

ネットワーク仮想記憶を利用した連続メディアの伝送方式

藤川 賢治[†] 岡部 寿男^{††} 美濃 導彦[†] 池田 克夫[†]

[†] 京都大学工学部情報工学教室

^{††} 京都大学大型計算機センター

動画・音声などの連続メディアの伝送に、ネットワーク仮想記憶を利用する方式を提案する。ネットワーク仮想記憶を利用することにより、ユーザインタフェースは通信を意識しないものとなり、ユーザは、ネットワーク上の記憶資源を、単一の計算機に対して行うのと同じように利用することができる。この方式を実現するため、連続メディアの QoS (サービス品質) を利用する。連続メディアは、再生時にいくらかのデータ損失が起きていても、それが許容範囲内であれば実用上問題がない。QoS を利用することにより、連続メディアを受信し再生するクライアントが実際にデータを読み出す前に、カーネルがデータをプリフェッチすることが可能となる。またページフォールトが起きなくなるため、実時間性を保証することができる。さらに通信にネットワーク仮想記憶を利用することにより、通信データの主記憶上でのコピー回数を減らすことができ、通信のオーバーヘッドを減少させることができる。

Transferring Continuous Media Using a Network Virtual Memory

FUJIKAWA Kenji[†] OKABE Yasuo^{††} MINOH Michihiko[†] IKEDA Katsuo[†]

[†] Department of Information Science, Kyoto University

^{††} Data Processing Center, Kyoto University

We propose a method using a network virtual memory for transferring continuous media. It provides user interfaces which makes users unconscious of the communication, and enables users to use memory resources on a network as if there were on a single computer. In order to establish such a mechanism, QoS (Quality of Service) of continuous media is utilized, where some data loss is permitted in actual use. In this case, a kernel can prefetch data before a client which receives continuous media and replays it actually reads data. Page faults are prevented and real-time scheduling is guaranteed. Furthermore using a network virtual memory for communication decreases the number of copying data for communication on primary memory, so the overhead of communication may decrease.

1 はじめに

ATM に代表される高速なネットワークの普及により、従来のネットワークでは不可能であった、動画・音声データなどの連続メディアの実時間伝送が可能となってきた。

これらのデータの伝送は、一般に、送信側がバッファを確保しデータを送信するプリミティブを呼

び出し、受信側がバッファを確保しデータを受信するプリミティブを呼び出すといった手順で行われる。

一方ネットワーク仮想記憶¹は、ユーザが上記のような手順なしに、サイト間でネットワーク上の

¹ ネットワーク仮想記憶は、分散共有記憶 [1]、共有仮想記憶 [2]、分散仮想記憶 [3] などいろいろな呼び方があるが、ここではまとめてネットワーク仮想記憶と呼ぶことにする。

記憶資源の共有を行うために使用される。ネットワーク仮想記憶により、ユーザインタフェースは、ネットワーク上の通信を意識させないものとなり、ユーザは、ネットワーク上の記憶資源を、単一の計算機に対して行うのと同じように利用することができる。

同一計算機内のサーバ・クライアント間の連続メディア通信に共有メモリを利用した研究として、Cyclic Shared Buffer 方式 [5] を用いる方式がある。しかしネットワーク仮想記憶は、連続メディアの伝送には使用されていない。これは、連続メディアの伝送は実時間で処理することが不可欠であるのに対して、ネットワーク仮想記憶は処理時間の見積りが困難なため実時間処理には適していないためである。

本稿では、MPEG などの連続メディアを想定し、その QoS(サービス品質)を指定することで、ネットワーク仮想記憶を利用した連続メディアの伝送方式を提案する。QoS を指定することにより、積極的にデータをプリフェッチし、実時間性を保証することができる。このような機能を実現するため、ネットワーク仮想記憶を提供する DM-2 カーネル [4] に機能拡張を行った。この方式により、ユーザがデータの送信・受信を意識しないプログラムをネットワーク上での連続メディア伝送においても記述することが可能となる。

以下本稿では、2章でシステムの構成、3章で QoS の利用、4章でアプリケーションへの適用に関して述べる。5章で利点と問題点について考察し、6章でプロトタイプの実装と評価を行う。

2 システム構成

2.1 対象システム

本研究の目的は、ワークステーション、パソコンクラスの複数台の計算機がネットワークによって接続された環境上で、マルチメディアアプリケーションが利用できるシステムを構築することである。連続メディアの伝送を必要とする実際のアプリケーションとして、蓄積型メディア (VOD など) と非蓄積型メディア (遠隔講義、遠隔会議など) を考えている。

ネットワークとしては、QoS の保証が可能な ATM ネットワークを想定しており、ネットワークの QoS の確保には RSVP [6]、ST-II [7] などを利用することを考えている。

本システムでは、連続メディアの伝送を、アプリケーションが明示的に行うのではなく、カーネルがネットワーク仮想記憶を提供することにより暗黙的に行う。アプリケーションプログラマは、通信プリミティブを使用することなく、データを通常のメモリ上で読み/書きする手順でデータの受信/送信が行える。

2.2 システムの構成要素

連続メディアの伝送は、クライアント・サーバ方式に基づいて行われる。システムは以下の要素から構成される (図 1)。

- カーネル
ネットワーク仮想記憶を処理する部分を外部タスクとする方式もあるが、カーネル・外部タスク間通信オーバーヘッドを削減し、実時間性の保証を容易にするため、本システムではカーネルがネットワーク仮想記憶を提供する。
- 送信サイト
蓄積型メディアを置いてある、または連続メディアの実時間送信を行うサーバタスクが起動している計算機。
- 受信サイト
連続メディアの受信を行うクライアントタスクが起動している計算機。
- タスク
タスク・スレッドモデルによりタスクを構成する。タスクに対して資源割当を行い、スレッドにより連続メディアの再生などの処理を行う。
- サーバ (サーバタスク)
連続メディアの実時間送信を行う。蓄積型メディアの送信にはサーバタスクは使われない。
- クライアント (クライアントタスク)
連続メディアの受信を行い、動画・音声などの再生を行う。
- オブジェクト
カーネルは連続メディアをオブジェクトとしてタスクに提供する。タスクは、オブジェクトをタスクの仮想記憶空間にマッピングすることによって連続メディアへのアクセスを可能とする。オブジェクトの実体は、自サイト・他サイトの主記憶・二次記憶上に存在する。

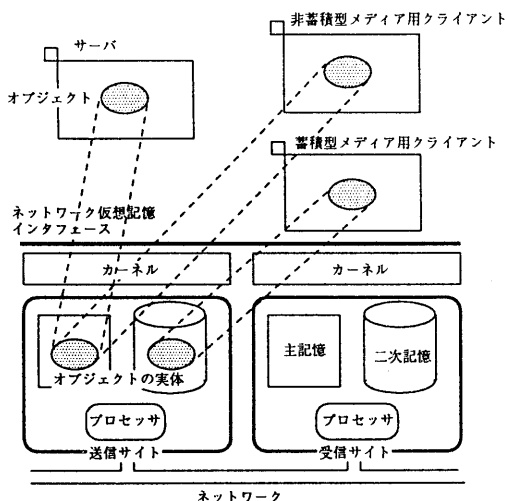


図 1: システム構成

2.3 対象データ形式

一般に通信データは伝送中にいくらかの損失が起きる。提案する方式では、カーネルはデータの損失が起きた場合でも、再送要求は行わず、そのデータは失われたものとして上位アプリケーションにはその分のデータは提供しない。これは再送が行われると実時間性の保証が難しくなるためである。ここでは、連続メディアは、再生時にいくらかのデータ損失が起きていても、それが許容範囲内であれば実用上問題がないことを仮定している。

2.4 カーネル内の処理

連続メディアの再生を行うスレッドは周期的スレッドとして生成される。周期的スレッドのスケジューリングポリシーとしては Rate Monotonic を使用する。

カーネルは、連続メディアとして生成されたオブジェクトには、連続アクセスを前提とした処理を行う。カーネルはこのオブジェクトに使用するページフレームを必要十分な数だけ固定的に用意し、ページング処理を行う。ページングアルゴリズムは FIFO で行い、ページアウト時はディスクへの書き込みなどは行わず、データは破棄する。また対象を連続メディアと限定しており、次に必要とするデータを完全に予測できる。そのため、カーネルは、アプリケーションが使用する前にデータをプリフェッチすることができる。

再生するデータの損失が起きた場合には、該当するアドレスのデータはアプリケーションから使えなくする必要がある。この方法として、そのアドレスを読みに行くとページフォルトを起こし読めなくするといった仕組みにするというものも考えられる。しかし想定しているハードウェアでは、記憶管理はページ単位 (例えば 4kbytes) でしか行えず、失われたサイズのデータだけ読めなくすることができない。そこで、カーネルはアプリケーションに読める範囲を通知するという手法で対応する。データの損失が起きた場合でも、周期的スレッドは起動されるが、データが読めないことを知り、その周期の処理では再生を行わないようにする。

2.5 QoS の利用

一般的な QoS(サービス品質) は、転送レート、遅延、エラー率などの属性によって定義される。本稿では連続メディアはオブジェクトとして定義され、タスクは連続メディアの実体が他のサイトに存在する場合であっても、タスクの仮想記憶中に存在するように見え、転送は意識されない。よって本稿では、オブジェクトには再生レートと遅延と呼ぶ QoS を定義する。これらはいずれも平均値を使用する。

再生レートはオブジェクト生成の際に決定され、オブジェクトは再生レートに基づいてネットワークの転送レートが決定される。再生レートは周期的スレッドのスケジューリングにも使用される。一方、遅延は、再生側、すなわちクライアントによって定義される。

周期的スレッドのスケジューリングおよびそれが可能かどうかの決定は次の要素によって決定される。

- R : 再生レート
- τ_i : スレッド内処理時間
- S : スレッドが周期一回の処理で必要とするデータ量
- τ_d : スレッドが要求する遅延時間
- T : スレッドの周期
- τ_{dd} : ディスク読み込みの遅延
- τ_{dn} : ネットワーク遅延

これらの関係については具体的なアプリケーションを例にして次章で説明する。

2.6 プリミティブ

連続メディアを扱うアプリケーションのためのプリミティブを以下に示す。

- `char *mo_open(name)`
`name` で指定されたオブジェクトをスレッドの仮想空間にマッピングし、そのアドレスを返す。
- `void create_thread(func, period, time, delay)`
周期 `period` で関数 `func` から開始される周期的スレッドを生成する。`time` により一回の処理時間、`delay` により許容遅延を宣言する。
- `next(mo_info)`
周期的スレッドの一回の処理を終了させる。
- `mo_info`
`mo_info` は配列であり、要素それぞれが一つのオブジェクトに対応する。`func` の引数で受け取るものは、オブジェクトの利用できるデータ範囲の情報が格納され、`next` に与える引数の場合は、サーバがどこまでオブジェクトに書き込みを行ったかをカーネルに通知するために使用する。

3 アプリケーションへの適用

3.1 蓄積型メディアへの適用

本システムでは、蓄積型メディアは QoS が指定されたオブジェクトとして送信サイトの二次記憶に蓄えられる。

クライアントが蓄積型メディアのデータを受信するには、まずクライアントの仮想空間上にオブジェクトをマッピングし、次に周期的スレッドを生成する。周期的スレッドは、データの到着を待って起動され、オブジェクトが存在するアドレスにアクセスして処理を行い、処理を終了する。毎回の起動時には、`mo_info` によりオブジェクトの利用できるデータ範囲がカーネルから通知される。このようなクライアントの処理の流れを図 2 に示す。

送信サイトにおいて、カーネルは、再生レート R で、ディスクからの入力、ネットワークへの出力を行う。

スレッドの周期 T と周期一回に処理されるデータのサイズ S と再生レート R には $T = \frac{S}{R}$ の関

```
....
char *p; /* オブジェクトのアドレス */
/* オブジェクトを利用可能とする */
p = mo_open(name);
/* 周期的スレッドを起動 */
create_thread(func, period, time, delay);
....
func(mo_info)
{
    /*
     * p が指すアドレスからデータを読み込み
     * 再生して p を増やす
     */
    next(NULL);
}
```

図 2: クライアントの処理の流れ

係があり、ユーザはこれらの関係を満たすようにアプリケーションを記述する必要がある。 $\tau_i < T$ で、かつ、すでに起動しているスレッドの処理に影響を及ぼさない場合にこのスレッドのスケジューリングが可能である。クライアントが複数のオブジェクトを使用する場合には、すべてのオブジェクトに対して上記のことが満たされねばならない。

送信サイトの二次記憶の読み出しから、周期スレッドの処理が始まるまでの遅延は、カーネル内での処理のオーバーヘッドが無視できるのならば、 $\tau_{dd} + \tau_{dn}$ で表される。これがタスクが指定する遅延 τ_d 以下でなければならない。ただし蓄積型メディアの場合は、VOD などに使用されるため遅延は問題にならないと考えられる。

複数のクライアントがいる場合、同時に同じ画像を見るのであればマルチキャストによりデータを送ることが可能である。クライアントが同時に同じ画像を見ないのであれば、データを受信する際、送信サイトがそれぞれに対してストリームを生成する。

3.2 非蓄積型メディアへの適用

非蓄積型メディアの伝送では、まずサーバが QoS を指定したオブジェクトを仮想空間上に生成し、そのオブジェクトが存在するアドレスに連続的に転送するデータを書き込む。サーバは、放送型のアプリケーション、すなわちデータを受け取るクライ

アントの状態を意識せずデータを送信するものとする。

クライアントは、オブジェクトを仮想空間上にマッピングすることで、いつの時点からでもデータを受信することができる。オブジェクトに対するアクセスによりデータの読み込みを行い、動画・音声を再生する。クライアントの処理は蓄積型メディアの場合と同様である。サーバの処理も同様の流れとなるが、func 時の mo_info は使用せず、next 時に引数として mo_info を与えることにより、書き込んだデータ量をカーネルに通知する。

ネットワークのレート制御およびスレッドの周期に関しては、蓄積型メディアの場合と同様である。

サーバがデータを書き込み始めてから、クライアントの読み込みが始まるまでの遅延は、 $\tau_{dn} + T$ で表される。これが要求する遅延 τ_d よりも小さくならなければならない。

クライアントが複数存在する場合には、蓄積型メディアと異なり、常にクライアントは同じデータを受け取る。

また本方式では、サーバとクライアントの両方の機能を実現するタスクを記述することもできる。このようなタスクの例として、サーバから受け取った連続メディアを加工して他のクライアントへ提供するものがあげられる。タスクは、受信用オブジェクトと送信用オブジェクトの mo_open を行い処理する。

4 本方式の利点と問題点

本稿で提案した方式の利点と問題点に関して考察する。

本方式が従来までのものと最も異なる点は、ユーザはアプリケーションを記述するのに、通信のバッファを用意して通信プリミティブを呼び出すといったことをしなくてもよいということである。これはアプリケーションを記述する上で、ユーザの負担を軽減するものである。

また連続メディア用として宣言されているオブジェクトは、必要とする部分を完全に予測できるためプリフェッチが可能である。クライアントが実際にデータを読み込む前に必要とするデータを用意することができるため、スレッドがページフォールトを起こすことはない。このように対象が連続メディアの場合には、その QoS を利用することにより、ネットワーク仮想記憶の処理の予測が困難

であるという問題を回避し、実時間性を保証することができる。

カーネルが通信用バッファ管理を行っていることでさらに効率上の利点も生まれる。通常の socket などの通信プリミティブを利用した伝送では、カーネルとアプリケーションとの間で通信データの主記憶上でのコピーが行われる。通信のオーバーヘッドを小さくするには、このコピー回数を減らすことが有効であることが知られており、送信の場合には、ユーザバッファからカーネルバッファを介さず直接デバイスのバッファへコピーする方式によりデータの主記憶上でのコピー回数を減らすことができる。しかし受信の場合には、受信のプリミティブが呼び出されるまではデータを格納すべきユーザバッファが分からないため、一旦カーネルバッファに保持しておく必要があり、受信の場合のようにコピー回数を減らすことが難しい。

シングルコピー方式 [8] では、受信の場合においても、カーネルバッファを介さず、通信デバイス上のバッファとユーザバッファとの間で直接コピーすることでコピー回数を削減している。しかしこの方式では、受信サイトにデータが届いてから受信のシステムコールを発行されるまでの間の時間が長いと、デバイス上のバッファに受信データが入り切れなくなってしまうといった問題がある。

提案した方式では、受信、送信いずれの場合においてもカーネルが管理するバッファにデータを格納し、このバッファをアプリケーションに仮想記憶を利用して提供する。このためなんら特殊なことを行わなくても送受信時のコピー回数を削減することができる。

本方式の問題点として、途中で受信データが失われたとき再送が行われない通信を考えているため、サーバのオブジェクトとクライアントのオブジェクトのデータの一貫性が保たれないといったことがあげられる。これまでのネットワーク仮想記憶の研究においてはデータの一貫性を保証することは、非常に大きな問題であったが、本方式においては、使用するデータを連続メディアに限定することにより、これが問題としないようにしている。

5 プロトタイプの実装と評価

今回 DM-2 カーネルに機能拡張を行い、提案した方式のプロトタイプの実装を行った。実際

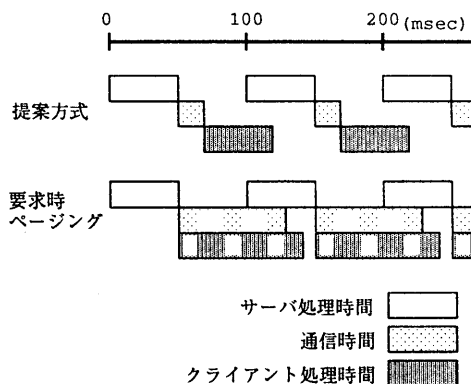


図 3: 連続メディア伝送時の処理の内訳

に使用したシステムは、IBM-PC 互換機 (CPU i486DX2/66) 2 台をイーサネットで接続したものである。イーサネットではネットワークの QoS が保証されないので、今回の実験では計算機 2 台のみをネットワークに接続し、極力外乱に影響されないようにした。

一方の計算機でサーバを、もう一方ではクライアントを起動し、本方式と要求時ページングによるページ整合性制御方式とで、連続メディアの伝送を行った。データは、1 フレームが 50 kbit で、毎秒 10 フレーム、1 フレームの再生にプロセッサの処理が 50msec かかるものを想定している。図 3 に実験結果を示す。

要求時ページングでは、データ伝送がクライアントが読み出そうとした時に始まるため、通信とクライアントの処理が交互に行われることになる。このため、サーバの次の周期が始まった時でも、前の周期の通信が終了せず、サーバ処理と通信が重なってしまう。また提案方式の通信速度は 2.5 Mbps であるのに対して、要求時ページングによる方式は 1.5 Mbps である。これは要求時ページングでは通信時に通信確認や再送を行うためである。

実験結果より、提案する方式では処理時間の見積りが容易であり、ネットワーク仮想記憶を利用した連続メディア伝送の実時間性を保証することが可能であることが示された。

6 おわりに

本研究では、QoS を指定し、ネットワーク仮想記憶を利用して連続メディアを伝送する方式について述べた。これにより、ユーザは、連続メディアを

伝送するプログラムの記述の際に通信を意識する必要がなくなる。システムは、実時間性を保証し、通信性能を向上させることができる。またプロトタイプの実装と評価を行った。

今後の課題として、ATM ネットワーク上での実装、実際のアプリケーションでの評価を行うことがあげられる。

謝辞

本研究を行うにあたり貴重な御意見、御指示を与えて下さった皆様に深く感謝致します。

参考文献

- [1] Stumm, M. and Zhou, S., "Algorithms Implementing Distributed Shared Memory", IEEE COMPUTER, Vol.23, No.5, pp.54-64, 1985.
- [2] Li, K. and Hudak, P., "Memory Coherence in Shared Virtual Memory Systems", ACM Transactions on Computer Systems, Vol.7, No.4, pp.321-359, 1989.
- [3] 篠原拓嗣, 藤川賢治, 大久保英嗣, 津田孝夫, "分散仮想記憶に基づくオペレーティングシステム DM-1 の構成", 情処研報 93-OS-61-7, pp.49-56, 1993.
- [4] 青木秀貴, 篠原拓嗣, 藤川賢治, 大久保英嗣, 津田孝夫, "分散オペレーティングシステム DM-2 におけるメモリ管理部の実現", 情処研報 95-OS-71, 1995.
- [5] 西尾信彦, "Real-Time Mach 3.0 上の連続メディア処理のための強調サーバ群の設計と実験", 情処研報 94-OS-63-8, 1994.
- [6] Braden, R., Zhangk, L., Estrin, D., Herzog, S. and Jamin S. "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", Internet Draft, draft-ietf-rsvp-spec-07, 1995.
- [7] Topolovic, C., Casner, S., Lynn, C. Jr., Park, P. and Schroder, K., "Experimental Internet Stream Protocol, Version 2 (ST-II)", RFC1190, 1990.
- [8] Banks, D. and Prudence, M., "A High-Performance Network Architecture for a PARISC Workstation", IEEE Journal on selected areas in communications, Vol. 11, No.2, pp.191-202, 1993.