(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-165022
(P2010-165022A)
(43)公開日　平成22年7月29日(2010.7.29)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 15/167	(2006.01)	G06F	15/167	610A	5B045
G06F 9/54	(2006.01)	G06F	9/46	480A	
G06F 15/17	(2006.01)	G06F	15/17		

審査請求 未請求　請求項の数 10　ＯＬ　（全 24 頁）

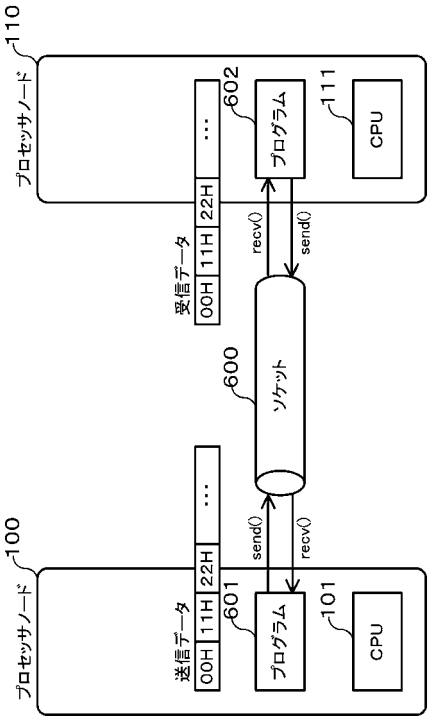
(21)出願番号　特願2009-4678(P2009-4678)
(22)出願日　平成21年1月13日(2009.1.13)

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号
(74)代理人　100110607
弁理士　間山　進也
(72)発明者　本橋　弘臣
東京都大田区中馬込１丁目３番６号　株式会社リコー内

Ｆターム(参考)　5B045 BB02 BB12 BB28 BB29 BB32 DD02

(54)【発明の名称】プロセッサ間通信装置、プロセッサ間通信方法、プログラムおよび記録媒体

(57)【要約】
【課題】各プロセッサノード間での共有メモリを用いた通信処理で発生する通信オーバーヘッドを低減し、高速通信が可能なプロセッサ間通信装置、プロセッサ間通信方法、プログラムおよび記録媒体を提供すること。
【解決手段】本発明に係るプロセッサ間通信装置は、第１および第２のプロセッサノードからアクセス可能に共有した共有メモリを備え、第１のプロセッサノードでのプログラムに従って、通信データを共有メモリに書き込む書込手段と、第１および第２のプロセッサノードを接続するソケットと、書込通知情報をソケットを介して第２のプロセッサノードに送信する第１の通知手段と、書込通知情報に応じて共有メモリから通信データを読み出す読出手段と、読出通知情報をソケットを介して第１のプロセッサノードに送信する第２の通知手段とを備えている。
【選択図】図６



【特許請求の範囲】
【請求項１】
第１および第２のプロセッサノードから構成され、前記第１および第２のプロセッサノードのそれぞれからアクセスすることが可能に共有して設けられた共有メモリを備えたプロセッサ間通信装置において、
前記第１のプロセッサノードで動作しているプログラムに従って、通信データを前記共有メモリに書き込む書込手段と、
前記書込手段により前記通信データが前記共有メモリに書き込まれたことを通知する書込通知情報を、ソケット通信により前記第２のプロセッサノードに送信する第１の通知手段と、
前記第１の通知手段により送信された前記書込通知情報に応じて、前記共有メモリから通信データを読み出す読出手段と、
前記読出手段により前記通信データが読み出されたことを通知する読出通知情報を、前記ソケット通信により前記第１のプロセッサノードに送信する第２の通知手段と
を備える、プロセッサ間通信装置。
【請求項２】
前記共有メモリは、
前記通信データに対して、循環的に書き込み処理を行うことが可能なリングバッファと、
前記リングバッファ内での通信データを書き込む際の開始位置を示すライトポインタと、
前記リングバッファ内での通信データを読み出す際の開始位置を示すリードポインタと
を含む、請求項１に記載のプロセッサ間通信装置。
【請求項３】
前記共有メモリは、前記第１および第２のプロセッサノード内にそれぞれ設けられたメインメモリの記憶領域の一部を用いて成る、請求項１または２に記載のプロセッサ間通信装置。
【請求項４】
前記書込手段により前記共有メモリに書き込まれた前記通信データのデータ量が、所定量に達した場合に、前記通信データが書き込まれたことを通知するデータ量通知情報を、前記ソケット通信を介して前記第２のプロセッサノードに送信するデータ量通知手段を備える、請求項１～３に記載のプロセッサ間通信装置。
【請求項５】
前記第１の通知手段が、前記ソケット通信により前記第２のプロセッサノードに前記書込通知情報を送信するための条件を設定する条件設定手段を備える、請求項１～４に記載のプロセッサ間通信装置。
【請求項６】
前記共有メモリ内のうち空き領域を検出して、これらの検出した各領域から、前記書込手段が通信データを書き込むための領域を確保して割り当てる割当手段を備える、請求項１～５に記載のプロセッサ間通信装置。
【請求項７】
前記割当手段により割り当てられた、前記書込手段が通信データを書き込むための領域のアドレスを通知するための割当通知情報を、前記ソケット通信を介して前記第２のプロセッサノードに送信する送信手段を備える、請求項１～６に記載のプロセッサ間通信装置。
【請求項８】
プロセッサ間通信装置が実行するプロセッサ間通信方法であって、前記プロセッサ間通信方法は、プロセッサ間通信装置が、
前記プロセッサ間通信装置に構成されている第１および第２のプロセッサノードのうち、前記第１のプロセッサノードで動作しているプログラムに従って、前記第１および第２

のプロセッサノードのそれぞれからアクセスすることが可能に共有して設けられた共有メモリに通信データを書き込むステップと、
前記通信データが前記共有メモリに書き込まれたことを通知する書込通知情報を、前記第１および第２のプロセッサノードを、前記プログラム間で通信データの送受信が可能に接続するソケット通信を介して、前記第２のプロセッサノードに送信するステップと、
前記送信された前記書込通知情報に応じて、前記共有メモリから通信データを読み出すステップと、
前記通信データが読み出されたことを通知する読出通知情報を、前記ソケット通信を介して前記第１のプロセッサノードに送信するステップと
を備える、プロセッサ間通信方法。
【請求項９】
プロセッサ間通信装置がプロセッサ間通信を行うための装置実行可能なプログラムであって、前記プログラムは、プロセッサ間通信装置を、
前記プロセッサ間通信装置に構成されている第１および第２のプロセッサノードのうち、前記第１のプロセッサノードで動作している動作プログラムに従って、前記第１および第２のプロセッサノードのそれぞれからアクセスすることが可能に共有して設けられた共有メモリに通信データを書き込む書込手段と、
前記書込手段により前記通信データが前記共有メモリに書き込まれたことを通知する書込通知情報を、前記第１および第２のプロセッサノードを前記動作プログラム間で通信データの送受信が可能に接続するソケット通信により、前記第２のプロセッサノードに送信する第１の通知手段と、
前記第１の通知手段により送信された前記書込通知情報に応じて、前記共有メモリから通信データを読み出す読出手段と、
前記読出手段により前記通信データが読み出されたことを通知する読出通知情報を、前記ソケット通信を介して前記第１のプロセッサノードに送信する第２の通知手段と
して機能させる、装置実行可能なプログラム。
【請求項１０】
請求項９に記載のプログラムを格納した情報処理装置可読な記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、複数のプロセッサノードから構成され、各プロセッサノードのそれぞれからアクセスすることが可能に共有して設けられた共有メモリを備えたプロセッサ間通信装置、プロセッサ間通信方法、プログラムおよび記録媒体に関する。
【背景技術】
【０００２】
従来より、複数のプロセッサノードを備えたプロセッサ間通信装置の１例である疎結合型のマルチプロセッサシステムの動作効率は、各プロセッサノード間の通信処理で発生する通信オーバーヘッドからのシステムパフォーマンスへの影響をどのようにして低減させるかに左右される。それは、マルチプロセッサシステムが実行する各処理のジョブのうち、プロセッサノードの負荷が大きいジョブが存在すると、このジョブを複数の各処理毎に細分化して各プロセッサノードにそれぞれ割り当てたとしても、大抵はそれぞれの細分化されたタスクなどの各処理の間で何らかの関連性や依存性があるために、各段階の処理を経ていくに従って各プロセッサノード間で演算結果や計算結果などの情報を相互に受け渡す通信処理を行うからである。
【０００３】
このような各プロセッサノード間での通信処理では、従来ではＥｔｈｅｒｎｅｔ（登録商標）やＩｎｉｆｉＢａｎｄ，Ｍｙｒｉｎｅｔなどのインタフェースが利用されてきた。しかし、Ｅｔｈｅｒｎｅｔ（登録商標）には、送信元のプロセッサノードが動作している処理でデータを送信してから、受信側のプロセッサノードが動作している処理でデータを

受け取るまでの時間、即ち、通信時のレイテンシが長い、また、ＴＣＰ／ＩＰなどのプロトコル処理が重いという欠点があった。

【０００４】

通信時のレイテンシが長いと、プロセッサノード間で頻繁にデータを相互に受け渡す場合には、通信オーバーヘッドが増大してシステム全体のパフォーマンス、動作効率が低下する。また、プロトコル処理が重いと、貴重なＣＰＵ性能が本来の演算処理や計算処理などの本来の目的以外の処理で浪費されてしまう。

【０００５】

一方、ＩｎｉｆｉＢａｎｄやＭｙｒｉｎｅｔは、レイテンシが短く、プロトコル処理がハードウェア化されているためにＣＰＵの負担が軽いという利点があるが、これらのインタフェースカードは、Ｅｔｈｅｒｎｅｔ（登録商標）と比べると高機能・高性能であるがゆえに非常に高価であり、組み込み機器ではローコストの実現が困難であった。

【０００６】

また、マルチプロセッサシステムで用いられている組み込み機器では、ローコストが要求されていることもありＣＰＵの個数が多くても高々２～４個程度に留まっていることから、各プロセッサノード間での通信処理は共有メモリを利用して行われる場合が多かった。このような場合の共有メモリには、従来より、デュアルポートメモリとして参照されるメモリ素子が使用されていたが、このデュアルポートメモリの機能では、１個に対して２個のＣＰＵまでに限って接続可能であるという制限があった。

【０００７】

最近ではデュアルポートメモリ以外にも、２個のプロセッサノードのＰＣＩバス同士を接続するＰＣＩ－ＰＣＩブリッジや、２個以上の複数のプロセッサノードのＰＣＩ－Ｅｘｐｒｅｓｓバスをクロスバーで接続するＰＣＩ－Ｅｘｐｒｅｓｓスイッチを用いたＬＳＩが使用されている。

【０００８】

これらのＰＣＩ－ＰＣＩブリッジやＰＣＩ－Ｅｘｐｒｅｓｓスイッチでは、通常１つのプロセッサノードが占有利用するメインメモリに対して、他のプロセッサノードがダイレクトにアクセスすることができるため、ローコストで容易に共有メモリの機能を有することができる。また、これらのＰＣＩ－ＰＣＩブリッジやＰＣＩ－Ｅｘｐｒｅｓｓスイッチは、２個以上のＣＰＵを備えたマルチプロセッサシステムで利用されている。

【０００９】

特許第３７４３３８１号公報（特許文献１）に記載されたコンピュータシステムでは、複数のホストと、これらのホストのいずれからもアクセス可能な共有メモリと、ホスト間をつなぐ通信経路とを備えるコンピュータシステムにおいて、自ホスト内の、ホスト間で連携して動作するプログラムから発行されるホスト間処理同期要求を受理し、その旨を示す情報を共有メモリに記録し、前記旨を前記通信経路を介して他ホストに通知し、自ホストのホスト間処理同期要求の受理や他ホストから通知される、ホスト間処理同期要求の受理通知を契機とし、共有メモリを参照してホスト間処理同期成立の判断・認識を行う各ホスト内のホスト間処理同期手段と、ホスト間の通信が失われた場合にも対処できるように、他ホストからのホスト間処理同期要求の受理通知がなくても、一定時間毎に共有メモリの参照に基づくホスト間処理同期成立判断を前記ホスト間処理同期手段に行わせるべく処理同期確認要求を発行する各ホスト内のタイマ起動手段とを有している。

【発明の概要】

【発明が解決しようとする課題】

【００１０】

しかし、共有メモリは、他の一般的な通信処理と、ソフトウェア上での利便性が全く異なっており、共有メモリを利用するためには各プロセッサノードのプログラム間での同期や排他などの仕組みをプログラマが何らかの方法で実現しなければならないといった利便性での問題があった。さらには、密結合型のマルチプロセッサシステム向けに開発されソケット通信を利用しているソフトウェアを、共有メモリを備えた疎結合型のマルチプロセ

ッサシステムに移行しようとする場合には、プログラム内のソースコード中でソケット通信を利用している箇所の記述を、共有メモリを利用する処理に書き換えなければならないため、プログラマにとってソフトウェアの開発負担が非常に増大するという問題があった。

【００１１】

本発明は、このような課題に鑑みてなされたものであり、各プロセッサノード間での共有メモリを用いた通信処理で発生する通信オーバーヘッドを低減し、高速通信が可能なプロセッサ間通信装置、プロセッサ間通信方法、プログラムおよび記録媒体を提供することを目的とする。

【課題を解決するための手段】

【００１２】

以上の課題を解決するために本発明では、第１および第２のプロセッサノードから構成され、第１および第２のプロセッサノードのそれぞれからアクセスすることが可能に共有して設けられた共有メモリを備えたプロセッサ間通信装置において、第１のプロセッサノードで動作しているプログラムに従って、書込手段が通信データを前記共有メモリに書き込むと、第１および第２のプロセッサノードをプログラム間で通信データの送受信が可能に接続するソケット通信を介して、第１の通知手段が書込通知情報を第２のプロセッサノードに送信する。そして、読出手段が、第１の通知手段により送信された書込通知情報に応じて共有メモリから通信データを読み出し、第２の通知手段が、読出通知情報をソケット通信を介して第１のプロセッサノードに送信する。

【００１３】

このため、第１および第２の各プロセッサノード間で、同一の共有メモリを共有利用することによって、この共有メモリが分散共有メモリとして機能する。また、ソケットを介して書込通知情報や読出通知情報を送受信することによって、この共有メモリが複数のプロセッサノード間で利用できるというソケット通信の特徴と、広帯域であるという共有メモリの特徴とをそれぞれ有し、通信オーバーヘッドを低減し、高速通信を可能とすることができる。

【発明の効果】

【００１４】

本発明によれば、各プロセッサノード間での共有メモリを用いた通信処理で発生する通信オーバーヘッドを低減し、高速通信が可能なプロセッサ間通信装置、プロセッサ間通信方法、プログラムおよび記録媒体を提供することができる。

【図面の簡単な説明】

【００１５】

【図１】第１の実施形態におけるマルチプロセッサシステムの全体構成を示す説明図である。

【図２】第２の実施形態におけるマルチプロセッサシステムの全体構成を示す説明図である。

【図３】第２の実施形態におけるマルチプロセッサシステムのメインメモリの記憶領域の使用状態を示す説明図である。

【図４】第３の実施形態におけるマルチプロセッサシステムの全体構成を示す説明図である。

【図５】第１の実施形態におけるマルチプロセッサシステムのプロセッサノードから他のプロセッサノード内の共有メモリに対してＣＰＵが書き込みを行った処理を示す説明図である。

【図６】第１の実施形態におけるマルチプロセッサシステムの複数のプロセッサノード間がソケットで接続されこのソケット６００を利用して通信を行っている処理を示す説明図である。

【図７】第１の実施形態におけるマルチプロセッサシステムの複数のプロセッサノードのうち、２つのプロセッサノードが通信を行っている処理を示す説明図である。

【図８】第１の実施形態におけるマルチプロセッサシステムの複数のプロセッサノードのうち、２つのプロセッサノードが通信を行っている処理を示す説明図である。
【図９】第１の実施形態におけるマルチプロセッサシステムの複数のプロセッサノード間がソケットで接続されている場合の通信データの送受信処理を示すシーケンス図である。
【図１０】第１の実施形態におけるマルチプロセッサシステムの通信データの送受信処理のコネクション確立後の具体的な例を示すシーケンス図である。
【図１１】第１の実施形態におけるマルチプロセッサシステムの通信データの送受信処理のコネクション確立後の他の具体的な例を示すシーケンス図である。
【図１２】第１の実施形態におけるマルチプロセッサシステムのクライアントで実行されるコネクション確立および切断の処理を示すシーケンス図である。
【図１３】第１の実施形態におけるマルチプロセッサシステムのソケットの動作を指示するためのソケット指示情報を記憶したテーブルを示す説明図である。
【図１４】第１の実施形態におけるマルチプロセッサシステムのソケット指示情報に従って共有メモリに対して行った処理の例を示す説明図である。
【図１５】第１の実施形態におけるマルチプロセッサシステムのソケット指示情報に従って共有メモリに対して行った処理の他の例を示す説明図である。
【図１６】第１の実施形態におけるマルチプロセッサシステムの共有メモリ内の記憶領域で割り当てられたメモリウィンドウのデータ構成を示す説明図である。
【図１７】第１の実施形態におけるマルチプロセッサシステムの共有メモリ内のリングバッファに通信データを用いて行った処理を示す説明図である。
【図１８】第１の実施形態におけるマルチプロセッサシステムの複数のプロセッサノードのうち２つのプロセッサノードが共有メモリに通信データを用いて行った処理を示す説明図である。
【図１９】第１の実施形態におけるマルチプロセッサシステムの複数のプロセッサノードのうち２つのプロセッサノードが共有メモリに通信データを用いて行った他の処理を示す説明図である。
【図２０】第１の実施形態におけるマルチプロセッサシステムのプロセッサノードのハードウェア構成を示す説明図である。
【発明を実施するための形態】
【００１６】
〔第１の実施形態〕
以下、本発明に係る第１の実施形態をもって説明するが、本発明は、実施形態に限定されるものではない。図１は、本発明によるプロセッサ間通信装置の１つの例であるマルチプロセッサシステム１０の第１の実施形態での全体構成を示す説明図である。このマルチプロセッサシステム１０は、例えば共有メモリを備えた疎結合型のマルチプロセッサシステムであり、複数のプロセッサノードから構成され、共有メモリが各プロセッサノードのそれぞれからアクセス可能に共有して設けられている。
【００１７】
マルチプロセッサシステム１０は、図１に示すように、複数のプロセッサノード１００、１１０、１２０と、これらの各プロセッサノード１００、１１０、１２０間でデータなどの送受信を行うためのネットワークの１例であるＥｔｈｅｒｎｅｔ（登録商標）１３０と、各プロセッサノード１００、１１０、１２０のそれぞれからアクセス可能に共有して設けられた共有メモリ１４０とから構成されている。
【００１８】
プロセッサノード１００は、アプリケーションなどの各種の処理を行うためのプログラムを実行するＣＰＵ１０１と、ＣＰＵ１０１の処理に従って後述するようにメインメモリ１０３にアクセスしデータなどの読み出しを行うメモリコントローラ１０２と、ＣＰＵ１０１の処理によって得られた各種のデータを記憶するメインメモリ１０３とを備えている。
【００１９】

ＣＰＵ１０１は、予め用意されたプログラムを実行することにより、例えばアプリケーション上での各種の処理を行い、メモリコントローラ１０２や後述するホスト－ＰＣＩブリッジ１０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５を介して外部から入力された通信データを用いて演算処理や判定処理などを行う。

【００２０】

メモリコントローラ１０２は、メインメモリ１０３にアクセスして、ＣＰＵ１０１での処理によって得られた演算結果や判定結果などのデータをメインメモリ１０３内に形成された共有メモリに書き込んだり、また、メモリコントローラ１０２は、この共有メモリから通信データを始めとする各データを読み出し、ホスト－ＰＣＩブリッジ１０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５、Ｅｔｈｅｒｎｅｔ（登録商標）１３０を介して他のプロセッサノード１１０、１２０に送信するための処理を行う。

【００２１】

また、プロセッサノード１００は、プロセッサノード１００および外部との間のデータの入出力を行うホスト－ＰＣＩブリッジ１０４と、Ｅｔｈｅｒｎｅｔ（登録商標）１３０に接続されＥｔｈｅｒｎｅｔ（登録商標）１３０を介して外部との間でデータの送受信を行うＥｔｈｅｒｎｅｔ（登録商標）カード１０５とを備えている。

【００２２】

ホスト－ＰＣＩブリッジ１０４は、プロセッサノード１００の内部側がメモリコントローラ１０２に接続され、外部側がＥｔｈｅｒｎｅｔ（登録商標）カード１０５、共有メモリ１４０に接続されている。ホスト－ＰＣＩブリッジ１０４は、外部側の接続対象をＥｔｈｅｒｎｅｔ（登録商標）カード１０５または共有メモリ１４０に切り替えて、プロセッサノード１００およびＥｔｈｅｒｎｅｔ（登録商標）１３０間と、プロセッサノード１００および他のプロセッサノード１１０、１２０間でのデータの入出力を行う。

【００２３】

Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５は、例えばプロセッサノード１００のＰＣＩカードスロットルに装着されてＥｔｈｅｒｎｅｔ（登録商標）１３０と直接接続され、Ｅｔｈｅｒｎｅｔ（登録商標）１３０を介して外部との間でデータの送受信を行う。

【００２４】

Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５は、他のプロセッサノード１１０、１２０のうちのいずれかから要求された通信データを要求するための要求情報を受信する。また、Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５は、メモリコントローラ１０２により通信データが共有メモリ１４０に書き込まれたことを通知する書込通知情報を、他のプロセッサノード１１０、１２０にそれぞれ送信したり、メモリコントローラ１０２により通信データが読み出されたことを通知する読出通知情報を、他のプロセッサノード１１０、１２０にそれぞれ送信したり、また、他のプロセッサノード１１０、１２０のＥｔｈｅｒｎｅｔ（登録商標）カード１０５により送信された読出通知情報に従って、メモリコントローラ１０２により読み出された通信データを、通信データを要求したプロセッサノードに送信する処理を行う。

【００２５】

プロセッサノード１１０、１２０は、プロセッサノード１００と同様の構成となっており、ＣＰＵ１０１、メモリコントローラ１０２、メインメモリ１０３、ホスト－ＰＣＩブリッジ１０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５と同様の機能を有するＣＰＵ１１１、１２１、メモリコントローラ１１２、１２２、メインメモリ１１３、１２３、ホスト－ＰＣＩブリッジ１１４、１２４、Ｅｔｈｅｒｎｅｔ（登録商標）カード１１５、１２５を備えている。

【００２６】

図２０は、プロセッサノード１００のハードウェア構成を示す説明図である。プロセッサノード１００は、図２０に示すように、ＣＰＵ１０１にメモリコントローラ１０２が接続され、このメモリコントローラ１０２に対してメインメモリ１０３がデータの書き込みおよび読み出しが可能に、また、ホスト－ＰＣＩブリッジ１０４が通信データなどの各デ

ータの送受信が可能にそれぞれ接続されている。
【００２７】
また、ホスト－ＰＣＩブリッジ１０４には、複数種類のＰＣＩデバイス機器が接続可能なＰＣＩバス４０１が接続され、このＰＣＩバス４０１にＰＣＩバスを介して共有メモリ１４０に接続されたＰＣＩ－ＰＣＩブリッジ４０２と、複数種類の各ＰＣＩデバイス４０３と、Ｅｔｈｅｒｎｅｔ（登録商標）１３０に接続されたＥｔｈｅｒｎｅｔ（登録商標）カード１０５とがそれぞれ接続されている。
【００２８】
メモリコントローラ１０２は、ＣＰＵ１０１のホストバスに接続され、ＣＰＵ１０１からメインメモリ１０３やＥｔｈｅｒｎｅｔ（登録商標）カード１０５、ＰＣＩ－ＰＣＩブリッジ４０２、各種ＰＣＩデバイス４０３に対するリードまたはライトの要求を受け取ると、対象のデバイスに対してリクエストを振り分ける役目がある。マルチプロセッサシステム１００の具体的な例としてパソコンを用いた場合、一般的には例えばＩｎｔｅｌ製のｘ８６　ＣＰＵを搭載しているが、その際にはマザーボードというメイン基板の上にノースブリッジやＭＣＨ（Ｍｅｍｏｒｙ　Ｃｏｎｔｒｏｌｌｅｒ　Ｈｕｂ）として参照されるチップセットが搭載されており、図２０のメモリコントローラ１０２はこれらのチップセットに相当している。
【００２９】
一方、ホスト－ＰＣＩブリッジ１０４は、サウスブリッジやＩＣＨ（Ｉ／Ｏ　Ｃｏｎｔｒｏｌｌｅｒ　Ｈｕｂ）として参照されるチップセットに相当し、ＣＰＵ１０１から各種Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５、ＰＣＩ－ＰＣＩブリッジ４０２、各種ＰＣＩデバイス４０３に対するアクセス要求を発行したり、あるいはＰＣＩデバイス４０３からメインメモリ１０３に対するＤＭＡリクエストを受け取ってメモリコントローラ１０２に送信する機能を有する。
【００３０】
図５は、例えばプロセッサノード１１０から他のプロセッサノード１００内の共有メモリ１４０に対してＣＰＵ１０１が書き込みを行った処理を示す説明図である。図５に示すように、プロセッサノード１００のメインメモリ１０３の一部の記憶領域がプロセッサノード１１０のメインメモリ１１３内のメモリウィンドウにマッピングされている。この時に、プロセッサノード１１０で動作しているプロセスがメモリウィンドウの記憶領域に対して書き込みを行うと、この書き込まれたデータと記憶領域を示すアドレスがプロセッサノード１００に送信され、リモートメモリとして利用されているプロセッサノード１００内のメインメモリ１０３に対して書き込み処理が行われる。そのメインメモリ１０３に対してプロセッサノード１００で動作しているプロセスが読み出しを行うことにより、プロセッサノード１１０からプロセッサノード１００に対して通信データを伝達することが可能となる。
【００３１】
図５は、別々のプロセッサノードで動作しているプロセスの間で同一の共有メモリ１４０を共有利用することでデータの伝達を行うことが可能となっているが、このようなメモリを分散共有メモリとして参照する。分散共有メモリは、複数のプロセッサノード間で利用できるというソケット通信の特徴と、広帯域であるという共有メモリの特徴とをそれぞれ有している。
【００３２】
マルチプロセッサシステム１０が備えているＯＳのうち、例えばＵＮＩＸ（登録商標）のような各プロセッサノード１００、１１０、１２０でのプログラム内のプロセス間でのメモリ保護機能を持っているＯＳではプロセス間でグローバル変数等を用いてデータの伝達を行うことができない。このようなＯＳとしては、プロセス間でデータの送受信などのコミュニケーションを行うための機能として、ソケット通信や共有メモリといったＩＰＣ（Ｉｎｔｅｒ－Ｐｒｏｃｅｓｓ　Ｃｏｍｍｕｎｉｃａｔｉｏｎ）を実現する手段を備えている。ソケット通信は、ネットワーク透過な機能なので、各プロセッサノード１００、１

１０、１２０内のみならず、各プロセッサノード１００、１１０、１２０間でデータをやり取りする場合にも使用することができるという長所があるが、その反面、通信オーバーヘッドが多いために大量のデータ送受信の際には処理効率が低下するという欠点もある。共有メモリ１４０は、通信オーバーヘッドが低く大量のデータ送受信でも効率良く実行できるが（広帯域）、通常はそれぞれのプロセッサノード１００、１１０、１２０内でしか利用できないため、複数のプロセッサノード１００、１１０、１２０によって構成されるマルチプロセッサシステムではあまり利用されることがなかった。

【００３３】

図６は、複数のプロセッサノード１００、１１０、１２０間がソケット６００で接続され、例えばプロセッサノード１００で動作しているプログラムがこのソケット６００を利用してソケット通信を行っている処理を示す説明図である。ここで、ソケット６００は、複数のプロセッサノード１００、１１０、１２０のそれぞれの間でＩＰアドレス、ポート番号が設定されて、このソケット６００によって接続された各プロセッサノードが相互にソケット通信を行うことが可能となっている状態を示す。

【００３４】

各プロセッサノード１００、１１０、１２０のうち、プロセッサノード１００、１１０間は、図６に示すように、例えばＥｔｈｅｒｎｅｔ（登録商標）１３０などの各種のネットワークを透過するソケット６００で接続された二つのプログラム６０１、６０２の間で双方向に通信データを送受信することができる。

【００３５】

このソケット６００で接続されることによって示されるソケット通信では、各プロセッサノード１００、１１０、１２０のうち、任意の２つのプロセッサノードで動作しているそれぞれのプログラム間で通信データを送受信することができる。また、ソケット６００を通して送受信可能なデータは、バイトストリームであり、ここで、バイトストリームとは、図６に示すように、送受信のそれぞれのデータでバイトデータが一列に並んだデータであり、送信元が送出した各バイトデータがその送出された送出順序のままで受信側のプロセッサノードに届くようになっている。

【００３６】

図７、図８は、複数のプロセッサノード１００、１１０、１２０のうち、例えばプロセッサノード１００、１１０で動作しているプログラムが通信を行っている処理を示す説明図である。プロセッサノード１００、１１０間では、図７に示すように、プロセッサノード１００、１１０でそれぞれ内部のメインメモリ１０３、１１３からプログラムを読み出して動作し、各プログラムの処理に従って相互に通信データの書き込みを要求する。

【００３７】

プロセッサノード１００からプロセッサノード１１０への書き込みの要求では、図８に示すように、プロセッサノード１００のメインメモリ１０３の一部の記憶領域がプロセッサノード１１０のメインメモリ１１３内のメモリウィンドウにマッピングされている。この時に、プロセッサノード１１０のプログラムで動作しているプロセスＢがメモリウィンドウの記憶領域に対して書き込みを行うと、この書き込まれたデータと記憶領域を示すアドレスがプロセッサノード１００に送信され、リモートメモリとして利用されているプロセッサノード１００内のメインメモリ１０３に対して書き込み処理が行われる。そのメインメモリ１０３に対してプロセッサノード１００で動作しているプロセスが読み出しを行うことにより、プロセッサノード１１０からプロセッサノード１００に対して通信データを伝達することが可能となる。

【００３８】

一方、プロセッサノード１１０からプロセッサノード１００への書き込みの要求では、プロセッサノード１１０のメインメモリ１１３の一部の記憶領域がプロセッサノード１００のメインメモリ１０３内のメモリウィンドウにマッピングされている。そして、同様にしてプロセッサノード１００からプロセッサノード１１０に対して通信データが伝達される。

【００３９】
図９は、図６に示すように、複数のプロセッサノード１００、１１０、１２０間がソケット６００で接続されている場合の通信データの送受信処理を示すシーケンス図である。この場合の送受信処理では、図９に示すように、一方のサーバ９０内のプロセッサノード１００と、もう一方のクライアント９１内のプロセッサノード１１０とが、それぞれソケット６００を介して接続されている。
【００４０】
サーバ９０は、ステップＳ９０１でクライアント９１から送信されたソケット６００使用要求（ｓｏｃｋｅｔ）のデータを受け取ると、これに応じて同様のデータと、（ｂｉｎｄ）、（ｌｉｓｔｅｎ）、（ａｃｃｅｐｔ）の各データを生成し、サーバ９０およびクライアント９１間のコネクションを確立するための処理行う。
【００４１】
サーバ９０、クライアント９１は、ステップＳ９０２でソケット６００を初期化する処理を行い、各種のデータを送受信可能な状態に設定する。
【００４２】
クライアント９１は、ステップＳ９０３でソケット６００を介して送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１１１で動作するプログラムに一度戻して、この信号に従って通信データを共有メモリ１４０に送信し、通信データを共有メモリ１４０に書き込む処理を行う。
【００４３】
クライアント９１は、ステップＳ９０４でステップＳ９０３において共有メモリ１４０に通信データを書き込んだことを通知するための書込通知情報を、ソケット６００を介してサーバ９０に送信する。
【００４４】
一方、サーバ９０は、ステップＳ９０３においてクライアント９１と同様に、ソケット６００を介して送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１０１で動作するプログラムに一度戻す。そして、サーバ９０は、ステップＳ９０４においてクライアント９１から送信された書込通知情報を受信し、ステップＳ９０５でこの書込通知情報に従って、通信データを共有メモリ１４０から読み出す処理を行う。
【００４５】
サーバ９０は、ステップＳ９０６でステップＳ９０５において共有メモリ１４０から通信データを読み出したことを通知するための読出通知情報を、ソケット６００を介してクライアント９１に送信する。
【００４６】
クライアント９１は、ステップＳ９０７でサーバ９０およびクライアント９１間のコネクションを切断することを通知するための切断通知信号を、ソケット６００を介してサーバ９０に送信する。一方、サーバ９０は、ステップＳ９０８で切断通知信号を受信すると、これに応じてコネクションを切断することに応答することを通知するための切断通知信号を、ソケット６００を介してクライアント９１に送信する。
【００４７】
サーバ９０、クライアント９１は、ステップＳ９０９でソケット６００の動作を終了する処理を行い、各種のデータを送受信不可能な状態に設定する。また、サーバ９０は、サーバ９０およびクライアント９１間のコネクションを切断するための処理を行う。
【００４８】
図１０は、図９に示すように、通信データの送受信処理のコネクション確立後の具体的な例を示すシーケンス図である。この例の送受信処理では、図１０に示すように、クライアント９１がステップＳ１００３で送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１０１で動作するプログラムに一度戻して、この信号に従って２５６ＫＢの２つの通信データを共有メモリ１４０に送信する。クライアント９１は、２つのそれぞれの通信データを共有メモリ１４０の記憶領域上にコピーして書き込む処理を２回行う。

【００４９】
クライアント９１は、ステップＳ１００４でステップＳ１００３において共有メモリ１４０に通信データを書き込んだことを通知するための書込通知情報を、サーバ９０に送信する。
【００５０】
一方、サーバ９０は、ステップＳ１００３においてクライアント９１と同様に、送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１０１で動作するプログラムに一度戻し、ステップＳ１００５で通信データの待ち状態となる。そして、サーバ９０は、ステップＳ１００４においてクライアント９１から送信された書込通知情報を受信し、ステップＳ１００６でこの書込通知情報に従って、５１２ＫＢの通信データを共有メモリ１４０の記憶領域上からコピーして読み出す処理を行う。
【００５１】
サーバ９０は、ステップＳ１００７でステップＳ１００６において共有メモリ１４０から通信データを読み出したことを通知するための読出通知情報を、クライアント９１に送信する。
【００５２】
図１１は、図９に示すように、通信データの送受信処理のコネクション確立後の他の具体的な例を示すシーケンス図である。この例の送受信処理では、図１１に示すように、クライアント９１がステップＳ１１０３で送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１１１で動作するプログラムに一度戻して、この信号に従って１．５ＭＢの通信データを共有メモリ１４０に送信する。クライアント９１は、この通信データを５１２ＫＢのそれぞれ３つの同一容量の通信データに分割し、これらのうち２つの通信データを共有メモリ１４０の記憶領域上にコピーして書き込む処理を２回行う。
【００５３】
クライアント９１は、ステップＳ１１０５でステップＳ１１０３において共有メモリ１４０に通信データを書き込んだことを通知するための書込通知情報を、それぞれ２回の書き込み処理毎にサーバ９０に送信する。ここで、例えば共有メモリ１４０の容量が１ＭＢの場合には、共有メモリ１４０の記憶領域が満たされ書き込む処理が不可能となるため、ステップＳ１１０４で通信データの待ち状態となる。
【００５４】
一方、サーバ９０は、上述のステップＳ１１０３でクライアント９１と同様に、送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１０１で動作するプログラムに一度戻す。そして、サーバ９０は、ステップＳ１１０５においてクライアント９１から送信された書込通知情報をそれぞれ受信し、ステップＳ１１０６でこの書込通知情報に従って、１つ目の通信データを共有メモリ１４０の記憶領域上からコピーして読み出す処理を行う。
【００５５】
サーバ９０は、ステップＳ１１０７でステップＳ１１０６において共有メモリ１４０から通信データを読み出したことを通知するための読出通知情報を、クライアント９１に送信する。
【００５６】
クライアント９１は、ステップＳ１１０７においてサーバ９０から送信された書込通知情報を受信し、これに応じて通信データの待ち状態を解除する。そして、クライアント９１は、ステップＳ１１０８で共有メモリ１４０の記憶領域に書き込まれた通信データを消去した上で、残りの３つ目の通信データを共有メモリ１４０の記憶領域上にコピーして書き込む処理を行う。
【００５７】
クライアント９１は、ステップＳ１１０９でステップＳ１１０８において共有メモリ１４０に通信データを書き込んだことを通知するための書込通知情報をサーバ９０に送信する。
【００５８】

図１２は、図１１に示すように、クライアント９１で実行されるコネクション確立および切断の処理を示すシーケンス図である。これらの処理では、図１２に示すように、マルチプロセッサシステム１０の起動時にまず、クライアント９１が、ステップＳ１２０１でメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４を介して、共有メモリ１４０内で通信データを書き込むための記憶領域を確保する。そして、クライアント９１は、ステップＳ１２０２、Ｓ１２０３でソケット６００使用要求（ｓｏｃｋｅｔ）のデータをＣＰＵ１１１からメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４に送信して初期化する。また、コネクションを確立し各種のデータが送受信可能な状態に設定する。

【００５９】

クライアント９１は、ステップＳ１２０４で共有メモリ１４０内で通信データを書き込むための記憶領域を割り当てるための割り当て要求をメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４に送信する。また、クライアント９１は、ステップＳ１２０５で共有メモリ１４０を書き込みや読み出しなどのアクセス待ち状態に設定する。

【００６０】

クライアント９１は、ステップＳ１２０６でメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４により共有メモリ１４０にアクセスし、通信データを書き込むための記憶領域を割り当てるための処理を行う。例えば、クライアント９１は、共有メモリ１４０内の記憶領域のうち、データが書き込まれていない空き領域、またはデータが消去済みの領域を検出して、これらの検出した各領域から通信データを書き込むための領域を確保し、この確保した領域内の各アドレスなどを割り当てる処理を行う。クライアント９１は、ステップＳ１２０６において記憶領域を割り当てた割り当て結果を、ステップＳ１２０７でメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４によりＣＰＵ１１１に送信する。

【００６１】

クライアント９１は、ステップＳ１２０８でステップＳ１２０７において送信された割り当て結果や割り当てた各アドレスを通知するための割当通知情報を、サーバ９０に送信する。そして、クライアント９１は、ステップＳ１２０９で、サーバ９０から送信された、サーバ９０で同様に共有メモリ１４０内の記憶領域で割り当てられた割り当て結果や割り当てた各アドレスを通知するための割当通知情報を、受信する。

【００６２】

クライアント９１は、ステップＳ１２１０、Ｓ１２１１でソケット６００を介して送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１１１で動作するプログラムに一度戻して、この信号に従って通信データを共有メモリ１４０に送信し、通信データを共有メモリ１４０に書き込む処理を行う。ここで、通信データは、共有メモリ１４０内の記憶領域のうち、ステップＳ１２０６において割り当てた領域に書き込まれる。

【００６３】

クライアント９１は、ステップＳ１２１２でステップＳ１２１１において共有メモリ１４０に通信データを書き込んだことを通知するための書込通知情報を、ソケット６００を介してサーバ９０に送信する。

【００６４】

一方、サーバ９０では、ステップＳ１２１０においてクライアント９１と同様に、ソケット６００を介して送信された信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１０１で動作するプログラムに一度戻す。そして、サーバ９０では、ステップＳ１２１１においてクライアント９１から送信された書込通知情報を受信し、この書込通知情報に従って、通信データを共有メモリ１４０から読み出す処理が行われる。

【００６５】

サーバ９０は、ステップＳ１２１３でステップＳ１２１１において共有メモリ１４０から通信データを読み出したことを通知するための読出通知情報を、ソケット６００を介してクライアント９１に送信する。

【００６６】

クライアント９１は、ステップＳ１２１４でサーバ９０およびクライアント９１間のコネクションを切断することを通知するための切断通知信号を、ソケット６００を介してサーバ９０に送信する。一方、サーバ９０では、ステップＳ１２１５で切断通知信号を受信すると、これに応じてコネクションを切断することに応答することを通知するための応答通知信号を、ソケット６００を介してクライアント９１に送信する。

【００６７】

クライアント９１は、ステップＳ１２１６で共有メモリ１４０内の記憶領域を解放するための解放要求をメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４に送信する。また、クライアント９１は、ステップＳ１２１７で共有メモリ１４０を書き込みや読み出しなどのアクセス待ち状態に設定する。

【００６８】

クライアント９１は、ステップＳ１２１８でメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４により共有メモリ１４０にアクセスし、ステップＳ１２０６において割り当てた記憶領域を解除し、解放するための処理を行う。クライアント９１は、ステップＳ１２１８において記憶領域を解放した解放結果を、ステップＳ１２１９でメモリコントローラ１０２、ホスト－ＰＣＩブリッジ１０４によりＣＰＵ１１１に送信する。

【００６９】

クライアント９１は、ステップＳ１２２０でソケット６００の動作を終了する処理を行い、各種のデータを送受信不可能な状態に設定する。そして、処理を終了するための信号（ｐ＿ｃｏｎｄ＿ｓｉｇｎａｌ）をＣＰＵ１０１で動作するプログラムに戻して、この信号に従ってプログラムの実行を終了する。

【００７０】

図１３は、ソケット６００の動作を指示するためのソケット指示情報を記憶したテーブル１３００を示す説明図である。このテーブル１３００では、図１３に示すように、複数種類の各ソケット指示情報と、各ソケット指示情報のそれぞれで指示されるソケット６００の動作の内容の情報とが関連付けられて記憶されている。例えば、テーブル１３００では、図１３に示すように、ソケット指示情報である「ｓｏｃｋｆｄ」と、ソケット６００の動作の内容を示す「ソケットディスクリプタ」とが関連付けらて記憶されている。また、他のソケット指示情報である「ｓｔａｔｅ」と、ソケット６００の動作の内容を示す「ソケットステート」とが関連付けられて記憶されている。

【００７１】

図１４は、ソケット指示情報に従って共有メモリ１４０に対して行った処理の例を示す説明図である。サーバ９０は、図１４に示すように、共有メモリ１４０にアクセスし、共有メモリ１４０内の記憶領域から通信データを書き込むための領域を確保し、この確保した領域内の各アドレスなどを割り当てるための処理を行う。そして、サーバ９０は、記憶領域を割り当てたことを示すソケット指示情報「ｌｏｃａｌ．ｂｕｆｆ＿ａｄｒｓ」をソケット６００を介してホスト－ＰＣＩブリッジ１０４によりＣＰＵ１０１に送信する。

【００７２】

また、サーバ９０は、図１４に示すように、共有メモリ１４０にアクセスし、共有メモリ１４０内の記憶領域から、クライアント９１により書き込まれた通信データを読み出すための領域を確保し、この確保した領域内の各アドレスなどを割り当てるための処理を行う。そして、サーバ９０は、記憶領域を割り当てたことを示すソケット指示情報「ｒｅｍｏｔｅ．ｂｕｆｆ＿ａｄｒｓ」をソケット６００を介してホスト－ＰＣＩブリッジ１０４によりＣＰＵ１０１に送信する。

【００７３】

一方、クライアント９１は、図１４に示すように、共有メモリ１４０にアクセスし、共有メモリ１４０内の記憶領域から通信データを書き込むための領域を確保し、この確保した領域内の各アドレスなどを割り当てるための処理を行う。そして、クライアント９１は、記憶領域を割り当てたことを示すソケット指示情報「ｌｏｃａｌ．ｂｕｆｆ＿ａｄｒｓ」をソケット６００を介してホスト－ＰＣＩブリッジ１１４によりＣＰＵ１１１に送信す

る。

【００７４】

また、クライアント９１は、図１４に示すように、共有メモリ１４０にアクセスし、共有メモリ１４０内の記憶領域から、サーバ９０により書き込まれた通信データを読み出すための領域を確保し、この確保した領域内の各アドレスなどを割り当てるための処理を行う。そして、クライアント９１は、記憶領域を割り当てたことを示すソケット指示情報「ｒｅｍｏｔｅ．ｂｕｆｆ＿ａｄｒｓ」をソケット６００を介してホスト－ＰＣＩブリッジ１１４によりＣＰＵ１１１に送信する。

【００７５】

図１５は、ソケット指示情報に従って共有メモリ１４０に対して行った処理の他の例を示す説明図である。図１５に示すように、例えばサーバ９０やクライアント９１などが共有メモリ１４０にアクセスし、共有メモリ１４０内の記憶領域から通信データを書き込むための複数の領域を確保し、これらの確保したそれぞれの領域内の各アドレスなどを割り当てるための処理を行う。

【００７６】

そして、図１５に示すように、１つ目（メモリウィンドウ１）の記憶領域を割り当てたことを示すソケット指示情報をソケット６００を介してＣＰＵ１０１に送信する。ソケット指示情報には、割り当てた領域の容量（２５６ＭＢ）を示す「ｌｏｃａｌ．ｂｕｆｆ＿ｓｉｚｅ」と、割り当てた領域のアドレスを示す「ｌｏｃａｌ．ｂｕｆｆ＿ａｄｒｓ」とが含まれている。

【００７７】

また、２つ目（メモリウィンドウ２）の記憶領域を割り当てたことを示すソケット指示情報をソケット６００を介してＣＰＵ１０１に送信する。ソケット指示情報には、割り当てた領域の容量（５１２ＭＢ）を示す「ｌｏｃａｌ．ｂｕｆｆ＿ｓｉｚｅ」と、割り当てた領域のアドレスを示す「ｌｏｃａｌ．ｂｕｆｆ＿ａｄｒｓ」とが含まれている。

【００７８】

また、３つ目（メモリウィンドウ３）の記憶領域を割り当てたことを示すソケット指示情報をソケット６００を介してＣＰＵ１０１に送信する。ソケット指示情報には、割り当てた領域の容量（１２８ＭＢ）を示す「ｌｏｃａｌ．ｂｕｆｆ＿ｓｉｚｅ」と、割り当てた領域のアドレスを示す「ｌｏｃａｌ．ｂｕｆｆ＿ａｄｒｓ」とが含まれている。

【００７９】

図１６は、共有メモリ１４０内の記憶領域で割り当てられたメモリウィンドウのデータ構成を示す説明図である。このメモリウィンドウには、図１６（ａ）に示すように、マジックナンバーと、バッファサイズと、ライトポインタと、リードポインタと、リングバッファの各データが含まれている。マジックナンバーは、メモリウィンドウの先頭位置を識別するための識別情報である。

【００８０】

バッファサイズは、メモリウィンドウの共有メモリ１４０内での全体の容量を示す例えばバイト数などのデータである。ライトポインタは、リングバッファ内での通信データを書き込む際の開始位置を示すポインタであり、リングバッファ内でのオフセットアドレスが含まれている。リードポインタは、リングバッファ内での通信データを読み出す際の開始位置を示すポインタであり、リングバッファ内でのオフセットアドレスが含まれている。

【００８１】

メモリウィンドウに含まれるリングバッファは、記憶領域内の開始位置が空きとなっている場合には、通信データの内容をリングバッファの領域内に全て書き込むと、これに応じて、リングバッファ内の先頭位置に戻って書き込み処理を続けて行うことが可能であり、循環的に書き込み処理が可能なバッファである。リングバッファには、図１６（ｂ）に示すように、ソケット６００を介して送受信する際の複数にそれぞれ分けられたデータ群であるパケットが含まれている。

【００８２】
それぞれの各パケットには、図１６（ｃ）に示すように、パケットデータサイズと、パケットデータが含まれている。パケットデータサイズは、パケットデータ全体の容量を示す例えばバイト数などのデータであり、４ｂｙｔｅから成る。パケットデータには、通信データの内容が複数の各パケット毎に分けて含まれている。
【００８３】
図１７は、共有メモリ１４０内のリングバッファに通信データを用いて行った処理を示す説明図である。図１７に示すように、まず、（１）で例えばサーバ９０やクライアント９１などが共有メモリ１４０にアクセスし、共有メモリ１４０のリングバッファを初期状態に設定する。即ち、ライトポインタおよびリードポインタの位置をリングバッファ内で先頭位置に設定する処理を行う。
【００８４】
また、（２）で、通信データの内容を含む１つ目のパケットをリングバッファに書き込み、その後、これに伴ってライトポインタの位置をこのパケットの終端位置に移動させる。ここで、通信データの内容を書き込む処理は、４ｂｙｔｅ単位毎に行う。５ｂｙｔｅなどのデータを書き込むときには２つの４ｂｙｔｅ単位毎に８ｂｙｔｅ分まで書き込む処理を行う。（３）で、続けて通信データの内容を含む２つ目のパケットを書き込み、その後、これに伴ってライトポインタの位置をこのパケットの終端位置に移動させる。
【００８５】
次に、（４）で、他のサーバ９０やクライアント９１などが共有メモリ１４０にアクセスし、共有メモリ１４０のリングバッファから、上述の（１）で書き込まれた１つ目のパケットを読み出す処理を行う。このとき、この読み出したパケットを消去し、その後、これに伴ってリードポインタの位置を２つ目のパケットの開始位置に移動させる。
【００８６】
また、（５）で、サーバ９０やクライアント９１などが共有メモリ１４０にアクセスし、通信データの内容を含む３つ目のパケットをリングバッファに書き込む処理を行う。このとき、通信データの内容を書き込んでいき、リングバッファの領域内に全て書き込んだ場合には、リングバッファ内の先頭位置に戻って上述の（４）で消去した領域に続けて書き込む処理を行う。その後、これに伴ってライトポインタの位置も同様に、リングバッファ内の先頭位置に戻ってこのパケットの終端位置に移動させる。
【００８７】
（６）で、通信データの内容を含む４つ目のパケットを書き込み、その後、これに伴ってライトポインタの位置をこのパケットの終端位置に移動させる。ここで、通信データの内容を書き込んでいき、リングバッファの領域内に空きが残り４ｂｙｔｅとなった場合には、この時点でリングバッファ内がフル状態として、書き込み不可に設定する。
【００８８】
図１８は、複数のプロセッサノード１００、１１０、１２０のうち、例えばプロセッサノード１００、１１０が共有メモリ１４０に通信データを用いて行った処理を示す説明図である。図１８に示すように、まず、プロセッサノード１００のＣＰＵ１０１は、ステップＳ１８０１で共有メモリ１４０にアクセスして、リングバッファ内で先頭位置（０ｘＥ８００　１００８）を指定する。そして、ＣＰＵ１０１は、１つ目の通信データをリングバッファに書き込み、その後、これに伴ってライトポインタの位置をこの通信データの終端に含まれるパケットの終端位置（０ｘ０００００　ＦＦＦ０）に移動させる。
【００８９】
一方、プロセッサノード１１０のＣＰＵ１１１は、ステップＳ１８０２で共有メモリ１４０にアクセスし、共有メモリ１４０のリングバッファから、ステップＳ１８０１において書き込まれた通信データをパケット毎に順次読み出す処理を行う。このとき、この読み出したパケットを消去し、その後、これに伴ってリードポインタの位置を、この読み出したパケットの次のパケットの開始位置に移動させる。
【００９０】

ＣＰＵ１１１は、ステップＳ１８０３、Ｓ１８０４で共有メモリ１４０に続けてアクセスし、共有メモリ１４０のリングバッファから、ステップＳ１８０１において書き込まれた通信データをパケット毎に順次読み出す処理を繰り返し行っていく。このとき、これらの読み出した各パケットをそれぞれ消去し、その後、これに伴ってリードポインタの位置を、これらの読み出したパケットの次のパケットの開始位置に移動させる。

【００９１】

次に、ＣＰＵ１０１は、ステップＳ１８０５で共有メモリ１４０にアクセスして、リングバッファ内で先頭位置（０ｘＥ８００　１００８）を指定する。そして、ＣＰＵ１０１は、２つ目の通信データをリングバッファに書き込み、その後、これに伴ってライトポインタの位置をこの通信データの終端に含まれるパケットの終端位置（０ｘ０００１　００１０）に移動させる。

【００９２】

ＣＰＵ１１１は、ステップＳ１８０６、Ｓ１８０７で共有メモリ１４０に続けてアクセスし、共有メモリ１４０のリングバッファから、ステップＳ１８０６、Ｓ１８０７において書き込まれた通信データをパケット毎に順次読み出す処理を行う。このとき、これらの読み出した各パケットをそれぞれ消去し、その後、これに伴ってリードポインタの位置を、これらの読み出したパケットの次のパケットの開始位置に移動させる。

【００９３】

図１９は、複数のプロセッサノード１００、１１０、１２０のうち、例えばプロセッサノード１００、１１０が共有メモリ１４０に通信データを用いて行った他の処理を示す説明図である。図１９に示すように、まず、プロセッサノード１００のＣＰＵ１０１は、ステップＳ１９０１、Ｓ１９０２で共有メモリ１４０にアクセスして、共有メモリ１４０内の２つのリングバッファ内でそれぞれ先頭位置（０ｘＥ８００　１００８）、（０ｘＥ８００　１００Ａ）を指定する。そして、ＣＰＵ１０１は、２つの通信データをこれらの各リングバッファにそれぞれ書き込み、その後、これに伴って各ライトポインタの位置をこれらの各通信データの終端に含まれるパケットの終端位置（０ｘ００００　ＦＦＦ０）、（０ｘ００００　００００）に移動させる。

【００９４】

一方、プロセッサノード１１０のＣＰＵ１１１は、ステップＳ１９０３で共有メモリ１４０にアクセスし、共有メモリ１４０のリングバッファから、ステップＳ１９０１、Ｓ１９０２において書き込まれた通信データをパケット毎に順次読み出す処理を行う。このとき、これらの読み出した各パケットをそれぞれ消去し、その後、これに伴ってリードポインタの位置を、これらの読み出したパケットの次のパケットの開始位置に移動させる。

【００９５】

次に、ＣＰＵ１０１は、ステップＳ１９０４、Ｓ１９０６で共有メモリ１４０にアクセスして、共有メモリ１４０内の２つのリングバッファ内でそれぞれ先頭位置（０ｘＥ８００　１００８）、（０ｘＥ８００　１００Ａ）を指定する。そして、ＣＰＵ１０１は、２つの通信データをこれらの各リングバッファにそれぞれ書き込み、その後、これに伴って各ライトポインタの位置をこれらの各通信データの終端に含まれるパケットの終端位置（０ｘ００００　００１０）、（０ｘ００００　０００１）に移動させる。

【００９６】

ここで、上述のステップＳ１９０４およびＳ１９０６での処理の間、ステップＳ１９０５でＣＰＵ１１１は、共有メモリ１４０にアクセスし、共有メモリ１４０のリングバッファから、ステップＳ１９０４、Ｓ１９０６において書き込まれた通信データをパケット毎に順次読み出す処理を行う。このとき、これらの読み出した各パケットをそれぞれ消去し、その後、これに伴ってリードポインタの位置を、これらの読み出したパケットの次のパケットの開始位置に移動させる。

【００９７】

また、ＣＰＵ１１１は、ステップＳ１９０７で共有メモリ１４０に続けてアクセスし、共有メモリ１４０のリングバッファから、ステップＳ１９０４、Ｓ１９０６において書き

込まれた通信データをパケット毎に順次読み出す処理を繰り返し行っていく。このとき、これらの読み出した各パケットをそれぞれ消去し、その後、これに伴ってリードポインタの位置を、これらの読み出したパケットの次のパケットの開始位置に移動させる。

【００９８】

以上のように、第１の実施形態におけるマルチプロセッサシステム１０では、各プロセッサノード１００、１１０、１２０がネットワークを透過するソケット６００で互いに接続されて、双方向に通信データが送受信可能となっている。共有メモリ１４０に対して通信データを用いて書き込みまたは読み出しの各処理を行う際には、クライアント９１のＣＰＵ１１１が信号に従って通信データを共有メモリ１４０に送信し、通信データを共有メモリ１４０に書き込む処理を行う。そして、ＣＰＵ１１１は、ソケット６００を介してサーバ９０に書込通知情報を送信し、共有メモリ１４０に通信データを書き込んだことを通知する。

【００９９】

一方、サーバ９０は、この書込通知情報に従って通信データを共有メモリ１４０から読み出す処理を行う。また、サーバ９０は、ソケット６００を介してクライアント９１に読出通知情報を送信し、共有メモリ１４０から通信データを読み出したことを通知する。

【０１００】

このため、各プロセッサノード１００、１１０、１２０の間をソケット６００で接続して、各プロセッサノード１００、１１０、１２０で動作しているプログラム上のプロセスの間で、同一の共有メモリ１４０を共有利用することによって、この共有メモリ１４０が分散共有メモリとして機能する。この共有メモリ１４０は、複数のプロセッサノード間で利用できるというソケットの特徴と、広帯域であるという共有メモリ１４０の特徴とをそれぞれ有し、通信処理で発生する通信オーバーヘッドを低減し、高速通信を可能とすることができる。

【０１０１】

〔第２の実施形態〕

以下、本発明に係る第２の実施形態をもって説明する。図２は、本発明におけるプロセッサ間通信装置の１つの例であるマルチプロセッサシステム１０の第２の実施形態での全体構成を示す説明図である。このマルチプロセッサシステム１０は、複数のプロセッサノード２００、２１０、２２０と、これらの各プロセッサノード２００、２１０、２２０間でデータなどの送受信を行うためのＥｔｈｅｒｎｅｔ（登録商標）２３０と、各プロセッサノード２００、２１０、２２０のそれぞれをデータなどの送受信が可能に接続するＰＣＩバス２４０とから構成されている。

【０１０２】

プロセッサノード２００は、ＣＰＵ２０１と、メモリコントローラ２０２と、メインメモリ２０３と、ホスト－ＰＣＩブリッジ２０４と、Ｅｔｈｅｒｎｅｔ（登録商標）カード２０５とを備えている。これらの各構成要素は、第１の実施形態におけるプロセッサノード１００が備えているＣＰＵ１０１、メモリコントローラ１０２、メインメモリ１０３、ホスト－ＰＣＩブリッジ１０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５と同様の機能を有しており説明を省略する。

【０１０３】

また、プロセッサノード２００は、上述の各構成要素に加えて、ＰＣＩバス２４０に接続され、このＰＣＩバス２４０を介して外部との間でデータの送受信を行うＰＣＩ－ＰＣＩブリッジ２０６を備えている。ＰＣＩ－ＰＣＩブリッジ２０６は、例えばプロセッサノード２００内で、ホスト－ＰＣＩブリッジ２０４およびＰＣＩバス２４０間に直接接続され、ＣＰＵ２０１に対してＰＣＩバス２４０を介して外部との間でデータの送受信を行う。

【０１０４】

プロセッサノード２１０、２２０は、プロセッサノード２００と同様の構成となっており、ＣＰＵ２０１、メモリコントローラ２０２、メインメモリ２０３、ホスト－ＰＣＩブ

リッジ２０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード２０５と同様の機能を有するＣＰＵ２１１、２２１、メモリコントローラ２１２、２２２、メインメモリ２１３、２２３、ホスト－ＰＣＩブリッジ２１４、２２４、Ｅｔｈｅｒｎｅｔ（登録商標）カード２１５、２２５を備えている。

【０１０５】

そして、図示していないが、第２の実施形態においてもマルチプロセッサシステム１０では、各プロセッサノード２００、２１０、２２０がネットワークを透過するソケット６００で互いに接続されて、双方向に通信データが送受信可能となっている。

【０１０６】

図３は、メインメモリ２０３の記憶領域の使用状態を示す説明図である。この第２の実施形態では、図３に示すように、メインメモリ２０３、２１３、２２３内の記憶領域の一部を用いて共有メモリとして利用する。このような構成とすることによって、上述の第１の実施形態において単一の共有メモリ１４０を各プロセッサノード１００、１１０、１２０と別個に設ける構成と比較して、コストの低減を図ることが可能となる。

【０１０７】

〔第３の実施形態〕

以下、本発明に係る第３の実施形態をもって説明する。図４は、本発明におけるプロセッサ間通信装置の１つの例であるマルチプロセッサシステム１０の第３の実施形態での全体構成を示す説明図である。このマルチプロセッサシステム１０は、複数のプロセッサノード３００、３１０、３２０と、これらの各プロセッサノード３００、３１０、３２０間でデータなどの送受信を行うためのＥｔｈｅｒｎｅｔ（登録商標）３３０と、各プロセッサノード３００、３１０、３２０のそれぞれをデータなどの送受信が可能に接続するＰＣＩ－Ｅｘｐｒｅｓｓスイッチ３４０とから構成されている。

【０１０８】

プロセッサノード３００は、ＣＰＵ３０１と、メモリコントローラ３０２と、メインメモリ３０３と、ホスト－ＰＣＩブリッジ３０４と、Ｅｔｈｅｒｎｅｔ（登録商標）カード３０５とを備えている。これらの各構成要素は、第１の実施形態におけるプロセッサノード１００が備えているＣＰＵ１０１、メモリコントローラ１０２、メインメモリ１０３、ホスト－ＰＣＩブリッジ１０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード１０５と同様の機能を有しており説明を省略する。

【０１０９】

プロセッサノード３１０、３２０は、プロセッサノード３００と同様の構成となっており、ＣＰＵ３０１、メモリコントローラ３０２、メインメモリ３０３、ホスト－ＰＣＩブリッジ３０４、Ｅｔｈｅｒｎｅｔ（登録商標）カード３０５と同様の機能を有するＣＰＵ３１１、３２１、メモリコントローラ３１２、３２２、メインメモリ３１３、３２３、ホスト－ＰＣＩブリッジ３１４、３２４、Ｅｔｈｅｒｎｅｔ（登録商標）カード３１５、３２５を備えている。

【０１１０】

そして、図示していないが、第３の実施形態においてもマルチプロセッサシステム１０では、各プロセッサノード３００、３１０、３２０がネットワークを透過するソケット６００で互いに接続されて、双方向に通信データが送受信可能となっている。

【０１１１】

この第３の実施形態では、メインメモリ３０３、３１３、３２３内の記憶領域の一部を用いて共有メモリとして利用する。このような構成とすることによって、上述の第１の実施形態において単一の共有メモリ１４０を各プロセッサノード１００、１１０、１２０と別個に設ける構成と比較して、コストの低減を図ることが可能となる。

【０１１２】

〔他の実施形態〕

上述の実施形態において、ソケット６００を介して書込通知情報を送信することによって、共有メモリ１４０に通信データを書き込んだことを通知しているが、共有メモリ１４

０に書き込まれた通信データのデータ量が所定量に達した場合に、通知するようにしても良い。

【０１１３】

この場合には、プロセッサノード１００、１１０、１２０を始めとする各プロセッサノードが予め共有メモリ１４０に書き込まれたデータ量を検出する検出部を備え、共有メモリ１４０への通信データの書き込み処理が開始されると、これに応じて書き込まれたデータ量を検出する。そして、このデータ量が予め設定された所定量に達すると、通信データが書き込まれたこと、および書き込まれたデータ量を通知するデータ量通知情報を生成し、ソケット６００を介して送信する。

【０１１４】

また、上述の実施形態において、プロセッサノード１００、１１０、１２０を始めとする各プロセッサノードが、共有メモリ１４０に書き込まれた通信データのデータ量や、メモリウィンドウに含まれるリングバッファの空きの領域が例えば４ｂｙｔｅの所定量まで減少した場合など、通知情報で通知する条件をユーザの要望に応じて、任意に設定することが可能であっても良い。

【０１１５】

また、本発明の上記機能は、Ｃ、Ｃ＋＋、Ｊａｖａ（登録商標）、Ｊａｖａ（登録商標）Ａｐｐｌｅｔ、Ｊａｖａ（登録商標）Ｓｃｒｉｐｔ、Ｐｅｒｌ、Ｒｕｂｙなどのレガシープログラミング言語、オブジェクト指向プログラミング言語などで記述された装置実行可能なプログラムにより実現でき、装置可読な記録媒体に格納して頒布することができる。

【０１１６】

これまで本発明を図１～図２０に示した第１～第３の実施形態をもって説明してきたが、本発明はこれに限定されるものではない。他の実施の形態、追加、変更、削除など、本発明の要旨を変更しない範囲内で変更することができ、いずれの態様においても本発明の作用・効果を奏する限り、本発明の範囲に含まれるものである。

【符号の説明】

【０１１７】

１０…マルチプロセッサシステム、１００、１１０、１２０、２００、２１０、２２０、３００、３１０、３２０…プロセッサノード、１３０、２３０、３３０…Ｅｔｈｅｒｎｅｔ（登録商標）、１４０…共有メモリ、２４０…ＰＣＩバス、３４０…ＰＣＩ－Ｅｘｐｒｅｓｓスイッチ

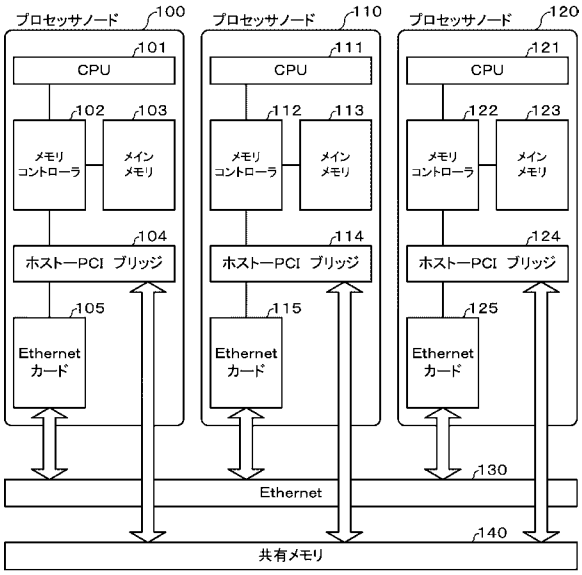
【先行技術文献】

【特許文献】

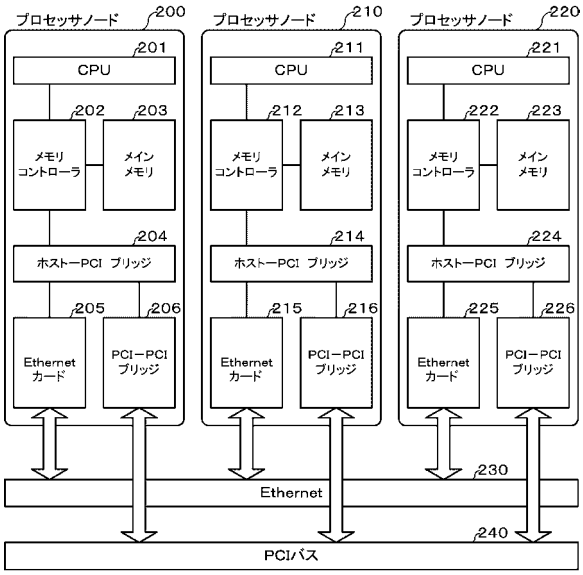
【０１１８】

【特許文献１】特許第３７４３３８１号公報

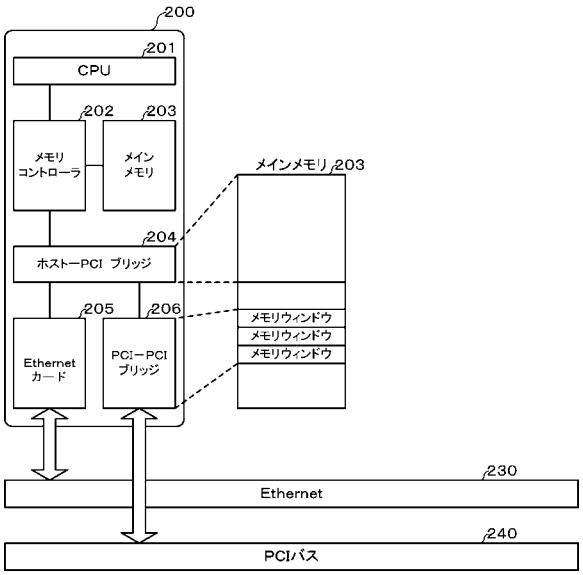
【図１】


10
プロセッサノード
100
101
CPU
102
103
メモリコントローラ
メインメモリ
104
ホストーPCI ブリッジ
105
Ethernetカード
プロセッサノード
110
111
CPU
112
113
メモリコントローラ
メインメモリ
114
ホストーPCI ブリッジ
115
Ethernetカード
プロセッサノード
120
121
CPU
122
123
メモリコントローラ
メインメモリ
124
ホストーPCI ブリッジ
125
Ethernetカード
130
Ethernet
140
共有メモリ

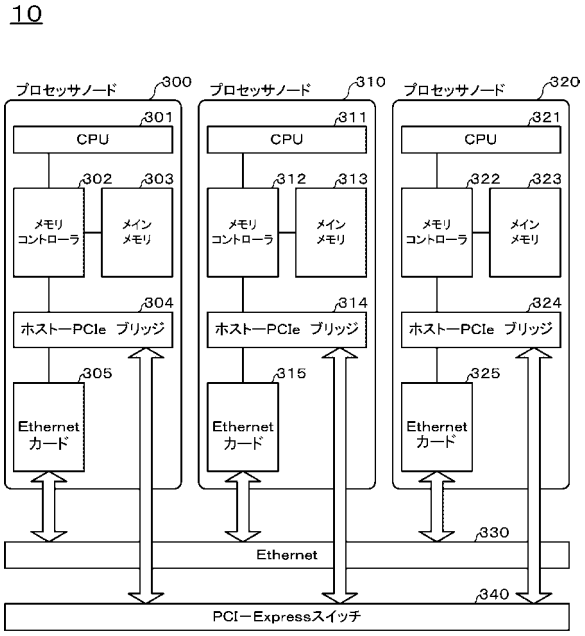
【図２】


10
プロセッサノード
200
201
CPU
202
203
メモリコントローラ
メインメモリ
204
ホストーPCI ブリッジ
205
206
Ethernetカード
PCIーPCIブリッジ
プロセッサノード
210
211
CPU
212
213
メモリコントローラ
メインメモリ
214
ホストーPCI ブリッジ
215
216
Ethernetカード
PCIーPCIブリッジ
プロセッサノード
220
221
CPU
222
223
メモリコントローラ
メインメモリ
224
ホストーPCI ブリッジ
225
226
Ethernetカード
PCIーPCIブリッジ
230
Ethernet
240
PCIバス

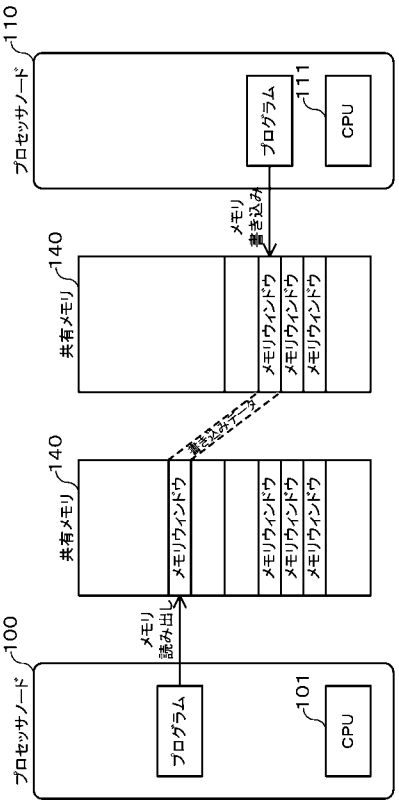
【図３】


200
201
CPU
202
203
メモリコントローラ
メインメモリ
メインメモリ 203
204
ホストーPCI ブリッジ
205
206
Ethernetカード
PCIーPCIブリッジ
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
230
Ethernet
240
PCIバス

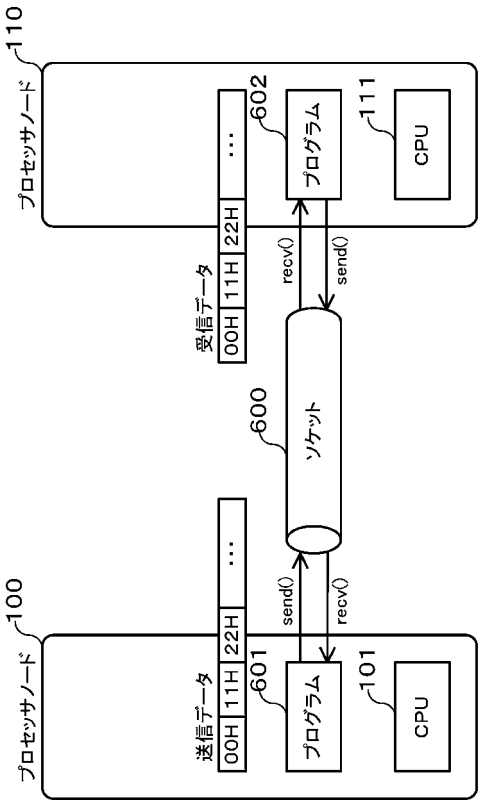
【図４】


10
プロセッサノード
300
301
CPU
302
303
メモリコントローラ
メインメモリ
304
ホストーPCIe ブリッジ
305
Ethernetカード
プロセッサノード
310
311
CPU
312
313
メモリコントローラ
メインメモリ
314
ホストーPCIe ブリッジ
315
Ethernetカード
プロセッサノード
320
321
CPU
322
323
メモリコントローラ
メインメモリ
324
ホストーPCIe ブリッジ
325
Ethernetカード
330
Ethernet
340
PCIーExpressスイッチ

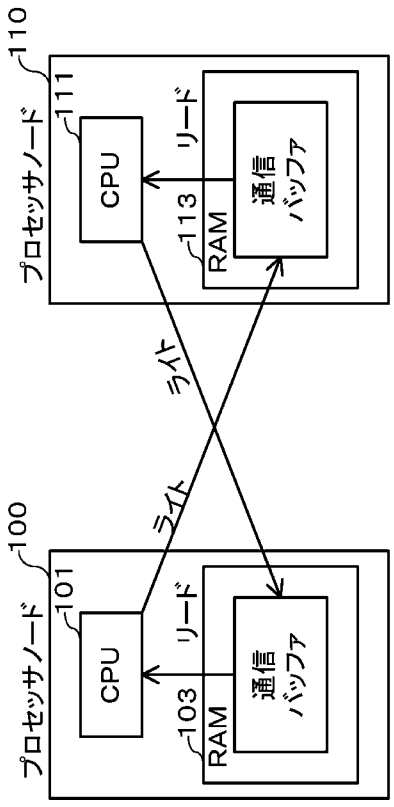
【図５】


プロセッサノード
110
プログラム
111
CPU
メモリ
書き込み
共有メモリ
140
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
書き込みデータ
共有メモリ
140
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
メモリ
読み出し
プロセッサノード
100
プログラム
101
CPU

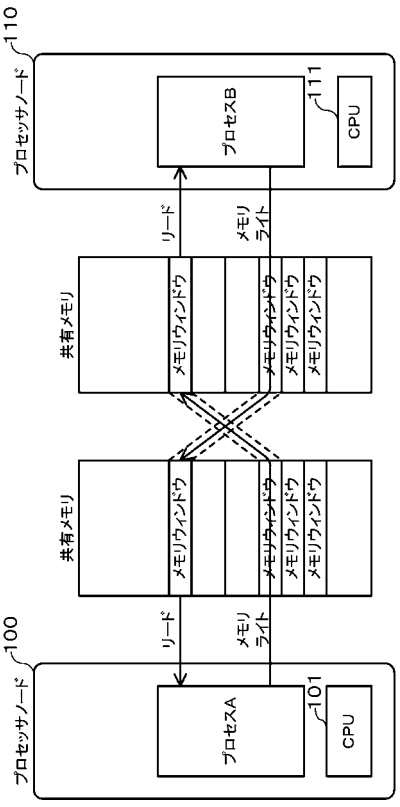
【図６】


プロセッサノード
110
602
プログラム
111
CPU
…
22H
11H
00H
受信データ
recv()
send()
600
ソケット
send()
recv()
…
22H
11H
00H
送信データ
601
プログラム
101
CPU
プロセッサノード
100

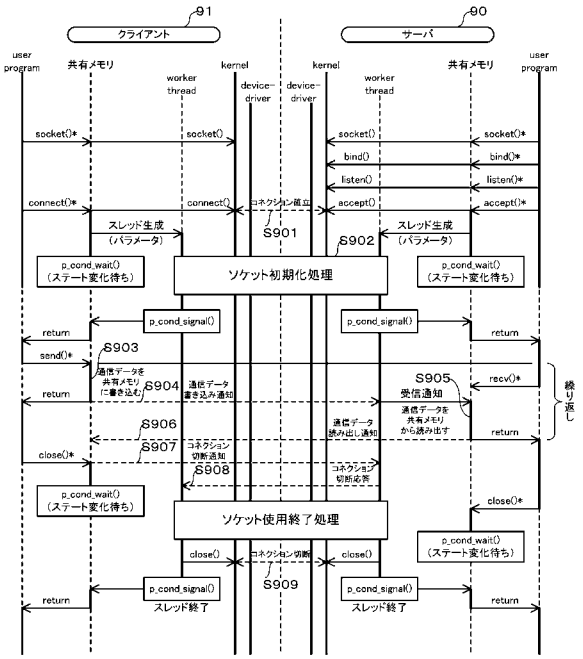
【図７】


プロセッサノード
110
111
CPU
リード
113
RAM
通信
バッファ
ライト
ライト
プロセッサノード
100
101
CPU
リード
103
RAM
通信
バッファ

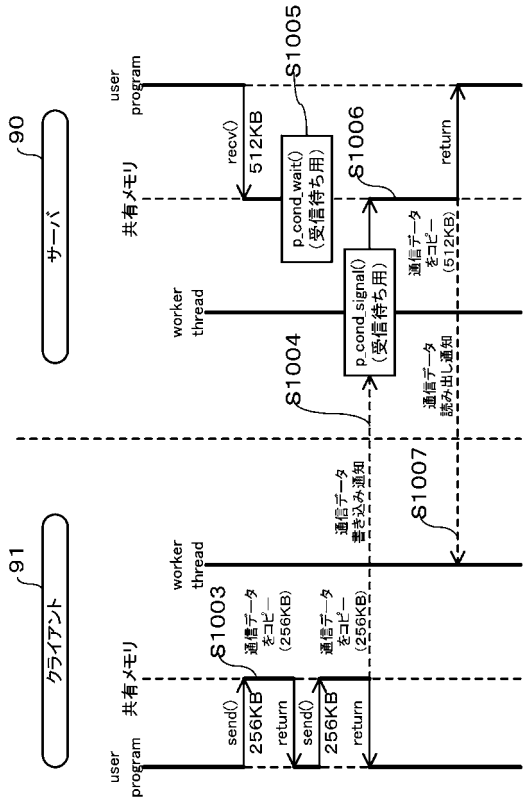
【図８】


プロセッサノード
110
プロセスB
111
CPU
リード
メモリ
ライト
共有メモリ
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
共有メモリ
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
メモリウィンドウ
リード
メモリ
ライト
プロセッサノード
100
プロセスA
101
CPU

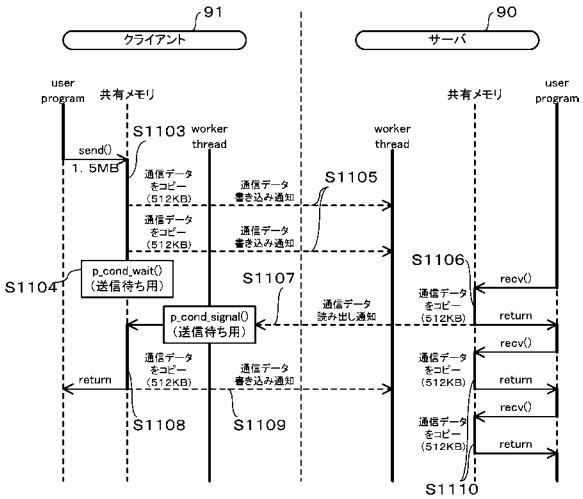
【図９】



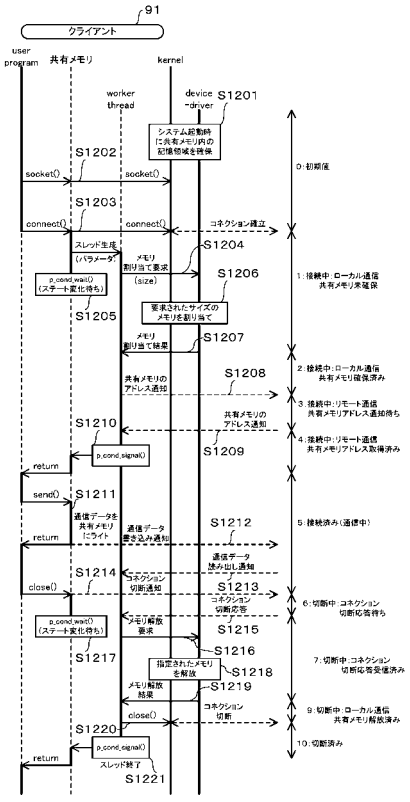
【図１０】



【図１１】



【図１２】

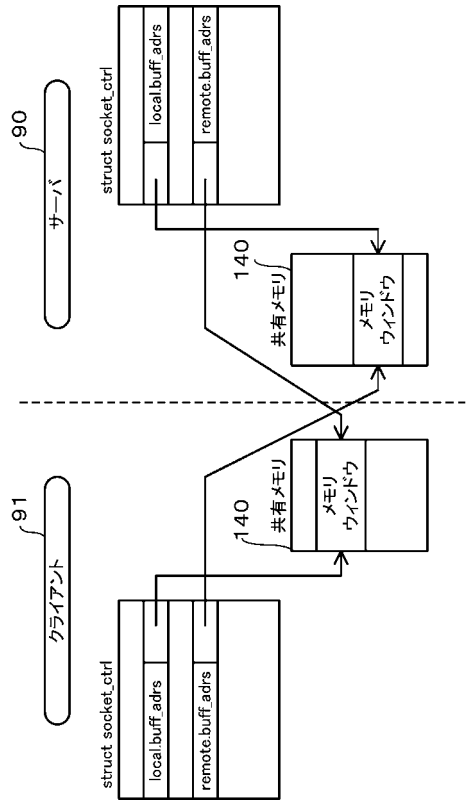


【図１３】

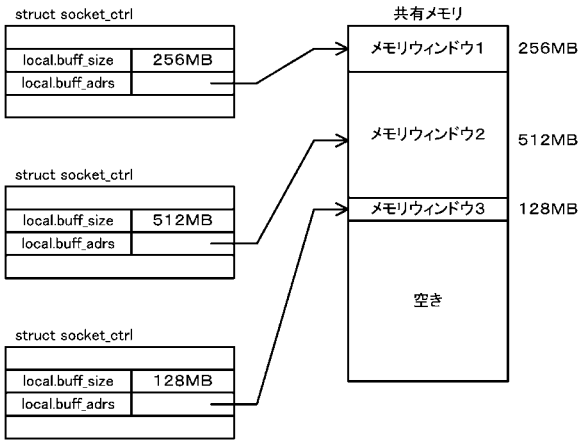
1300

変数名	内容	値
sockfd	ソケットディスクリプタ	
state	ソケットステート	
tx_write_notify_mode	通信データ書き込み通知モード	
rx_packet_count	通信パケットカウンタ	
tx_write_byte	通信データの書き込み済みバイト数	
rx_read_byte	通信データの読み込み済みバイト数	
local.buff_size	ローカル通信メモリ：メモリサイズ	
local.buff_adrs	ローカル通信メモリ：メモリアドレス	
local.ring_buff_size	ローカル通信バッファ：リングバッファサイズ	
local.ring_buff_adrs	ローカル通信バッファ：リングバッファアドレス	
remote.buff_size	リモート通信メモリ：メモリサイズ	
remote.buff_adrs	リモート通信メモリ：メモリアドレス	
remote.ring.buff_size	リモート通信バッファ：リングバッファサイズ	
remote.ring.buff_adrs	リモート通信バッファ：リングバッファアドレス	

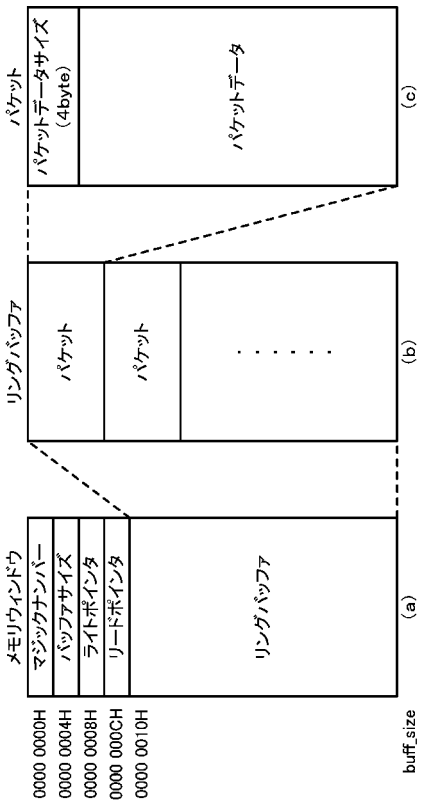
【図１４】



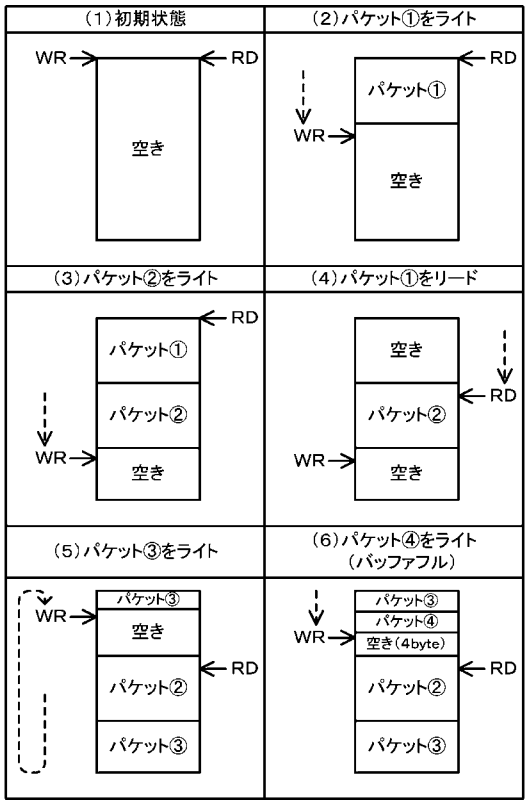
【図１５】



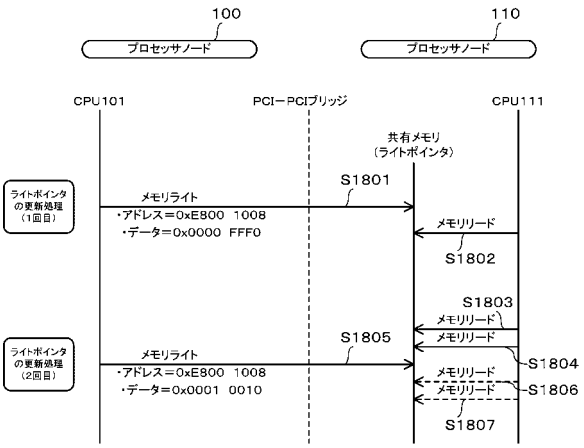
【図１６】



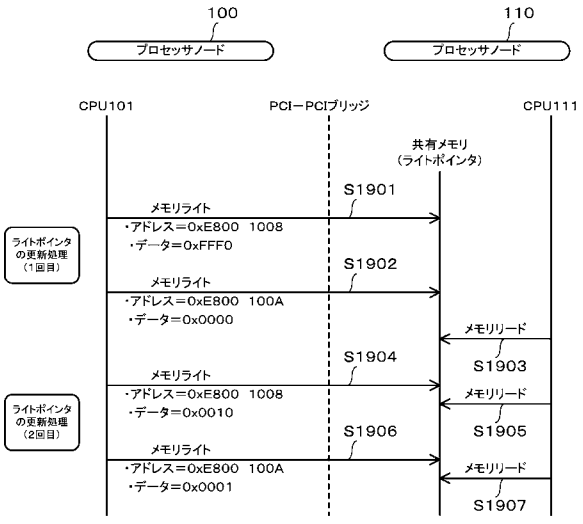
【図１７】



【図１８】



【図１９】



【図２０】