(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3644158号
(P3644158)
(45)発行日　平成17年4月27日(2005.4.27)　(24)登録日　平成17年2月10日(2005.2.10)

(51)Int.Cl.[7]　F I
G06F 15/17　G06F 15/17　635B

請求項の数 4　(全 13 頁)

(21)出願番号　特願平8-304426
(22)出願日　平成8年11月15日(1996.11.15)
(65)公開番号　特開平10-143486
(43)公開日　平成10年5月29日(1998.5.29)
審査請求日　平成13年10月3日(2001.10.3)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100075096
弁理士　作田　康夫
(72)発明者　坂口　明彦
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　佐川　暢俊
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

最終頁に続く

(54)【発明の名称】並列計算機におけるデータ送受信方法

(57)【特許請求の範囲】

【請求項１】

複数台の計算機とこれらを相互接続する通信路からなる並列計算機の該計算機間のデータ転送方法であって、

該各計算機内に、通信相手となる計算機毎に固定された領域である静的受信バッファ領域と、通信相手となる計算機毎の割り当てが固定されていない領域である動的受信バッファ領域を予め確保するステップと、

送信データのデータ長が予め定められた値より短い場合には、送信元の計算機からのリモートメモリ書き込みにより送信先の前記静的バッファ領域に送信データを書き込んでデータ送受信を行うステップと、

送信データのデータ長が予め定められた値より長い場合には、送信元の計算機から送信先の計算機に該送信データのデータ長を含む情報を伝達するステップと、前記送信先の計算機にて、前記情報により前記送信データを受信すべき動的バッファ領域を決定し、該決定した動的バッファ領域のアドレスを前記静的バッファ領域を用いたリモート書き込みにより前記送信元の計算機に返送するステップと、前記送信元の計算機からのリモート書き込みによって、返送されたアドレスの示す動的バッファ領域に前記送信データを書き込むステップを有することを特徴とする並列計算機におけるデータ送受信方法。

【請求項２】

複数台の要素計算機を通信路によって結合しており、任意の要素計算機上のデータ転送用メモリ領域上にあるデータを任意の他の要素計算機上のデータ転送用メモリ領域に書き

込むリモートメモリ書き込み機構を有する並列計算機における任意の２つの要素計算機の間のデータ送受信方法であって、
各要素計算機は、利用者プログラムからの通信初期化要求を契機として、該データ転送用メモリ領域上に、通信相手となる計算機毎にあらかじめアドレスの固定された静的受信バッファ領域と、通信が発生するたびにそこからバッファが動的に割り当てられる動的受信バッファ領域を確保するステップと、
利用者プログラムからの送信要求を契機として、送信側要素計算機は、利用者メモリ上の送信データを該データ転送用メモリ領域にコピーし、データ長に関する情報を含むヘッダを構成し、該送信データのデータ長があらかじめ定められた値より短い場合には、該ヘッダおよびデータを受信側の要素計算機上に用意された該静的バッファ領域にリモートメモリ書き込み機構を用いて書き込み、該送信データのデータ長があらかじめ定められた値より長い場合には、該ヘッダを受信側の要素計算機上に用意された該静的バッファ領域にリモートメモリ書き込み機構を用いて書き込むステップと、
受信側の要素計算機は、該ヘッダの到着を契機として、該ヘッダを参照してデータ長を取得し、該送信データのデータ長があらかじめ定められた値より長い場合には、該データ長が必要なバッファを該動的バッファ領域上に確保してそのバッファのアドレス情報を送信側の要素計算機に通知するステップと、
送信側の要素計算機は、該送信データのデータ長があらかじめ定められた値より長い場合には、該アドレス情報の到着を契機として、該アドレス情報を参照してデータを受信側の要素計算機の該確保された動的バッファ上にリモートメモリ書き込み機構を用いて書き込むステップと、
利用者プログラムからの受信要求を契機として、受信側要素計算機は、該静的バッファ領域または該動的バッファ領域に書き込まれた該データを利用者メモリ上の受信バッファにコピーするステップとを有するデータ送受信方法。
【請求項３】
利用者プログラムから複数の送信要求が発行された場合に、該ヘッダ情報中に次の送信要求時に使用する静的受信バッファのアドレス情報を格納することにより、複数の送信要求の発行順を保証するステップを有する請求項２記載のデータ送受信方法。
【請求項４】
複数台の要素計算機を通信路によって結合しており、任意の要素計算機上のデータを他の任意の要素計算機のメモリに書き込むリモートメモリ書き込み機構を有する並列計算機における任意の２つの要素計算機の間のデータ送受信方法であって、
各要素計算機は、該データ転送用メモリ領域上に、通信が発生するたびにそこからバッファが動的に割り当てられる動的受信バッファ領域と、２つの送信バッファを確保する。該動的受信バッファ領域は、あらかじめ定められた固定長の複数のブロックよりなるように構成するステップと、
送信側要素計算機は、利用者プログラムからの送信要求を契機として、該送信データを、該あらかじめ定められた複数の固定長のパケットに区切り、データの長さに関する情報を含むヘッダを構成し、該ヘッダを受信側の要素計算機に通知するステップと、　受信側の要素計算機は、該ヘッダ情報の到着を契機として、該ヘッダ情報を参照して必要な数の該ブロックよりなる受信バッファを該動的受信バッファ領域上に確保し、該バッファのアドレス情報を送信側の要素計算機に通知するステップと、
送信側要素計算機は、該バッファのアドレス情報の到着を契機として、該複数のパケットのうちｎ番目のパケットを該データ転送用メモリ領域上の該２つの送信バッファのうち一方にコピーし、既に該データ転送用メモリ領域上の該２つの送信バッファのうちもう一方にコピーされたｎ－１番目のパケットを、リモートメモリ書き込み機構を用いて、受信側の要素計算機上の該動的受信バッファのｎ－１番目のブロックに書き込み、上記２つのステップをすべての該複数のパケットについて順次適用するステップと、
利用者プログラムからの受信要求を契機として、受信側要素計算機は、該受信バッファの複数のブロックに書き込まれた該データを利用者メモリ上の受信バッファに順次コピー

するステップとを有するデータ送受信方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は複数の要素計算機（プロセッシングユニット、以下PU）を通信網によって結合した並列計算機におけるPU間のデータ送受信方法に係わり、特にメッセージパッシングの高速性とデータの安全性の確保するデータ送受信方法に関する。
【０００２】
【従来の技術】
並列計算機は、複数のPUを通信網によって結合し、それらを同時に稼働させることによって処理速度を向上させる。本発明では特に、各PUがそれに付随するメモリ空間のみをアクセスすることができる分散メモリ型の並列計算機を対象とする。分散メモリ型並列計算機では、他のPUのメモリ上にあるデータを直接アクセスすることはできない。データが必要となる度に送受信を行ってそのデータを自PUに移動する必要がある。
【０００３】
分散メモリ型並列計算機では、PU間のデータのやりとりをすべてプログラム中に記述する必要がある。ここでPU間で受け渡されるデータをメッセージと呼ぶ。このメッセージをやりとりすることをメッセージパッシングと呼ぶ。並列計算機用プログラムでは、他のPUで必要となるデータが自PUのメモリ上にある場合には自PUはあらかじめこれらデータを他のPUへ送信し、他のPUのメモリ上にあるデータを自PUが必要とする場合には自PUはあらかじめ他のPUからこれらデータを受信しておくような指示を各PUのプログラム中に明示的に記述する必要がある。
【０００４】
多くの並列計算機システムでは、このようなPU間のメッセージの送受信をサポートする目的で、メッセージパッシングライブラリと呼ばれる関数（あるいはサブルーチン）群があらかじめ用意されており、通信はCやFORTRANなどのプログラムからの関数コールとして記述できるようになっている。メッセージパッシングライブラリの中には、異なる並列計算機ハードウェア上にインプリメントされ、事実上の標準としての通信環境を提供するものも現れている。米国のOak Ridge National Laboratoryで開発されたPVMや、近年標準化が進められているMPIはその例である。これらの通信ライブラリをコールすることにより書かれた並列プログラムは、異なる並列計算機上でも再コンパイルのみで動作させうる可能性（可搬性）が高い。
【０００５】
通信ライブラリでPU間のメッセージの受け渡しを行うには、送信側PUでメッセージの送信関数をコールし、受信側PUでそれに対応するメッセージの受信関数をコールし、これらの間でメッセージを送受信する方式が現在一般に用いられている。送信関数より受信関数が先にコールされた場合には受信関数はデータの到着までブロック（停止）し、送信関数が先にコールされた場合には受信関数の開始までブロックするか、メッセージがシステム内にバッファリングされる。これはsend/receive方式と呼ばれる。
【０００６】
PU間のデータ移動を高速に実現する方法として、リモートメモリ書き込み機構を持つ分散メモリ型の並列計算機がある。リモートメモリ書き込みでは、各PUは相手PUの介入なしに直接相手PU内の特定メモリ領域へのデータ転送が可能である。リモートメモリ書き込みを行うことのできる特定メモリ領域は、リモートメモリ書き込み領域と呼ばれる。リモートメモリ書き込み領域は、実アドレスが連続であり、スワッピングされないために各PUが随時データ転送することができる。
【０００７】
図２は、リモートメモリ書き込み機構を持つ並列計算機上でメッセージパッシングを実現するための従来手法を示す。各PU間の実際のデータ転送は、リモートメモリ書き込み領域間で行われる。まず送信関数がコールされると、送信側のユーザプログラム中のバッファ

(２０１)からリモートメモリ書き込み領域内のバッファ(２０２)へデータがコピーされる。次にそこから受信側のリモートメモリ書き込み領域内のバッファ(２０３)にリモートメモリ書き込みを用いてデータが転送される。最後に受信関数がコールされて、リモートメモリ書き込み領域(２０３)からユーザプログラム中のバッファ(２０４)にメッセージがコピーされて、メッセージの受け渡しが完了する。

【０００８】

【発明が解決しようとする課題】

上述のように、リモートメモリ書き込みでは相手PUの介入なしにデータの転送を行う際は、受信側のリモートメモリ書き込み領域の使用を確認しないと、受信側で必要なメモリ領域を書き込みデータで上書きしてしまい、受信側のデータを壊してしまう恐れがあった。そこで受信側のリモートメモリ書き込み領域を確認しないと、続けてデータ転送を行うことができなかった。また、送信側でのユーザプログラムのバッファからリモートメモリ書き込み領域へのデータ転送、送信側のリモートメモリ領域から受信側のリモートメモリ領域へのデータ転送、受信側でのリモートメモリ書き込み領域からユーザプログラム内のバッファへのデータ転送と、計３回のデータ転送が必要であった。また、リモートメモリ書き込み領域に別々に送られてきたデータ間の順序を保証する事も出来なかった。

【０００９】

本発明の目的は、リモートメモリ転送を限られた量の転送用メモリで高速に行うことと、相手PUが独自に送ってきたデータの順序を保証することにある。

【００１０】

【課題を解決するための手段】

本発明は、複数台の計算機とこれらを相互接続する通信路からなる並列計算機において、各計算機内に、通信相手となる計算機毎に固定された領域である静的受信バッファ領域と、通信の発生時に動的に割り当てられる動的受信バッファ領域を確保するステップと、送信データのデータ長が予め定められた値より短い場合には、送信先の該静的バッファ領域に送信データを書き込むステップと、送信データのデータ長が予め定められた値より長い場合には、送信先の該動的バッファ領域のアドレスを該静的バッファ領域を用いて受信するステップと、送信先での該受信したアドレスの該動的バッファ領域に当該送信データを書き込むステップを設けることによって、達成される。

【００１１】

また予め定められた値より長いデータを転送する場合には、パイプライン処理でデータを転送することで、限られたメモリ量において高速にデータ転送を行うことができる。

【００１２】

また、送られたデータの順序性を、使用した転送用メモリ（バッファ）を順につなぐことで保証することができる。

【００１３】

【発明の実施の形態】

以下、図を参照して本発明の詳細を説明する。

【００１４】

まず、本発明の実装方法の具体例を図を参照して説明する。図１に本発明の全体構成図を示す。１０１,１０２はPU（プロセッサユニット）を示し、１０３,１０４はそれらのCPU、１０５,１０６はメモリである。１０７はそれらを結ぶ通信路（ＰＵを相互接続できるネットワークであればよい）である。PUの数は任意であるが、説明のために２つのPUからなる並列計算機を示している。１０８,１０９はOS(オペレーティングシステム)である。ユーザプログラムを実行する際には、まずメモリ上にユーザプログラムが図１に示されない補助記憶装置等からローディングされる(１１０,１１１)。なお、ユーザプログラムはあらかじめ本発明のメッセージパッシングライブラリ(１１２,１１３)とリンクされているものとする。メッセージパッシングライブラリ中には他PUからリモートメモリ書き込み可能なリモートメモリ書き込み領域(１１４,１１５)が設けられている。さらに、リモートメモリ書き込み領域内部は通信相手PUごとにあらかじめアドレスが割り当てられている

静的バッファ(１１６,１１７)とアドレスが動的に変化する動的バッファ(１１８,１１９)が存在する。

【００１５】

以上の構成要素のうち、メッセージパッシングライブラリが本発明の特徴をなす構成要素である。以下、詳細に説明する。

【００１６】

（Ａ）バッファの構成

図３、図４を用いて、本発明におけるメッセージパッシング用のバッファ構成について説明する。上述したように本発明では、静的バッファ(３０１)と動的バッファ(４０１)の二種類のバッファを用いる。なお、静的バッファ３０１は図１の静的バッファ１１６、１１７に相当し、動的バッファ４０１は図１の動的バッファ１１８、１１９に相当する。

【００１７】

まず、静的バッファは通信相手PU（＃０、＃１、＃２、・・・＃ｎ）ごとに複数のブロック（図３ではPU＃１に対して６個のブロックが示される）が用意されている。静的バッファの各ブロックは、大きく分けてヘッダ(３０２)とメッセージ本体(３０３)の二つに分かれており、さらにヘッダ内にはtag(３０４)、length(３０５)、first address(３０６)、last address(３０７)の情報が含まれている。tagは対応する送受信の組を選択するための識別子、lengthは通信するメッセージの長さ、first addressは動的バッファを割り当てた時の先頭アドレス、last addressは動的バッファを割り当てた時の最終アドレスを格納するための領域である。なお、通信路（ネットワーク）内での送信先ＰＵおよび送信元ＰＵの識別情報は別途管理され、メッセージはネットワーク内を転送されるものとする。静的バッファは各PUごとに送信用(３０１)と受信用(３０９)の同一形状のバッファが用意されており、バッファの使用状況などの情報が通信相手PUと共有化されている。静的バッファは通信相手PUごとにあらかじめ設定されており初期化の時点でお互いのPUがアドレスを知ることが出来る。送信側は送信用バッファに空きがある限りは常に受信側の受信用バッファにデータの転送を行う事が出来る。

【００１８】

動的バッファは、通信相手PUごとに区別されていない複数のブロック(４０１)からなる。各ブロックは、ヘッダ(４０２)とメッセージの本体(４０３)とに分けられ、ヘッダは受信側がデータが到着したかどうかの確認を行うために使用される。なお、ヘッダ部分の領域の構成は静的バッファの構成と同じであり、送信先ＰＵおよび送信元ＰＵのネットワーク内での識別情報（アドレス）は別途管理されるものとする。さらにメッセージ長に合わせたバッファ量を選択するために、各ブロックは幾つかで束になって管理されている（図４の複数ブロック４０１ではでは、この束を太線の枠で示している）。その束ごとに管理ヘッダ(４０４)に登録されており、受信側PUはメッセージ長に合わせて最適なサイズのバッファ束を取得する（図４の例では１ブロックの束と４ブロックの束がそれぞれ複数面用意されており、メッセージが１ブロックのサイズより小さい時は１ブロックの束が、それより大きい時には４ブロックの束が取得される）。受信側PUは取得したバッファ束の先頭アドレスと最終アドレスを静的バッファのヘッダ内のfirst address、last addressに格納し送信側PUへと送信することになる。また、送信側は動的バッファを２面用意しており(４０５)、これを交互に使用することでパイプライン処理が可能となる。パイプライン処理の詳細については後述する。

【００１９】

（Ｂ）通信プロトコル

一般にメッセージ長が短い時には、より高速にメッセージの転送が行われる（レイテンシが低い）ことが求められ、一方メッセージ長が長い時には、単位時間当りにより大量のメッセージの転送が行われる（スループットが大きい）ことが求められる。この２つの必ずしも両立しない要求を満たすため、本発明ではメッセージ長が短いメッセージを送信する場合に使用するショートプロトコルとメッセージ長が長いメッセージを送信する場合に使用するロングプロトコルの２つの通信プロトコルを用意し、これを切り替えて用いること

で遅延の少ないデータ転送を実現する。

【００２０】

図５は、ショートプロトコルのタイミングチャートを表している。ショートプロトコルは、メッセージの転送に静的バッファを用いる。ユーザプログラムにより送信関数がコールされると(５０１)、送信側PUは静的バッファのヘッダとメッセージ本体を受信側に送信する(５０３)。一方、受信関数がコールされると(５０２)、受信側PUは静的バッファでメッセージを受け取り、送信側に受信完了通知を送信し(５０４)する。１往復のデータ通信でメッセージの通信を完了することができる。

【００２１】

しかし、リモートメモリ書き込み領域には限りがあるため、静的バッファの長さ、数量には制限が生じる。そのため全てのメッセージを静的バッファで送信すると大量のメッセージを通信する時には、静的バッファが空くのを待つ必要があり、逆に通信速度が落ちてしまう。そのためメッセージ長が長い時には、PUごとに区別されていない、それがゆえに大量に用意の出来る動的バッファを用いてメッセージ転送を行う。これが、ロングプロトコルである。静的バッファのメッセージ本体部分の長さを境界として、静的バッファの容量より少ないメッセージ長のメッセージを送信する場合にショートプロトコルを用い、静的バッファの容量より大きいメッセージ長のメッセージを送信する場合にロングプロトコルを用いるように、制御される。

【００２２】

図６は、ロングプロトコルのタイミングチャートを表している。ユーザプログラムにより送信関数がコールされると(６０１)、送信側PUはまず、送信するデータ長の長さ（送信するデータのデータ量）を検出し、このデータ量が静的バッファの容量より大きいと、ロングプロトコルを用いると判定する。静的バッファの容量より送信するデータ量が小さい場合は、前述のショートプロトコルを用いる。ロングプロトコルの場合、静的バッファのヘッダを受信側に送信する(６０３)。一方、受信関数がコールされると(６０２)受信側PUは静的バッファでメッセージの情報を受け取り、それに合わせた動的バッファのアドレス情報を静的バッファを用いて送り返し(６０４)、以後送信側PUが受け取ったバッファ情報に基づきメッセージを送信し(６０５)、最後に受信側PUが送信側に受信完了通知を送信する(６０６)。２往復のデータ転送が必要でありショートプロトコルに比較してレイテンシは高くなるが、動的バッファは静的バッファに比べ大量のデータ転送を可能とするためスループットを大きくすることが可能である。

【００２３】

以下、各プロトコルの動作を詳細に説明する。

まず、ショートプロトコルの動作を図７を用いて説明する。送信側PUは、静的バッファのヘッダにメッセージ長と識別子を格納する(７０１)。さらにメッセージをユーザ領域からメッセージ本体部にコピーする(７０２)。次いで、送信側から受信側へ静的バッファのリモートメモリ書き込みを行う(７０３)。一方、受信側PUは、メッセージを静的バッファで受け取り(７０５)、そこからユーザ領域へとコピーし、受信が完了する(７０６)。最後に受信側は受信完了通知を静的バッファを使い送り出し(７０７)、それを受けて送信側も処理を終了する(７０４)。

【００２４】

次に、ロングプロトコルの動作を図８を用いて説明する。まず送信側PUはショートプロトコルと同様、静的バッファのヘッダにメッセージ長と識別子を格納する(８０１)。ロングプロトコルの場合静的バッファではメッセージを送りきれないためヘッダのみを受信側へリモートメモリ書き込みする(８０２)。受信側PUは静的バッファでメッセージの情報を受け取る(８０７)と、lengthに合わせて適当な長さの動的バッファを確保しその先頭アドレスと最終アドレスを取得する(８０８)。それらのアドレスを静的バッファのfirst address、last addressにセットして、送信側に送り返す(８０９)。送信側はバッファ情報を受け取り(８０３)、全てのメッセージを送信するまでループを繰り返し(８０４)、動的バッファのブロックを単位として、ユーザ領域からバッファにコピーし送信する(８０５)。さ

らに受信側も全てのメッセージを受信するまでループを繰り返し(８１０)、受信したブロックからユーザ領域にメッセージをコピーする(８１１)。最後に受信側は受信完了通知を静的バッファを使い送り出し(８１２)、それを受けて送信側も処理を終了する(８０６)。

【００２５】

（Ｃ）ロングプロトコルにおけるパイプライン転送

リモートメモリ書き込みを用いたメッセージパッシングでは、ユーザ領域からリモートメモリ書き込み領域へのコピー、送信側から受信側へのリモートメッセージ転送、リモートメモリ書き込み領域からユーザ領域へのコピー、とメッセージの転送が３回必要となる。したがって少なくともリモートメモリ書き込みの約３倍の時間が必要となる。そこで本発明では送信側の動的バッファを２面用意し、パイプライン処理を行う事で性能向上を図る。

【００２６】

以下に図９を用いてパイプライン処理時の動作を説明する。図９において送信側PUのリモートメモリ書き込み領域内のバッファ(２０２)は、図４における２面ある送信側の動的バッファ(４０５)を、受信側PUのリモートメモリ書き込み領域内のバッファ(２０３)は、受信側の動的バッファ(４０１)を簡易化して表している。受信側の動的バッファは多面用意されているが、ここではABCDのデータを転送するのに必要な４面のみ表記している。ステップ１で送信側においてユーザ領域からリモートメモリ書き込み領域へメッセージAのコピーを行う(９０１)。次にステップ２で、メッセージAを送信側から受信側へリモートメッセージ転送で送信する(９０２)と同時に、メッセージBをリモートメモリ書き込み領域へコピーする(９０３)。ステップ３では、受信側でメッセージAをリモートメモリ書き込み領域からユーザ領域へコピーし(９０４)、メッセージBを送信側から受信側へリモートメモリ転送し(９０５)、送信側でメッセージCをユーザ領域からリモートメモリ書き込み領域へコピーする(９０６)。ステップ４では、受信側でのメッセージBのコピー(９０７)、メッセージCの送信側から受信側へのリモートメモリ書き込み転送(９０８)、送信側でのメッセージDのコピー(９０９)を同時に行う。

【００２７】

図１０は、パイプライン処理の送信側PUと受信側PUごとの動作を示す。送信側では、まずデータAをユーザ領域からリモートメモリ書き込み領域へメモリコピーし(１００１)、次にデータAを受信側に送信すると同時にデータBのメモリコピーを行い(１００２)、以下順次同様の動作が続き(１００３,１００４)、最後にデータDの送信が行われる(１００５)。一方、受信側では、まずデータAを送信側から受信し(１００６)、次にデータBを受信すると同時にデータAのメモリコピーを行い(１００７)、以下順次同様の動作が続き(１００８,１００９)、最後にデータDのメモリコピーが行われる(１０１０)。

【００２８】

（Ｄ）メッセージパッシングライブラリのインタフェース

メッセージパッシングライブラリは、ユーザプログラムの中から関数コールの形でメッセージパッシングを行うための関数群である。以下に、本発明におけるメッセージパッシングライブラリの関数のインタフェースとその動作を説明する。なお、関数名称、引き数名称などは任意であり、必ずしもここで説明する仕様と同じである必要はない。

【００２９】

(１)Init()

本関数中で、メッセージパッシングライブラリは必要な初期化操作を行う。メッセージパッシングライブラリの使用時には、全てのPUが必ず最初に本関数をコールしなければならない。本関数がユーザプログラムからコールされると、リモートメモリ書き込み領域に静的バッファ(図１：１１６,１１７)と動的バッファ(図１：１１８,１１９)を作成し、各バッファの初期化を行う。静的バッファは各PUごとにアドレスが固定であり、全てのPUは静的バッファの送信時の相手先アドレスをこの初期化時に通信しあうことが出来る。以下に挙げる関数は、初期化関数をコールした後にのみ使用する事が出来る。

【００３０】

(２)Send(buf, dest, tag, length)
ここで、bufは送信するメッセージの格納されたユーザメモリの先頭アドレスで、destは送信先PU番号（ネットワーク内での送り先ＰＵを識別する情報）、tagはメッセージの識別子、lengthはメッセージの長さを表す。ユーザが本関数をコールすると、ライブラリはlengthによってショートプロトコル(図７：７０１～７０４)かロングプロトコル(図８：８０１～８０６)を用いてメッセージの送信を行う。
【００３１】
(３)Recv(buf, src, tag)
本関数のbufは受信したメッセージを格納するユーザメモリの先頭アドレスで、srcは送信元PU番号（ネットワーク内でＰＵを識別する情報）、tagはメッセージ識別子を表す。ユーザが本関数をコールすると、送信元ＰＵ番号が一致した静的バッファのヘッダ部分でメッセージの長さを受け取り、それに合わせてショートプロトコル(図７：７０５～７０７)またはロングプロトコル(図８：８０７～８１２)でメッセージの受信を行う。
【００３２】
（Ｅ）ノンブロッキング動作における順序性の保証
メッセージパッシングライブラリにおける送受信には、ブロッキング関数とノンブロッキング関数がある。ブロッキング関数とは、送受信関数がコールされてから送受信が完了するまでプログラムの動作をブロック(停止)する関数であり、ノンブロッキング関数とは、関数のコール後送受信が完了する前にリターンし、PUはその間に他の動作を行う事が可能な関数である。前項まではブロッキング関数を前提としていた。本項ではノンブロッキングを実現するための追加機構を説明する。ノンブロッキング関数では、送受信が完了する前に複数の送受信関数が発行される事がある。この時にtagの同じ送受信関数では、発行された順序で送信関数と受信関数が対応しない事がある。以下に本発明の送受信関数の順序性の保証法について図１１を用いて説明する。
【００３３】
本発明のメッセージパッシングライブラリは、送信時には、まず静的バッファのヘッダにメッセージ情報(tag, length)をセットして受信側へと連絡する(ショートプロトコル時にはメッセージの本体も同時に送信する)。この時に静的バッファのヘッダにnext(１１０１)というメンバを加え、このnextで次に送信する静的バッファのブロックを指定する。使用中の静的バッファの各ブロックは、nextによりチェーンでつながれており、チェーンの順に送信されることになる。受信側が受け取る静的バッファは送信側から送られたものであり、送信側と同様nextでつながっている。したがってチェーンの順で検索し、送信関数の発行順序を確定することが出来る。
【００３４】
また、送信側での送信関数発行時に静的バッファが空いていない時や、受信側での受信関数発行時に静的バッファのヘッダがまだ送られてきていない時には、送受信関数の順序を静的バッファのブロックの順序で表す事が出来ない。そこで本発明では、関数発行時にすぐに処理できない関数の順序を保持しておくために、未処理の関数の発行順序を管理するためのリクエストオブジェクトを導入する(１１０２,１１０３)。リクエストオブジェクトには、メッセージの情報を保持するtag(１１０４,１１０５)、dest(src)(１１０６,１１０７)、length(１１０８,１１０９)と順序を保持するnext(１１１０,１１１１)の計４つの要素を持つ。終了していない関数は静的バッファのブロックと同様、nextによってチェーンでつながれ、送信側、受信側、それぞれで、その順に処理される。静的バッファのブロックが順に処理された後は、リクエストオブジェクトの順に処理が進む。以上の方式によりノンブロッキング関数における順序性は保証される。
【００３５】
（Ｆ）ノンブロッキング動作の追加インタフェース
順序性が保証されれば、以下のインタフェースを追加することによってノンブロッキング関数を実現できる。
【００３６】

(１)Isend(buf, dest, tag, length)
各引き数はSendと同じ仕様である。ユーザが本関数をコールすると静的バッファに空きがある場合には使用するバッファをチェーンにつないでから転送し、静的バッファに空きがない場合にはリクエストオブジェクトを作成して本関数をチェーンにつなぐ。リクエストオブジェクトのチェーンは発行順に処理される。静的バッファを受信側に転送した後は、ショートプロトコル(図７：７０１～７０４)かロングプロトコル(図８：８０１～８０６)で非同期にデータが転送される。

【００３７】
(２)Irecv(buf, src, tag)
各引き数はRecvと同じ仕様である。ユーザが関数をコールするとリクエストオブジェクトを作成してチェーンにつなぐ。チェーンの先頭の関数から順に処理され、静的バッファのチェーンの先頭から順に対応するtagを持つ送信関数が発行されているかを検索する。対応する送信が検索された後は、ショートプロトコル(図７：７０４～７０７)かロングプロトコル(図８：８０７～８１２)で非同期にデータが転送される。

【００３８】
(３)Wait()
本関数は、ノンブロッキング関数の完了を待つための関数である。ノンブロッキング関数は、関数コール後すぐにリターンしてしまうため、関数がいつ完了するかユーザには分からない。そのためノンブロッキング関数の完了を明示するために本関数は使われる。本関数が発行されると、完了確認をしていないノンブロッキング関数が完了するまでPUはブロックされる。全ての関数が完了することで本関数も完了する。本関数完了後はまた新たにSend/Isend、Recv/Irecvが発行され通信が再開される。

【００３９】
【発明の効果】
本発明のリモートメモリ転送制御方式によれば、リモートメモリ書き込みを用いたデータ転送において、データの長さによってあらかじめPUごとにアドレスの割り当てられた領域を用いて転送するか、転送時に動的に割り当てられる領域を用いてパイプライン処理で転送するかを選択することができ、それによって高速にデータの転送が出来るようになる。図１２に示す通りパイプライン動作を導入すると、最初と最後の２回ずつを除き、３回のメッセージ転送が重なって生じる。したがって、従来のリモートメモリ書き込みを用いたメッセージパッシングに比べ、約３倍の性能が得られる。

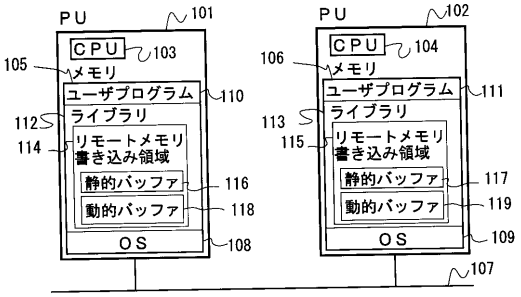
【００４０】
また、本発明のリモートメモリ転送制御方式によれば、送信関数が発行された順にデータが転送され、受信関数が発行された順に転送されたデータを受け取る事を保証することができる。これにより、ノンブロッキング動作を行う送受信関数におけるデータの順序性を保証することができるようになる。

【図面の簡単な説明】
【図１】本発明の実施例の全体構成図。
【図２】従来のリモートメモリ書き込みを用いたデータ転送制御方式。
【図３】静的バッファの説明図。
【図４】動的バッファの説明図。
【図５】ショートプロトコルのタイミングチャート。
【図６】ロングプロトコルのタイミングチャート。
【図７】ショートプロトコルのフローチャート。
【図８】ロングプロトコルのフローチャート。
【図９】パイプライン動作の説明図。
【図１０】パイプライン動作のフローチャート。
【図１１】順序性保証のための説明図。
【図１２】パイプライン動作の動作図。
【符号の説明】

１０１,１０２...要素計算機、１０３,１０４...CPU、１０５,１０６...メモリ、１０７...通信路、１０８,１０９...オペレーティングシステム、１１０,１１１...ユーザプログラム、１１２,１１３...メッセージ通信ライブラリ、１１４,１１５...データ転送用メモリ領域、１１６,１１７,３０１,３０９...静的バッファ、１１８,１１９,４０１,４０５,９１０,９１１...動的バッファ、２０１,２０４...ユーザバッファ、２０２,２０３...データ転送用バッファ、３０２...静的バッファのヘッダ、３０３,３０８...静的バッファのメッセージ本体、３０４...メッセージの識別子を格納する領域、３０５...メッセージの長さを格納する領域、３０６...確保した動的バッファの先頭アドレスを格納する領域、３０７...確保した動的バッファの最終アドレスを格納する領域、４０２...動的バッファにメッセージが到着しているかを確認するためのヘッダ、４０３...動的バッファのメッセージの本体、４０４...動的バッファの管理ヘッダ、１１０１...静的バッファの順序を格納する領域、１１０２,１１０３...リクエストオブジェクト、１１０４,１１０５...メッセージの識別子を格納するリクエストオブジェクトの領域、１１０６,１１０７...メッセージの送受信相手を格納するリクエストオブジェクトの領域、１１０８,１１０９...メッセージの長さを格納するリクエストオブジェクトの領域、１１１０,１１１１...リクエストオブジェクトの順序を格納する領域。

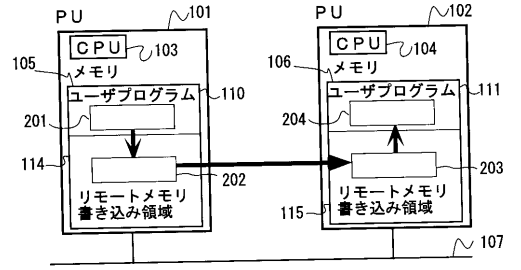
【図１】

図1



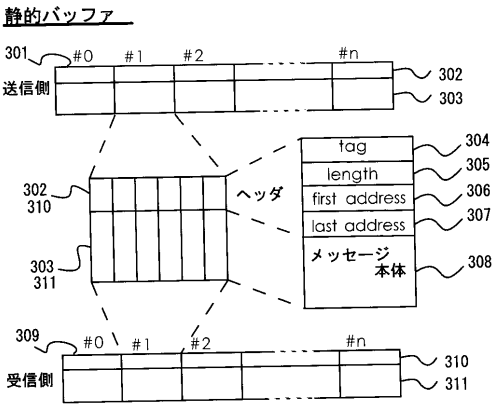
【図２】

図2



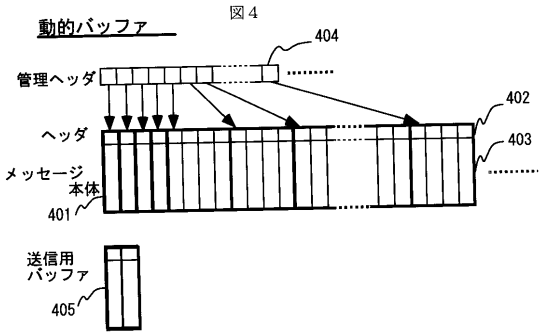
【図３】

図3



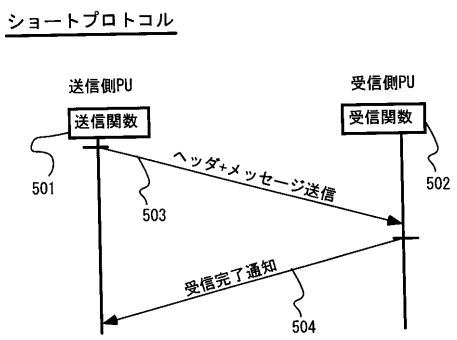
【図４】

図4



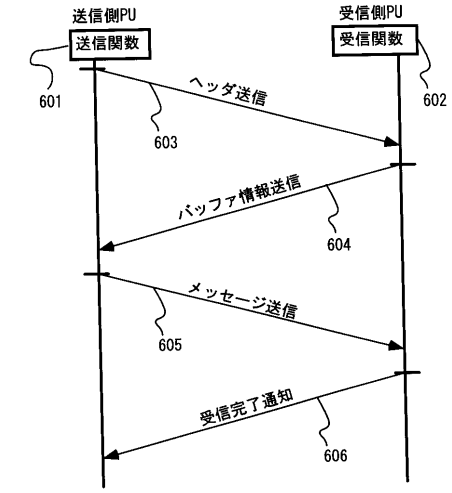
【図５】

図5



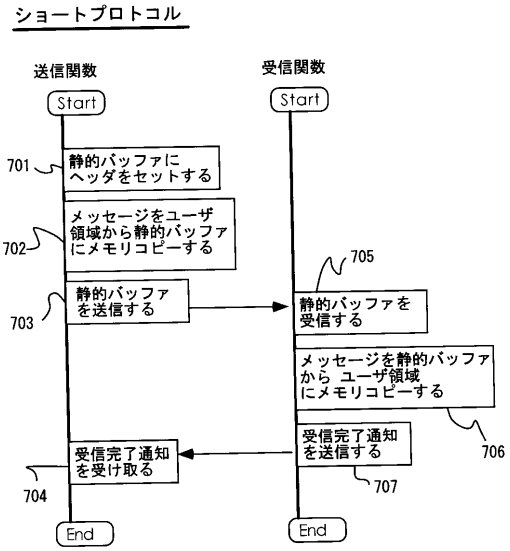
【図６】

図6



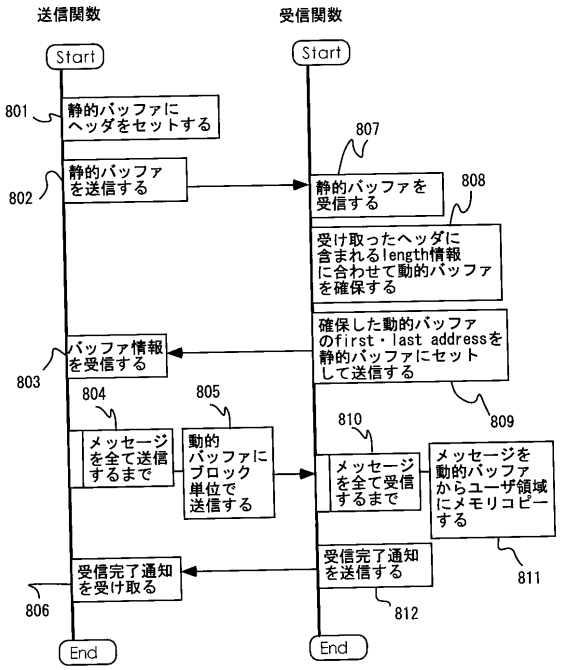
【図７】

図7



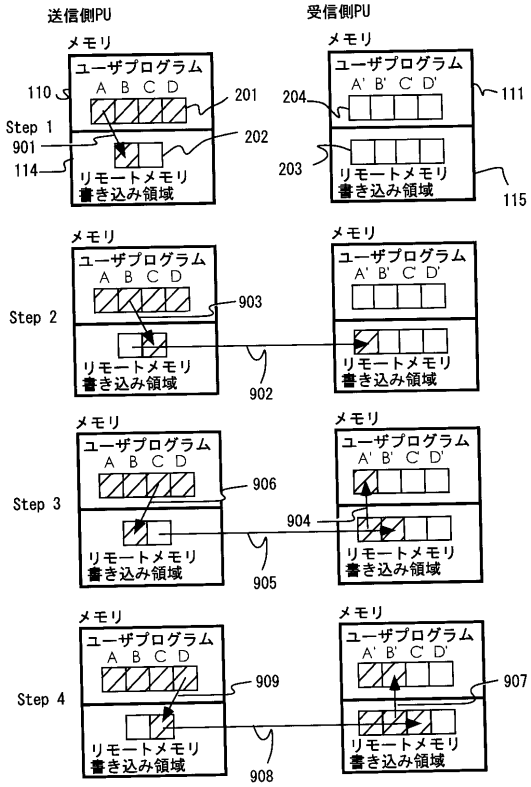
【図８】

図8



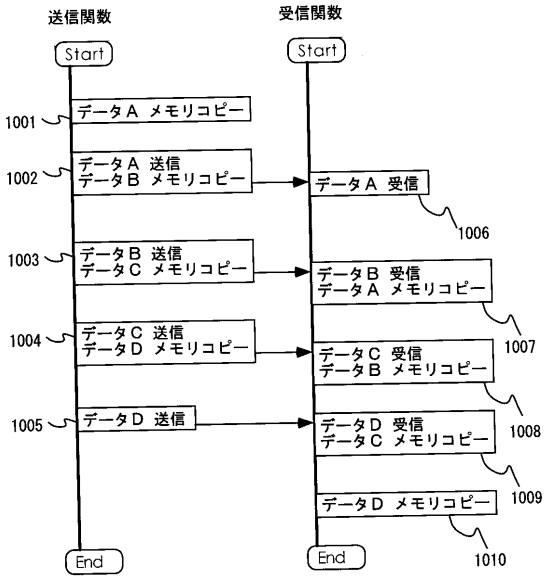
【図９】

図9



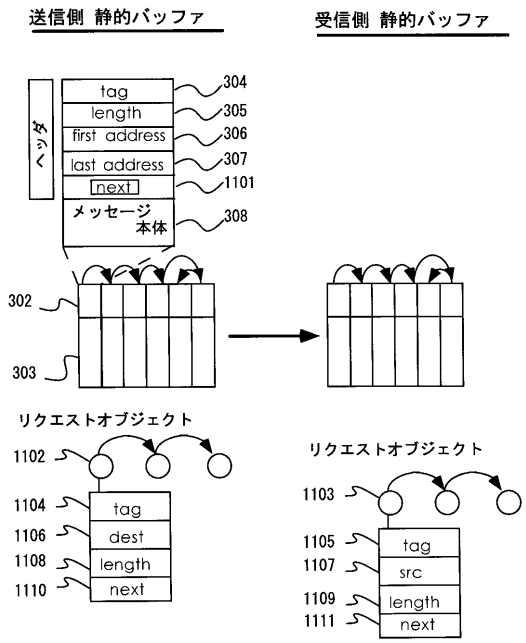
【図１０】

図１０



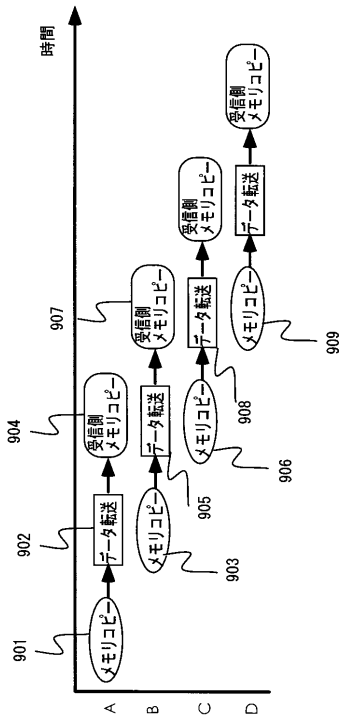
【図１１】

図１１



【図１２】

図１２



フロントページの続き

(72)発明者　今木　常之
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内

審査官　鳥居　稔

(56)参考文献　特開平０８－０１６５４０（ＪＰ，Ａ）
特開平０８－２６３４６１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 15/16-177
G06F 13/00