アプリケーションは、予約した以上のCPUリソースを使おうとしたことが通知される。予約がパッシブになった場合は、一定時間後に再びアクティブになる。その時、アプリケーションの優先度も再び予約がアクティブになった時の高い優先度に変更される。

従来の実時間システムなどでは、固定優先度方式などのスケジューリング方式を用いていたが、通常のアプリケーションも同時に実行する場合は好ましくない。そのため、予約がアクティブなアプリケーションの実行は固定優先度方式に基づき優先的に実行するが、その他のアプリケーションや予約がパッシブなアプリケーションに関しては、従来のラウンドロビン方式を用いる必要がある。

また、システム全体のCPU利用率をモニタすることにより、システム全体の負荷が重くなった場合には、重要度が低いアプリケーションのCPU利用率を下げ、重要度が高いアプリケーションのメディアの質を高くする。CPUの予約システムにより、アプリケーション間の干渉を制御することが可能となるので、受け付けたアプリケーションのサービスの質を満足することが可能となる。

3.2 メモリリソースの予約

連続メディアアプリケーションが用いているメモリがページアウトされ、後ほどアクセスされた時にページインが発生した場合や、空きページを確保するため、使用中のページをページアウトが発生した場合など、連続メディアアプリケーションの時間制約が簡単に満足できなくなる場合が発生する。メモリの予約はCPUの予約と異なり、容易に横取り可能ではないのでCPUの予約システムと同様のアプローチを考えることはできない。従来の解決法は、実行前に連続メディアアプリケーションが用いるすべてのメモリを物理メモリ中にワイヤリングしてしまうことにより、ページアウトの発生を避けるようにする方法が考えられる。しかし、Motifツールキットなどを用いた連続メディアアプリケーションの場合は、そのアプリケーションが巨大なため、使用するメモリ量が莫大となり、貴重なメモリ資源を無駄にしてしまう可能性がある。そのため、通常使わない部分に関してはメモリをワイヤリングしないようにする必要がある。

我々のメモリ予約システムでは、連続メディアアプリケーションは 実行前に必要な物理メモリ量を予約する。システムは、物理メモリの割り当てが可能な場合は、アプリケーションが予約しようとしたメモリ量とと現在使用中のメモリ量の差分をフリーページとして確保する。これにより、連続メディアアプリケーションがメモリを新たに必要とした場合は、メモリを予約したメモリより迅速に割り振ることにより、アプリケーションの実行を中断せず実行を続けることを可能とする。

アプリケーションの使用するメモリのうち連続メディア処理に必要な部分だけを物理メモリにワイヤリングする必要がある。我々のメモリ予約システムでは、連続メディアアプリケーションの中の連続メディア処理を行なうスレッドがアクセスしたメモリのみをワイヤリングする。そのため、アプリケーションのうち大きなコード部分を占めるユーザからのイベントを受け付ける部分は物理メモリにワイヤリングされないため、メモリリソースを有効に使用することが可能となる。また、アプリケーションが予約された以上のメモリを使用しようとした場合は、カーネルはアプリケーションに通知する。アプリケーションは時間制約を満足するため、より多くのメモリの予約をするか、アプリケーションが使用しているバッファを小さくする必要がある。

また、システム全体の空きメモリ量が少なくなった場合は、QOSマネジャに通知することにより重要度の低いアプリケーションの使用するメモリ使用量を減らすことが可能となる。メモリ予約システムにより、連続メディアアプリケーションの時間制約を保証しなくなるようなページフォルトを不必要にメモリをワイヤリングすることなく発生しないようにすることが可能となる。

4 適応的な連続メディア管理

本節では、前節までに述べた機能を用い、いかにして適応的な連続メディア処理を行なうかに関して述べる。はじめに、適応的な連続メディア処理を行なうための連続メディアオブジェクトの表現法に関して述べる。次に、メディアスケーリング可能な連続メディアオブジェクトのディスク上での表現法に関して述べる。最後に、連続メディアオブジェクトの表現法とリソース予約システムを用いた適応的な連続メディア処理管理の方式について述べる。

4.1 適応的な連続メディアオブジェクトの表現

連続メディアオブジェクトを表現する方法として、オーディオデータの表現としては、ステレオの左右のディジタルデータを異なるストリームとして表現したり、1つのストリームにステレオの左右のデータを交互に配置する表現などが考えられる。また、ビデオデータに関しても、各フレームのの各ピクセルのRGBを順々に並べて表現したものや（通常のRGB表現）、白黒部分とカラー部分を分けて（CCIR 601）順番に並べたものなどを考えることができる。しかし、2.2節で述べたメディアスケーリング可能な連続メディアオブジェクトを実現するためには、連続

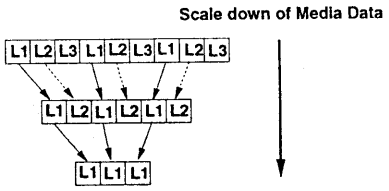

図 1: レベルの概念

メディアオブジェクトの表現を工夫する必要がある。基本的な考え方としては、最低限の品質のメディアを表現するデータと、インクリメンタルにメディアの品質を向上するデータを分離することである。つまり、最低限の品質のメディアデータにインクリメンタルに品質を向上するデータを付け加えることによりメディアの品質を向上することが可能となる。

従来の連続メディアオブジェクトを表現する方法だと、ネットワークにおけるパケットの紛失や、バッファがあふれたとき、連続メディアアプリケーションでの処理が時間制約を満足することができないとき、ランダムにデータをドロップする可能性があるため、最低の QOS の保証が困難となる。我々の方式では、できるだけインクリメンタルにメディアの品質を向上するためのデータをドロップすることにより、最低限のメディアの品質を保証する。

以上の問題点を解決する、我々の提案する連続メディアオブジェクトの表現法では、各メディアの表現にレベルというメディアの重要度を導入することである。直観的には、最低の品質を保証するためのデータに最高のレベルを与え、インクリメンタルに品質を改善するためのデータのうち、より品質を改善するためのデータにより低いレベルを割り当てる。そこで、システムが、データをドロップする必要がある時は、よりレベルの低いデータから優先的にドロップする。図 1では、一番上のメディアストリームが最も品質が高いメディアとなる。システムの負荷が思い時には、図に示すように、順番にレベルが低いデータをドロップしていくが、最低限の品質のメディアのみは表現できるようにする。

ここでは、レベルを表現する例として、以下の 5 つの場合を考える。1 つめの例としては、30fps のビデオデータを考える。ここで、最低 15bps のデータは表示したいとするときは、2 フレーム毎のデータのレベルを残りのフレームより高くする。また、可能なら、20fps を再生したいとすると、残り 15 フレームのうち 5 フレームのレベルを他の 10 フレームより高くする。2 つめの例としては、JPEG のプログレッシブモードを用いた時、より精度の高いデータを再生するために必要なデータの部分のレベルを残りのデータの部分より低くする。3 つめの例としては、MPEG を用いた時、I、P フレームを B フレームのレベルより高くすることが考えられる。また、4 つめの例としては、各フレームの白黒部分をカラー部分よりレベルを高くすることが考えられる。最後に、5 つめの例としては、ステレオオーディオの左右の和となるデータのレベルを左右の差分のデータより高くすることが考えられる。

以上のどの例の場合でも、より高いレベルのデータのみを処理することにより容易にメディアスケーリングを行なうことが可能となる。つまり、レベルの高いデータを処理することができない場合でも、レベルの低いデータの処理を行なうことにより、メディアの品質の低下をシステムの負荷に比例して順次落すことができるようになり、データのドロップによる予測不可能なメディアの品質の低下が発生しなくなる。

一般的にメディアの品質をどのように変化させるかに関しては、アプリケーションでは決めることができず、連続メディアオブジェクトの性質に依存したものであると考えられる。連続メディアの性質に関して詳しく知らないプログラマが、システムの負荷により、連続的にフレームレートを低下させるアプリケーションを作ったとする。連続的にフレームレートの変更により、品質の変更は連続的に行われるわけではないことがで示されているように、連続的なフレームレートの変更では、正しくメディアスケーリングを行うことができない。

また、我々の提案する方式では、ビデオの性質に依存したメディアスケーリングのポリシをアプリケーションの動的 QOS 制御のメカニズムと切り離すことができるため、制御の柔軟性が格段に向上する。例えば、動きが静的なビデオストリームでは、レベルが低いフレームを増やし、動きが激しいビデオでは、レベルが低いフレームを減らすことにより容易に連続メディアオブジェクトの意味を利用した柔軟なメディアスケーリングのポリシをアプリケーションを変更せずに採用することが可能となる。

4.2 適応的マルチメディア処理の実現

メディアスケーリング可能な連続メディアオブジェクトを処理するアプリケーションは、以下の 3 つのレベルでメディアスケーリングを実現する。

アプリケーションにおいて、コンポーネント間で用いられるバッファが足りなくなった場合は、データのドロップが発生する可能性がある。その場合は、バッファの中のレベルが低いデータを追い出し、後から追加されるレベルが高いデータをバッファに入れるようにする。ここで、連続的にデータがドロップする場合は、システムに要求を出して、より大きなメモリをバッファのために確保する。

また、ネットワークにおけるパケットロスを扱うため、レベルををネットワークが提供する優先度にマッピングする[3]。一般的に、優先度が低いほどよりパケットがロスされる可能性が増加する。また、ネットワーク上の遅延の増加により連続メディアオブジェクトの時間制約を満足でき

[3]例えば、ATM の Cell Loss Priority

ないことが検出された時は、レベルの低いデータを優先的にドロップする。ここで、ロスレートや遅延の定常的な増加によりメディアの品質が低下する場合は、ネットワークに対しQOSを変更する要求を出す必要がある。

最後に、アプリケーションが連続メディアオブジェクトを扱う時、時間制約を満足できないことが検出された場合は、現在のストリームを流れるデータのうちレベルの低いデータをドロップする。システムの負荷の増加によりレベルの低いデータを扱うことができない場合は、次のコンポーネントにデータを渡す前に扱うことができないレベルのデータをすべてフィルタリングして次のコンポーネントに送る。この時、データを送ってくるコンポーネントに対して、レベルを下げてデータを送るように要求を出す。しかし、マルチキャストを用いてデータを複数のコンポーネントに転送している場合は、送り側はメディアの品質を変更することができないため、データを受けとったコンポーネントでレベルが低いデータをドロップする。また、ストレージシステムやビデオキャプチャーのコンポーネントはディスクからのデータの読みだしやビデオキャプチャーからのデータの読み込みを減らす。

次に、リソース予約システムとメディアスケーリング可能な連続メディアオブジェクトの処理の関係に関して議論する。前節までで述べたメディアの表現法を用いた場合は、QOSの最低値と最高値は最高のレベルと最低のレベルにより表現される。つまり、最低のQOSを保証するため、最高のレベルのデータがすべて処理されることを保証する必要がある。システムのリソースの予約状況を管理するQOSマネジャは各アプリケーションのためのリソースの予約を行ない、アプリケーションの実行を開始した後、定期的に現在の処理されているデータの最高のレベルを通知する。アプリケーションはオーバランが発生したこと通知されると、現在の処理しているデータの最高レベルをオーバランが発生しないようになるまで低下する。また、オーバランが発生しない場合は、扱うデータの最高レベルを増加させる。そのため、アプリケーションは与えられたリソースに依存したレベルのデータを処理することが可能となる。QOSマネジャは、モニタした各アプリケーションのレベルと重要度を基に、重要度が高いアプリケーションのレベルが低くなっている場合は、重要度の低いアプリケーションの予約するリソース量を減少させる。

我々の方式は、従来のQOS管理方式と異なり、アプリケーションのQOSとリソース量の静的なマッピングが不必要となる。QOSとリソース使用量のマッピングを行うために、正確なリソースの使用量が必要となるため、実現が困難である。また、将来的にオペレーティングシステムにおける実時間処理の導入は一般的となると考えられるが、実時間処理の実現が可能なバスやネットワークはそれほど一般的に普及するとは考えにくい[4]。そのため、Ethernetのようなネットワーク上でも用いることが可能な我々のQOS管理方式の有効性は高いと考えられる。

5 現状と今後の課題

本論文では、従来のシステムでは、適応的なマルチメディア処理は困難であり、Real-Time Machが提供するマイクロカーネル技術、実時間処理、リソースの予約、マルチメディア向けのネットワークシステムとストレージシステムが重要であることを示した。また、適応的な連続メディア処理を行なうために必要なメディアの表現法と、それを用いた動的QOS制御に関して述べた。

現在、リアルタイムリソース管理、I/Oにおける優先度管理、メモリの予約システム、CRAS、NPSが実装が完了していて、これらを用いてフレームレートを動的に変更することにより動的QOS制御を実現するビデオオンデマンドシステムQtPlayが稼働中である。また、このシステムは、アプリケーションが使用しているリソースをグラフィカルに表示することが可能である。この実装により、実時間処理の有効性、メモリ予約の必要性、動的QOS制御の重要性が実証された。

参考文献

[1] L.Delgrossi, C.Halstrick, D.Hehmann, R.Herrtwich, O.Krone, J.Sandvoss, C.Vogt, "Media Scaling for Audiovisual Communication with the Heidelberg Transport System", In Proceedings of the ACM Multimedia'93, 1993.

[2] T.Nakajima and H.Tezuka, "A Continuous Media Application supporting Dynamic QOS Control on Real-Time Mach", ACM Multimedia'94, 1994.

[3] H. Tokuda, T. Nakajima, and P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System", In *Proceedings of 1st USENIX Mach Symposium, 1990*, October, 1990.

[4] ATMやFDDIのようなネットワークが各ワークステーションに直接接続されるような時代はなかなかこないと思われる。Ethernetはまだ長いこと使われると考えられる。また、Futurebusののような優先度をサポートしたバスもローエンドのマシンで提供されることは近い将来ないと考えられる。