マルチメディア通信と分散処理　67－9
（１９９４．１２．２）

通信処理サーバにおける高効率高速・多回線制御方式の検討

平田 俊明† 近藤 毅† 松永 和男‡ 高田 治†
†日立製作所システム開発研究所 ‡日立製作所ソフトウェア開発本部

本稿ではDMA(Direct Memory Access)による入出力方式とOSI(Open Systems Interconnection)等の階層型プロトコルを実装する通信処理サーバを対象に高速回線と同時に多数の低速回線を効率よく制御するための通信制御方式を検討する。検討は通信プロトコル処理の高速化と通信回線との入出力処理の高速化の２つの側面から行う。前者についてはデータ複写処理およびモジュール間のスケジュールオーバヘッドの削減方式を、後者については複数回線に対する入出力処理の並列化方式、入出力処理の一括化方式を検討する。また、データ複写処理を不要とするバッファ制御方式では、プロトコルヘッダの別バッファ化に伴うDMA転送効率の低下を防ぐショートバッファ詰め替え方式を検討する。検討した方式は通信制御プログラムの１実装方式として有効である。

A Study of efficient communication control method on communication server

Toshiaki HIRATA † Takeshi KONDOU † Kazuo MATSUNAGA ‡ Osamu TAKADA †
† Sysyems Developmet Laboratory,Hitachi,Ltd. ‡ Software Development Center,Hitachi,Ltd.

This paper proposes efficient communication control method on communication server. Proposed methods are concerned with efficiency of input/output to communication lines and with efficency of protocol proccessing. The former is the method of parallel and collective input/output during plural communication lines. The latter is the method which reduce load of data copying and module scheduling. Proposed methods are effective for implementation of multi-layered protocol (e.g.,OSI) proccessing.

1 はじめに

近年のダウンサイジング化、オープン化に伴い、オンラインシステムにおいては汎用大型計算機を中心とした集中型システムからサーバ機、ワークステーション、パソコンなどを統合した分散処理システムへと移行しつつある。このうち、サーバ機では分散処理環境で通信処理やトランザクション処理を行なうため多数の端末を通信回線で接続する形態で使用されることがある。

本稿ではサーバ機（以降、通信処理サーバと記す）における通信制御処理の高速化を検討する。通信制御処理の高速化方式については種々の研究が行われている [1][2][3]。これらは通信プロトコル処理の高速化に関する検討が中心であり通信媒体との入出力処理も含めた検討は十分でない。本稿ではDMAによる入出力方式と階層型プロトコルを実装する通信処理サーバを対象に高速回線と同時に多数の低速回線を効率よく制御するための通信制御方式を検討する。検討は通信プロトコル処理の高速化と通信回線との入出力処理の高速化の２つの側面から行なう。通信プロトコル処理ではアプリケーション空間からシステム空間への移動、データの分割等によるデータ複写処理およびモジュール間のスケジュールのオーバヘッドが大きく、これらの効率化が課題である。これらに対して以下の方式を検討した。

(1) データ複写処理の最適化方式
(2) データ分割によるデータ複写を不要とするバッファ制御方式
(3) モジュール間のスケジュールのオーバヘッドを削減する実行制御方式

一方、通信回線との入出力処理の効率化のためには、入出力処理の並列化、ソフトウエア処理の削減、回線制御装置との間のデータ転送処理の効率化が課題である。これらに対して以下の方式を検討した。

(1) 複数の通信回線に対する入出力処理の並列化方式
(2) パケット分割データの一括出力処理方式
(3) ショートバッファ詰め替えによる入出力アダプタとの間のデータ転送処理の効率化方式
(4) 受信バッファの一括準備方式

以下、２章ではシステム構成を、３章では高速化に対する課題を、４章では高速化方式を検討する。

2 システム構成

2.1 通信処理サーバを用いたシステム構成例

図１に本稿で対象とする通信処理サーバを用いたシステム構成の一例を示す。本例はメインフレームとワークステーション／ＰＣをサーバで接続する垂直分散型のシステムである。通信処理サーバは通信回線により多数のワークステーション／ＰＣと接続している。

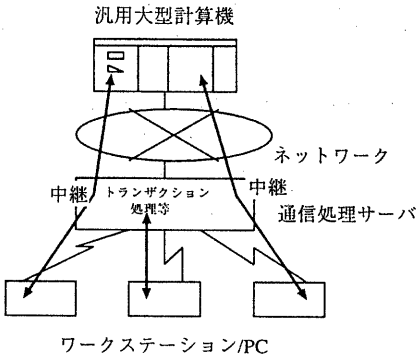

図 1: 通信処理サーバを用いたシステム構成

2.2 通信処理サーバの構成

図２に本稿で対象とする通信処理サーバのハードウエア構成および前提とする入出力制御方式を、図３に通信制御プログラムの構成を示す。通信処理サーバはCPU(Central Processing Unit)、主記憶装置および回線制御装置から構成し、入出力効率を考えDMA(Direct Memory Acssess)方式により回線制御装置と主記憶装置間のデータ転送を行なうこととする。入出力の起動は回線制御装置内のレジスタを主記憶空間にマッピングした特定のアドレスに入出力要求をストアすることにより行なう（メモリマップドＩ／Ｏ）。通信制御プログラムはCPU上、システム空間にて動作し、アプリケーションプログラムに対してデータ送受信のためのインタフェース(API:Application Program Interface)を提供する。アプリケーションプログラムからデータ送信要求があると通信制御プログラムはアプリケーションプログラムが用意したデータをシステム空間に複写し、通信プロトコル処理を実施した後、回線制御装置に対して送信要求を行なう。回線からの受信デー

タはこの逆の処理を行ないアプリケーションプログラムにデータを引き渡す。

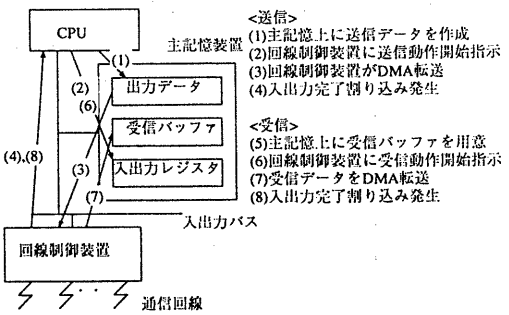

図 2: 通信処理サーバのハードウエア構成

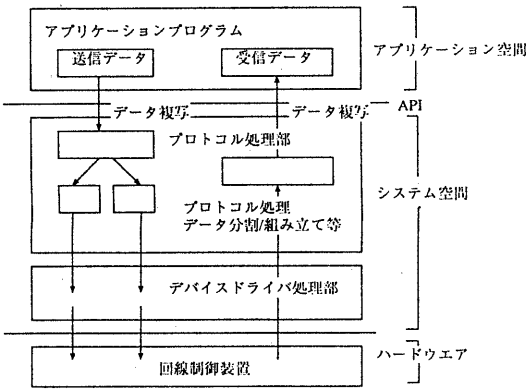

図 3: 通信制御プログラムの構成

3 高速化に対する課題と対応方針

高速化に対する課題を通信プロトコル処理の高速化と通信回線との入出力処理の高速化の 2 つの側面から示す。

3.1 通信プロトコル処理の高速化

通信プロトコル処理ではデータ複写処理とスケジュール処理の負荷が大きくこれらを削減することが課題である。データ複写処理はアプリケーション空間とシステム空間との間のデータ移動、データ分割、トレース／ログ等の保守情報の採取において発生しうる。データ複写処理の負荷を削減するため本稿ではデータ複写処理の高速化とデータ複写処理自体をなくすことが可能かの両面から検討する。スケジュール処理の高速化ではモジュール間の制御の受渡しでコールインタフェースを基本としながらもリカーシブコールを防ぐ方式を検討する。これは通信プロトコルを実現するマトリクス処理ではリカーシブコールが発生すると状態の不一致等の不具合が生じることがあるためである。

3.2 通信回線との入出力処理の高速化

多数の通信回線との入出力処理の効率化のためには、複数の回線に対する入出力処理の並列化、ソフトウエア処理の削減、回線制御装置との間のデータ転送効率の向上が課題である。これらに対しては、回線制御装置との入出力インタフェース方式とこれを使用するデバイスドライバ制御方式の両面から検討する。

4 高速化方式の検討

4.1 通信プロトコル処理の高速化

課題に対し以下の施策を検討する。

(1) データ複写処理の最適化方式

計算機システムのメモリ内のデータ複写は例えば 4 バイト単位、2 バイト単位、1 バイト単位のようにある特定の大きさについて可能であり、それぞれのデータ単位での複写は複写元のデータ格納先頭位置と複写先のデータ格納先頭位置の境界を一致させることで半端分を除き最大のデータ単位での複写が可能となり、最も効率のよいデータ複写処理が実現できる。前記方式はデータ複写処理において汎用的に適用できる方式であるが、特にユーザデータをシステム空間に複写する場合、システム空間上のデータ境界は DMA 転送時に最も効率のよい値に設定すべきである。この値は一般にデータ複写の最大データ単位の整数倍であり、ユーザデータの境界値をこの値に合わせればシステムトータルとしてのデータ複写処理の最適化がはかれる。ユーザプログラムにこの制約を課すことが困難な場合、図 4 に示すように通信制御プログラム側でデータ長によってシステム空間へのデータ複写を優先させるか、DMA 転送を優先させるかを決める処理を行なう方式も考えられる（データ長が短い場合、DMA 転送を優先し、長い場合はシステム空間へのデータ複写を優先させる）。

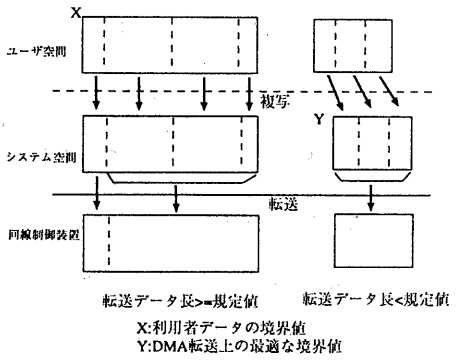

図 4: DMA 転送を考慮したデータ複写方式

(2) データ分割によるデータ複写を不要とするバッファ制御方式 [5]

図 5 に示すようにバッファ単位に設けたバッファコントロールブロック（以下 BCB:Buffer Control Block と記す）に参照カウント (cnt) とバッファヘッダ (以下、BH:Buffer Header と記す）を用いたデータチェインの概念を導入することにより複数の BH から同一のバッファを参照することを可能とする。

これにより連続した 1 つの領域からなるバッファ

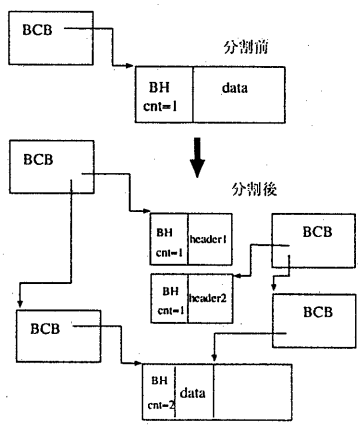

図 5: バッファ制御方式

を論理的に複数のバッファとして扱うことが可能となりデータ分割の際にデータコピーが不要となる。なお、本方式に付随して発生する DMA 転送時の問題点については 4.2 で検討する。

(3) スケジュールのオーバヘッドを削減する実行制御方式

図 6 に示すようにモジュール間のスケジュールについてモジュールのペア対応にスケジュールコントロールブロック (以降 SCB:Schedule Control Block と記す）を設け、あるモジュールからみて自側の制御領域と相手側の制御領域に分ける。

スケジュール時自側のモジュールの SCB 内領域に

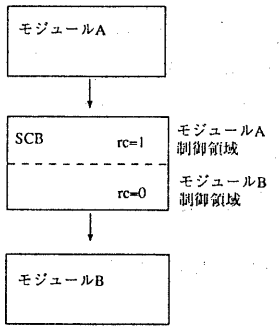

図 6: モジュール間のスケジュール方式

設けたリカーシブコールビット (rc ビット）を参照し、これがオフであるとき相手側のモジュールの SCB 内領域のリカーシブコールビットを設定して当該モジュールを呼び出す。前記ビットがオンのときリカーシブコールが発生したと判断し、当該要求をデイスパッチキューに登録して適当なタイミングでデイスパッチャが当該モジュールを呼び出す。なお、rc ビットは当該モジュールのリターン時にリセットする。

以上により通常はコールインタフェースとしリカーシブコール時にはデイスパッチャ経由とするスケジュール方式が実現できる。

4.2 通信回線との入出力処理の高速化

課題に対して以下の施策を検討する。

(1) 複数の通信回線に対する入出力処理の並列化方式

図 7 に示すように CPU はコマンドチェインを含む一連の入出力要求を配列から構成するキューに順次登録し回線制御装置に送信動作開始を指示する。一方、回線制御装置はこれを CPU の登録順にとりだし回線との入出力を行なう。

回線との入出力処理で待事象が発生したときは処理待となっている別の回線の処理を行なうことにより入出力処理の並列化をはかることができる。複数の回線との入出力要求は複数のアプリケーションが非同期に入出力要求を発行する環境ではごく一般的

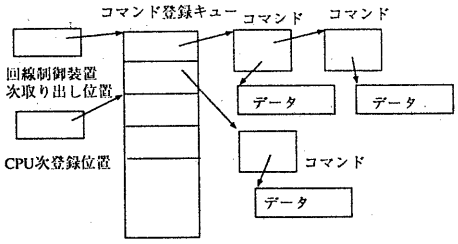

図 7: 回線制御装置との入出力方式

に発生しうる。また、完了割り込みは入出力要求とは非同期に受ける方式とするが、入出力要求が連続して発生する場合は回線制御装置の処理能力上、連続して要求できる個数を制限するか、CPU と回線制御装置間にフロー制御機構を設けるなどの配慮を行う。

(2) パケット分割データの一括入出力方式

例えば X.25[4] ではコネクション上のパケット長に制約がありデータ分割が発生する。図 8 に示しように分割したデータをコマンドチェイン機能を用いて一括して回線制御装置に渡すことにより入出力処理の負荷を削減する。

このとき、回線制御装置には複数のデータを受信

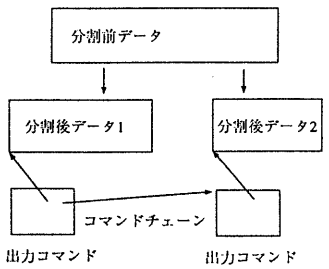

図 8: パケット分割データの一括出力方式

しこれを順次通信回線に送信する機能を設ける。本方式を採用することにより CPU 上の出力要求処理、入出力完了割り込み処理を 1 つのデータごとに入出力要求を行う方式と比較して最大 1/n (n はコマンドチェイン数) に削減することが可能となるばかりでなく、回線制御装置の入出力処理も削減できる。

(3) ショートバッファの詰め替えによる回線制御装置との間のデータ転送処理の効率化方式

本稿で提案したバッファ制御方式では階層型プロトコルに対しては図 5 のようにプロトコルヘッダが別バッファとなるため、短いバッファが連続する。このようなデータを入出力する場合データチェインを用いるのが一般的である (回線上は 1 つのデータ) がデータチェインを用いると DMA 転送においてはバッファ切替のオーバヘッドが発生する。計算機システムによってはこれは無視できないケースがあり、図 9 に示すように複数のショートバッファ上のデータを別の 1 つのバッファ (入出力コマンド用のバッファを使用することによりバッファ確保に係る余計な負荷は発生しないようにすることが可能) 上に複写してバッファ切替のオーバヘッドを削減する方式が有効である。このとき、バッファ切替のオーバヘッドとデータ複写のオーバヘッドのトレードオフにより、データ長により詰め替えを行なうか否かのスレッショルドが存在する。この値は計算機システムごとに評価し決定する。

本提案方式に対しては API においてプロトコルヘッ

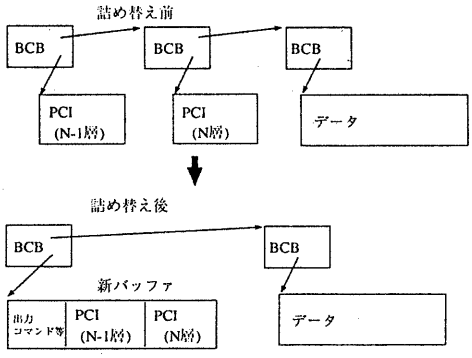

図 9: ショートバッファ詰め替え方式

ダ用に必要な長さのバッファを一括して事前に確保しておく方法があるが、API において実装したプロトコル階層を意識する必要があり汎用性が失われる。これに対して提案方式は前記制約のない汎用性の高い方式と言える。

(4) 受信バッファの一括準備方式

複数の受信バッファを一定の個数一括して準備し回線制御装置に渡しておき (受信用のコマンドのチェインにより要求) 全受信バッファを使いきった時点で再度回線制御装置に受信バッファを渡す方式とする。この方式により、回線制御装置から受信割り込みを受けるごとに受信バッファを準備する必要がなくデータの受信効率を向上できる。図 10 にこれらの様子を示す。本方式では回線制御装置に対する受信コマンドの発行処理を受信バッファを 1 個づ

つ用意する方式と比較して最大 1/n(n は受信バッファの準備個数) に削減できる。

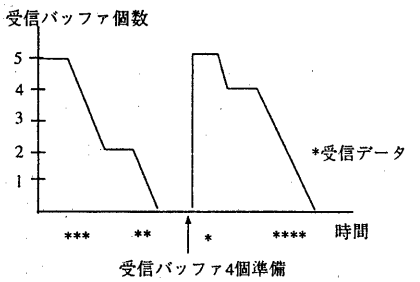

図 10: 受信バッファ一括準備方式

5 おわりに

本稿では DMA による入出力方式と階層型のプロトコルを実装する通信処理サーバを対象に少数の高速回線と同時に多数の低速回線を効率よく制御するための通信制御方式を検討した。検討は通信プロトコル処理の高速化と通信回線との入出力処理の高速化の 2 つの側面から行った。前者についてはデータ複写処理およびモジュール間のスケジュールオーバヘッドの削減方式を、後者については入出力処理の並列化方式、入出力処理の一括化方式を検討した。また、データ複写処理を不要とするバッファ制御方式では、プロトコルヘッダの別バッファ化に伴う DMA 転送効率の低下を防ぐショートバッファ詰め替え方式を検討した。検討した方式は通信プロトコルの 1 実装方式として有効である。

[謝辞]

日頃ご指導頂き、本研究においては有益なご討論、ご助言をいただいた日立製作所システム開発研究所および同ソフトウエア開発本部の関係者の方々に感謝いたします。

参考文献

[1] LIBA SVOBODOVA:Implementing OSI Systems,IEEE Journal on Selected Areas in Communications. VOL.7,NO.7,pp.1115-1130(Sep.1989)

[2] 石橋豊、他：階層型通信プロトコル処理方式と評価、信学技法 SSE91-74 (1991)

[3] 石坂充弘、他：OSI プロトコルにおける受信データの並列処理方式、電子情報通信学会論文誌 B-I Vol.J74-B-I NO.2 pp.116-128 (1991.2)

[4] CCITT:REDBOOK,Vol.VIII-fascicleVIII-3(1984)

[5] 井戸上彰、他：汎用 OSI7 層ボードにおけるプロトコルデータ単位のバッファ制御方式、情報処理学会第 44 回全国大会、 4L-12(1992)