(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3705836号
(P3705836)
(45)発行日　平成17年10月12日(2005.10.12)　(24)登録日　平成17年8月5日(2005.8.5)

(51)Int.Cl.[7]	F I		
G06F 12/08	G06F	12/08	531B
G06F 9/52	G06F	12/08	507Z
G06F 12/00	G06F	12/08	515N
G06F 15/167	G06F	12/08	519E
	G06F	12/08	541Z

請求項の数 8　(全 33 頁)　最終頁に続く

(21)出願番号　特願平6-300326
(22)出願日　平成6年11月9日(1994.11.9)
(65)公開番号　特開平7-191946
(43)公開日　平成7年7月28日(1995.7.28)
審査請求日　平成13年10月15日(2001.10.15)
(31)優先権主張番号　149,418
(32)優先日　平成5年11月9日(1993.11.9)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ラジイヴ・グプタ
アメリカ合衆国カリフォルニア州ロスアルトス　エコー・ドライヴ　1052
(72)発明者　アラン・エイチ・カープ
アメリカ合衆国カリフォルニア州サニーヴェイル　アイリス・アヴェニュー　910

最終頁に続く

(54)【発明の名称】コンピュータ・システムのメモリ共用方法

(57)【特許請求の範囲】

【請求項１】

複数のデータ・ブロックを備える共用データを大域メモリに記憶するステップと、

１つのデータ・ブロック中のデータにアクセスする各プロセッサごとに、

前記データ・ブロックのローカル・コピーをプロセッサのローカル・メモリに記憶するステップと、

それぞれ、最初は非書込み値にセットされる、ローカル・メモリ中の複数のローカル・コピー書込みフラグを、ローカル・コピー中の複数の第１のサイズのデータ要素と、１対１の関係で関連付けるステップと、

プロセッサによって修正されたローカル・コピーの第１のサイズのデータ要素ごとに、

前記データ要素に関連するローカル・コピー書込みフラグを書込み値にセットするステップと、

ローカル・コピーがローカル・メモリからフラッシュされたとき、ローカル・コピーとローカル・コピー書込みフラグを大域メモリに送るステップと、

書込み値にセットされた各ローカル・コピー書込みフラグごとに、そのようなローカル・コピー書込みフラグに関連する第１のサイズのデータ要素をデータ・ブロックに記憶することにより、ローカル・コピーとデータ・ブロックを大域メモリ中で併合するステップと、

複数のプロセッサが同時にデータ・ブロックにアクセスしたときに、

少なくとも２個のプロセッサが、共に書込み値にセットされ、かつプロセッサ自体のそ

れぞれのローカル・コピーの対応する位置にある第１のサイズのデータ要素に関連するローカル・コピー書込みフラグを送る書込み／書込みデータ・レースの発生を識別するステップと、
を含む複数のプロセッサ間のメモリ共用方法。
【請求項２】
ローカル・コピーとデータ・ブロックを大域メモリ中で併合するステップが、
それぞれ、最初は非書込み値にセットされる、大域メモリ中の複数のデータ・ブロック書込みフラグを、データ・ブロック中の複数の第１のサイズのデータ要素と関連付けるステップと、
最初は第１の値にセットされる特殊併合フラグを、大域メモリ中のデータ・ブロックと関連付けるステップと、
データ・ブロックが部分的に併合される間に記憶される各ローカル・コピーごとに、
前記大域メモリにおいて、部分的に併合されたデータ・ブロックの複製コピーをローカル・コピー用のテーブル項目に記憶するステップと、
特殊併合フラグを第２の値にセットするステップと、
特殊併合フラグが第１の値にセットされている場合、関連するデータ・ブロック書込みフラグが非書込み値にセットされている、データ・ブロックの対応する位置にある第１のサイズのデータ要素と異なるローカル・コピーのデータ要素を、データ・ブロックに記憶するステップと、
特殊併合フラグが第２の値にセットされており、ローカル・コピーは記憶されたが、データ・ブロックは部分的に併合されなかった場合、関連するデータ・ブロック書込みフラグが非書込み値にセットされている、データ・ブロックの対応する位置にある第１のサイズのデータ要素と異なるローカル・コピーのデータ要素を、データ・ブロックに記憶するステップと、
特殊併合フラグが第２の値にセットされており、ローカル・コピーが記憶され、同時にデータ・ブロックが部分的に併合された場合、ローカル・コピー用のテーブル項目中の複製コピーの対応する位置にある要素と異なるローカル・コピーのデータ要素をデータ・ブロックに記憶するステップとを含む請求項１に記載のメモリ共用方法。
【請求項３】
さらに、１つのデータ・ブロック中のデータにアクセスする各プロセッサごとに、
最小サイズの修正されたデータ要素を示し、最初は、第１のサイズのデータ要素のサイズである第１のサイズを示すようにセットされる、サブワード記憶フラグを、プロセッサのローカル・メモリで、ローカル・コピーと関連付けるステップと、
プロセッサがサブワード記憶動作をローカル・コピーに対して実行するとき、サブワード記憶フラグを第２のサイズを示すようにセットするステップと、
ローカル・コピーがローカル・メモリからフラッシュされたとき、ローカル・コピーおよびローカル・コピー書込みビットと共にサブワード記憶フラグを大域メモリに送るステップと、
サブワード記憶フラグが第２のサイズを示すようにセットされているとき、
最初は非書込み値に設定される大域メモリ中の複数のデータ・ブロック書込みフラグを、データ・ブロック中の複数の第２のサイズのデータ要素と１対１の関係で関連付けるステップと、
関連するフラグが非書込み値にセットされているデータ・ブロックの第２のサイズの各データ要素ごとに、ローカル・コピーの対応する位置にあるデータ要素とそのような第２のサイズのデータ要素が異なる場合、そのような第２のサイズのデータ要素の代わりにローカル・コピー・データ要素を記憶し、そのような第２のサイズのデータ要素に関連するデータ・ブロック書込みフラグを書込み値にセットするステップと、
サブワード記憶フラグが第１のサイズを示すようにセットされているとき、
書込み値にセットされた各ローカル・コピー書込みフラグごとに、そのようなローカル・コピー書込みフラグに関連する第１のサイズのデータ要素をデータ・ブロックに記憶す

るステップとを含む請求項１に記載のメモリ共用方法。
【請求項４】
さらに、１つのデータ・ブロック中のデータにアクセスする各プロセッサごとに、
最初は非読取り値にセットされるローカル・メモリ中の複数の読取りフラグを、ローカル・コピーの第１のサイズのデータ要素と、１対１の関係で関連付けるステップと、
第１のサイズのデータ要素がプロセッサによって読み取られたとき、前記データ要素に関連する読取りフラグを読取り値にセットするステップと、
ローカル・コピーがローカル・メモリからフラッシュされたとき、ローカル・コピーおよびローカル・コピー書込みフラグと共に読取りフラグを大域メモリに送るステップと、
複数のプロセッサが同時に同じデータ・ブロックにアクセスしたとき、
１個のプロセッサが書込み値にセットされたローカル・コピー書込みフラグを送り、他のプロセッサが読取り値にセットされた読取りフラグを送る読取り／書込みデータ・レースの発生を識別するステップとを含み、前記書込みフラグおよび読取りフラグが、前記プロセッサのそれぞれのローカル・コピーの対応する位置にある第１のサイズのデータ要素に関連付けられる請求項１に記載のメモリ共用方法。
【請求項５】
さらに、最初はロック解除値にセットされるロック・フラグを大域メモリ中の各データ・ブロックと関連付けるステップと、
第１のプロセッサの要求に応じてロック・フラグをロック値にセットするステップと、
ロック・フラグがロック値にセットされている間、どのプロセッサによるデータ・ブロックへのアクセスも禁止し、そのようなプロセッサによるデータ・ブロックへのアクセスまたはロックを求める要求を中断アクセス要求待ち行列（１６２）に入れるステップと、
第１のプロセッサの要求に応じてロック・フラグをロック解除値にリセットするステップと、
ロック・フラグがロック解除値にリセットされたとき、中断アクセス要求待ち行列中の要求を処理するステップとを含む請求項１に記載のメモリ共用方法。
【請求項６】
さらに、プロセッサの要求に応じて、大域メモリ中のデータに対して大域リダクション演算を実行し、大域リダクション演算の結果を大域メモリの指定された位置に記憶するステップを含む請求項１に記載のメモリ共用方法。
【請求項７】
さらに、データ・ブロックの複数の大域コピーを、複数の大域メモリのそれぞれに１つずつ記憶するステップと、
複数の大域メモリのそれぞれに複数のプロセッサを関連付けるステップと、
プロセッサがデータ・ブロックへのアクセスを要求するとき、そのようなプロセッサに関連する大域メモリ中のデータ・ブロックの大域コピーにアクセスするステップとを含み、
大域コピーにアクセスするステップが、
アクセス側プロセッサのローカル・メモリに大域コピーのローカル・コピーを記憶するステップと、
ローカル・コピーがローカル・メモリからフラッシュされたとき、アクセス側プロセッサに関連する大域メモリ中の大域コピーとローカル・コピーを併合するステップと、
１つの大域コピーのローカル・コピーがすべてそのような大域コピーと併合された後に、そのような大域コピーを他の大域コピーと併合するステップとを含む請求項１に記載のメモリ共用方法。
【請求項８】
共用メモリ多重処理コンピュータ・システムであって、
それぞれ複数のマルチビット・データ要素を備える共用データの複数のデータ・ブロックを記憶するための大域メモリを備える、１つまたは複数の大域メモリ装置と、
それぞれ、最初、ローカル・コピーがローカル・メモリに記憶されるときに非書込み値

に設定される、ローカル・コピーのデータ要素と１対１の関係で関連付けられた複数のフラグを備える書込みビットマスクと、１つのデータ・ブロックのローカル・コピーとを、データ・ブロック中のデータへのアクセスがプロセッサによって要求されたときに記憶するためのローカル・メモリと、前記プロセッサとを備える１つまたは複数の大域メモリ装置と相互接続された複数の処理装置とを備え、

各処理装置がさらに、それぞれ、プロセッサによって書込みが行われたデータ要素に関連する書込みビットマスク・フラグを書込み値にセットし、ローカル・コピーがローカル・メモリにおいて他のデータと置換されたとき、プロセッサによってデータ要素のどれかに書込みが行われている場合に書込みビットマスクおよび書込みが行われたデータ要素を大域メモリ装置に送るように動作するローカル・メモリ制御装置を備え、前記コンピュータ・システムがさらに、

関連する書込みビットマスク・フラグが書込み値にセットされた各データ要素をデータ・ブロックに記憶することによって、１個の処理装置から受け取ったローカル・コピーを併合するように動作し、複数のプロセッサが同時にデータ・ブロックにアクセスしたときに、少なくとも２個のプロセッサが、共に書込み値にセットされ、かつプロセッサ自体のそれぞれのローカル・コピーの対応する位置にあるデータ要素に関連する書込みビットマスク・フラグを送る書込み／書込みデータ・レースの発生を識別するように動作する大域メモリ制御装置を大域メモリ装置中に備える共用メモリ多重処理コンピュータ・システム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】

本発明は、全般的には、メモリに記憶されたデータが複数のプロセッサ間で大域的に共用されるコンピュータ・システムに関し、詳細には、個別のプロセッサにキャッシュされた共用データのコヒーレンスを維持するコンピュータ・システムのメモリ共用方法に関する。

【０００２】

【従来の技術】

多重処理コンピュータ・システムでは、単一のプログラムの各部分を構成する複数のプロセスの同時実行を可能にするように複数の独立のプロセッサが相互接続されている。そのようなコンピュータ・システムは、データ資源またはメモリ資源を共用する複数の別々のプログラムを実行するために使用することができる。

【０００３】

すべてのプロセッサによって大域的にアクセス可能なメモリにデータが記憶される共用メモリ多重処理システムは、多数のプログラマが好むフラット・メモリ・モデルの利点を提供する。フラット・メモリ・モデルを含むシステムでは、大域仮想アドレス指定を使用する個別のプロセッサによってデータにアクセスすることができる。これに対して、ある種の分散メモリ多重処理システムでは、個別のプロセッサしかアクセスできないメモリにデータが記憶される。そのようなシステムでは、あるプロセッサが他のプロセッサのメモリ中のデータにアクセスするには、前記他のプロセッサ上で走るプログラムと直接通信するしかない。したがって、プログラマのタスクは、プロセッサ間でメッセージを交換し、プロセッサ間でデータの位置を追跡できるようにしなければならないので、より複雑である。

【０００４】

多数の共用メモリ多重処理システムでは、各プロセッサが高速メモリ・アクセス用のローカル・メモリまたはキャッシュを備えている。大域メモリとプロセッサが物理的に離れているので、大域メモリ中のデータへの直接アクセスは比較的低速である。キャッシュは、プロセッサによって必要とされる共用データの一時記憶のためのより高速の中間メモリ・レベルを提供する。プロセッサが大域メモリのデータ・ブロックのデータにアクセスするとき、データ・ブロック全体が伝送され、プロセッサのキャッシュに記憶される。大域メ

モリからブロックを得なければならないので最初のデータ・アクセスはまだ低速であるが、同じデータ・ブロックへの以後のアクセスはキャッシュのより速い速度で行われる。プロセッサによる大部分のデータ・アクセスは時間的および空間的にローカル化される。すなわち、所与の期間中に、プロセッサによる大部分のデータ・アクセスは同じデータ・ブロックに対して行われる。したがって、キャッシングでは、すべてのデータ・アクセスを直接大域メモリに対して行う場合よりもずっと速い平均速度でデータがアクセスされる。

【０００５】

しかし、キャッシュを使用する共用メモリ多重処理システムは、キャッシュ・コヒーレンスを維持する次の問題に対処しなければならない。複数のプロセッサが同じデータ・ブロックのコピーを有し、キャッシュ・コピーが１個または複数のプロセッサによって修正されるとき、キャッシュされたデータ・ブロックには一貫性がなくなる。このため、プロセッサがそれぞれのプロセスを実行すると誤った結果が得られる恐れがある。

【０００６】

キャッシュ・コヒーレンスの欠如はたとえば、２個のプロセッサがそれぞれのキャッシュに同じデータ・ブロックのコピーを有し、それらのプロセッサが共にブロック中の同じデータ要素を修正したときに発生する。そのような状況では、プロセッサのキャッシュ・コピーは、相互に一貫性がなくなり、かつ大域メモリ・データ・ブロックとも一貫性がなくなる。どのプロセッサのコピーが大域メモリを最後に更新したかに応じて、大域メモリ・データ・ブロック中のデータ要素は一方のプロセッサまたは他方のプロセッサによって書き込まれたデータを有する（必ずしも、実際にプロセッサによって最後に書き込まれた値ではない）。これは、書込み／書込みデータ・レースと呼ばれることがある。

【０００７】

キャッシュ・コヒーレンスの欠如は、データ・ブロックの同じデータ要素に対して第１のプロセッサが書込みを行い、第２のプロセッサが読取りを行ったときにも発生する。第２のプロセッサのデータ・ブロックのキャッシュ・コピーが、大域メモリ中のデータ・ブロックが第１のプロセッサの書き込んだ値によって更新される前に得られたか、あるいは後に得られたかに応じて、第２のプロセッサはデータ要素の最初の値または第１のプロセッサによって書き込まれた値を読み取る。これは、読取り／書込みデータ・レースと呼ばれることがある。

【０００８】

キャッシュ・コヒーレンスの問題は広く研究されており、多数の解決策が利用可能である。これらの解決策の多くは順次一貫性を拡張することによって機能する。そのような解決策の１つは、単に共用データのキャッシングをバイパスすることである。他のプロセッサと共用するデータ要素への各アクセスは直接、大域メモリに対して行われる。このように、プロセッサは厳密に順次、共用データにアクセスする。この解決策は、直接大域メモリ・アクセスがキャッシュ・データ・アクセスよりもずっと低速であり、あるいは大量の共用データがあるシステムでは重大な速度上の欠点を示す。

【０００９】

書込み時無効化と呼ばれる順次一貫性の他の変形例は、複数のプロセッサによるデータ・ブロックの同時キャッシングを許可するが、システム中の他のすべてのプロセッサに無効化信号を送るように、データ・ブロックに書込みを行うあらゆるプロセッサに要求するものである。他のプロセッサは、無効化信号を受け取ると、次にデータ・ブロックにアクセスする前に、データ・ブロックの更新済みバージョンを大域メモリから得ておかなければならない。書込み時更新と呼ばれる順次一貫性の他の変形例は、書込みが実行されるたびに、書込み側プロセッサのデータ・ブロックの修正をシステム中の他のプロセッサに送るように前記書込み側プロセッサに要求する。

【００１０】

書込み時無効化方式と書込み時更新方式は共にあまり効率的ではない。どちらの方式の下でも、データ・ブロックが擬似的に共用されるとき、必要でないにもかかわらず無効化メッセージまたは更新メッセージが送られる。偽共用と呼ばれるこの状況では、複数のプロ

セッサが同じデータ・ブロックをキャッシュするが、書込み側プロセッサによって修正されたデータ要素は他のどんなプロセッサからもアクセス（読取りまたは書込み）されない（すなわち、データ・ブロックは共用されるが、データ要素は共用されない）。他のプロセッサによって修正されたデータ要素にどのプロセッサもアクセスしない限り、衝突は発生しない。偽共用は無効化メッセージおよび更新メッセージが発生する原因の大部分を占めるので、これらの方式では不要な通信トラフィックが過度なものになる。

【００１１】

偽共用の性能上の影響は、プロセッサが、データ・ブロックの一貫性のないキャッシュ・コピーを一時的に維持できるようにする、キャッシュ・コヒーレンス解決策によって低減することができる。一貫性のないキャッシュ・コピーは後で、プログラムされた同期点で調整される（すなわち、更新が伝搬される）（同期点はたとえば、プログラム中に置かれたロック操作やバリヤ操作などの適当な同期操作を使用することによってセットすることができる）。このような同期は、上述の書込み／書込みデータ・レースおよび読取り／書込みデータ・レースをなくすように妥当にプログラムしなければならない。しかし、同期の間に、複数のプロセッサがそれぞれのキャッシュ中の同じデータ・ブロックの異なる部分を修正することができる。遅延一貫性と呼ばれるこの種の解決策は、偽共用に関連する通信トラフィックを低減する。

【００１２】

１つの遅延一貫性解決策がA.Karp（エイ・カープ）およびV.Sarkar（ブイ・サーカ）著「Data Merging for Shared Memory Multiprocessors（共用メモリ複数プロセッサ用データ併合）」）（HP Laboratories Technical Report、（HP研究所技術報告）1992年11月）で提案されている。Karp-Sarkar （カープ・サーカ）データ併合では、複数のプロセッサが、それらのキャッシュ中の大域メモリ・データ・ブロックのコピーを記憶し、それらのそれぞれのキャッシュ・コピーの別々の部分を独立に修正することができる。各キャッシュ・コピーは後で、再び大域メモリに伝送され、データ・ブロックと併合される。

【００１３】

キャッシュ・コピーと大域メモリ・データ・ブロックの併合は、大域メモリ・データ・ブロックの各バイトごとに１つのビット・フラグを含むビットマスクを使用することによって実行される。ビットマスク中の各フラグは、大域メモリ・データ・ブロックの関連するバイトが併合において修正されたかどうかを示す。ビットマスクは、最初はフラグがクリアされており、ブロックが初めてどれかのプロセッサによってキャッシュされたときに大域メモリ・データ・ブロックに割り当てられる。プロセッサによって修正され、次いで（たとえば、他のデータ・ブロックのコピーと置換できるように）プロセッサのキャッシュからフラッシュされたキャッシュ・コピーは、再び大域メモリに伝送され、発信側大域メモリ・データ・ブロックと併合される。

【００１４】

修正された各キャッシュ・コピーは、まだセットされていないビットマスク中の各フラグごとに、発信側データ・ブロックの関連するバイトとキャッシュ・コピーの対応する位置にあるバイトを比較することによって、発信側データ・ブロックと併合される。バイトが異なる場合、キャッシュ・コピー中のバイトは修正されている。したがって、発信側データ・ブロック中のバイトの代わりに、修正されたキャッシュ・コピーが記憶される。このバイトに関連するフラグも、前記バイトが修正されたことを示すようにセットされる。しかし、バイトが同じである場合、フラグはクリアされたままであり、修正されていないキャッシュ・バイトは大域メモリ・データ・ブロックに記憶されない。したがって、修正された別々の部分を有するデータ・ブロックの２つ以上のキャッシュ・コピーを大域メモリで併合することができる。

【００１５】

また、Karp-Sarkar データ併合システムは、キャッシュ中にブロックのコピーがあるプロセッサの数を追跡するカウンタを各大域メモリ・データ・ブロックごとに維持している。カウンタは、プロセッサがブロックのコピーをキャッシュするたびに増分される。プロセ

ッサは、データ・ブロックのコピーがプロセッサ自体のキャッシュからフラッシュされたときには、前記コピーが修正されなかったときでも、必ず大域メモリに通知し、それによって、カウンタを減分できるようにする。データ・ブロックのビットマスクは、この数を追跡することによって、データ・ブロックのコピーを有するビットマスクの数が０に戻るときに再初期設定することができる。

【００１６】

Karp-Sarkar のデータ併合法の１つの欠点は、修正されたキャッシュ・コピーが大域メモリ・データ・ブロックと併合された後、併合が完了する（すなわち、ブロックのキャッシュ・コピーがすべて併合される）までどのプロセッサもブロックの他のコピーをキャッシュできないことである。併合が始まった後のデータの次のキャッシングを防ぐために、中断機構が提供されている。大域メモリ・データ・ブロックに中断ビットが関連付けられている。キャッシュ・コピーが初めてブロックと併合され、ブロックの他のコピーがキャッシュされたままであるとき、中断ビットがセットされる。中断ビットがセットされている間、ブロックへのアクセスを求めるどんな要求も中断待ち行列に入れられる。要求は併合が完了する（すなわち、キャッシュ・コピーを有するプロセッサの数が０になる）まで中断されたままであり、完了の時点で、要求が満たされ、ビットがクリアされる。

【００１７】

【発明が解決しようとする課題】

Karp-Sarkar 併合法では、併合が始まった後の大域メモリ・データ・ブロックへの次のアクセスを防ぐことが必要である。なぜなら、プロセッサは、データ・レースを発生させずに、プロセッサ自体のプログラムに従って同期点間で大域メモリ・データ・ブロックの特定のバイトを数回、妥当に修正できるからである。そのような修正の間に、キャッシュを使用して他のブロックのコピーを記憶できるように、プロセッサのブロックのキャッシュ・コピーをキャッシュからフラッシュすることができる。しかし、修正されたキャッシュ・コピーは、フラッシュされた後、大域メモリ・ブロックと併合され、修正されたバイトに関連するフラグがセットされる。プロセッサが再びブロックのコピーをキャッシュしてバイトを修正できる場合、大域メモリ・データ・ブロックと前記コピーを再び、妥当に併合することはできない（データ・ビットがすでにセットされているので、次のバイト修正は大域メモリに記憶されない）。中断機構によって、プロセッサは次の同期の前に同じバイトを再びキャッシュして修正することができる。しかし、これは、併合が完了してバイトに関連するフラグがクリアされた後に限られる。

【００１８】

【課題を解決するための手段】

本発明は、Karp-Sarkar 式データ併合のこの欠点を補正する（すなわち、データ・ブロックの併合が始まった後にデータ・ブロックへのアクセスを求める要求を中断する要件を不要にする）コンピュータ・システムのメモリ共用方法を提供する。本発明によれば、データ併合は、発信側データ・ブロックの各キャッシュ・コピーとビットマスクを関連付けることによって行われる。ビットマスクは、どのキャッシュ・コピー・データ要素が修正されたかを示すためにキャッシュ・コピー中の各データ要素ごとに１つある複数のフラグから成る。ビットマスク・フラグは、コピーが初めてキャッシュされるときに初期設定される。次いで、キャッシュ・コピーのデータ要素が修正されると、前記データ要素に関連するフラグは、書き込まれた値にセットされる。

【００１９】

キャッシュ・コピーがその後キャッシュからフラッシュされると、ビットマスクは、キャッシュ・コピーのデータ要素が大域メモリ・データ・ブロックに記憶されているかどうかを判定する。関連するフラグによって、修正されたことが示された各キャッシュ・コピー・データ要素は、大域メモリ・データ・ブロックに記憶される。したがって、併合時に、キャッシュ・コピーのデータ要素は、それが修正されているときには、発信側データ・ブロックの対応する位置に記憶され、発信側データ・ブロック要素が未修正のままであるときには記憶されない。そのような方法では、プロセッサは、大域メモリ・ブロックのコピ

ーを首尾よくキャッシュし、併合させ、併合が完了するのを待たずに、再びブロックのコピーをキャッシュすることができる。

【００２０】

本発明の一実施例では、ビットマスク・フラグに関連するキャッシュ・コピー・データ要素はバイトである（「バイト」の語は、本明細書では、一般に７つまたは８つの２進数（「ビット」）を有するが、必ずしもそうでなくてもよいマルチビット・データ要素を示すために使用される）。本発明の他の実施例では、ビットマスク・フラグに関連するキャッシュ・コピー・データ要素はデータ・ワードである（「データ・ワード」の語は、本明細書では、一般に４バイトであるが、必ずしもそうでなくてもよい整数個のバイトから成るデータ単位を示すために使用される）。キャッシュ・コピーのワードだけに関連するフラグを有するビットマスクを維持することによって、ビットマスクを記憶するのに必要とされるプロセッサ・キャッシュ中の記憶域は少なくなる。

【００２１】

本発明の他の態様によれば、ビットマスク・フラグがキャッシュ・コピー・データ・ワードに関連付けられる本発明の実施例では、サブワード記憶フラグもキャッシュ・コピーに関連付けられる。サブワード記憶フラグは、コピーがキャッシュされたときに初期設定され、続いて、プロセッサが、キャッシュされたコピー上でサブワード記憶動作（すなわち、ワード全体よりも少ない量のデータを記憶する動作）を実行するときにセットされる。キャッシュ・コピーは次いで、Karp-Sarkar データ併合法と同様に大域メモリ・データ・ブロックの各バイトごとにフラグがあるビットマスクを使用して併合される。大域メモリ・ビットマスクは、動的に割り振られたメモリに記憶することができる。本発明のこの態様は、サブワード・メモリ・アクセスを可能にしながら、キャッシュ記憶オーバヘッドを低減する。

【００２２】

本発明の他の態様によれば、読取りビットマスクおよび書込みビットマスクがキャッシュに維持される。読取りビットマスクおよび書込みビットマスクはそれぞれ、関連するデータ要素が、それぞれ読み取られ書き込まれるときにセットされる、各キャッシュ・コピー・データ要素ごとのフラグを含む。キャッシュ・コピーが大域メモリ・データ・ブロックと併合された後、デバッグやその他の目的のために読取り／書込みデータ・レースまたは書込み／書込みデータ・レースの発生を識別できるようにビットマスクが保持される。

【００２３】

本発明の他の態様によれば、各大域メモリ・データ・ブロックにロック・フラグが関連付けられる。ロック・フラグはロック側プロセッサの要求時にセットされる。ロック・フラグがセットされると、データ・ブロックへのアクセスの次の要求は中断要求待ち行列に入れられる。ロック・フラグはロック側プロセッサから要求された時にしかクリアできず、要求があった時点で、中断されていた要求が処理される。

【００２４】

本発明の他の態様によれば、大域メモリ中のデータに対してリダクション動作が実行される。大域メモリに対するそのようなリダクション動作は、大域メモリ制御装置によって実行することができる。

【００２５】

本発明の他の態様によれば、複数の大域メモリ装置を有するコンピュータ・システムにおいて、第１の大域メモリに記憶されたデータ・ブロックのコピーが第２の大域メモリにも記憶される。その場合、プロセッサはどちらかの大域メモリ装置からデータ・ブロックのキャッシュ・コピーを得る。すべてのキャッシュ・コピーが再び、データ・ブロックの大域メモリ・コピーと併合されると、大域メモリ・コピー同士しも併合される。複数の大域メモリ装置を階層として構成し、大域メモリ・コピーを１つ上のレベルの大域メモリで併合することが好ましい。この態様は、１つの大域メモリのデータ・ブロックに多数のプロセッサがアクセスしているときに、その大域メモリを停止して同時に他のプロセッサを遊休状態にするのを避けるために適用することができる。

【００２６】
本発明の他の特徴および利点は、添付の図面に関する好ましい実施例の詳細な説明から明らかになろう。
【００２７】
【実施例】
図１を参照すると、本発明の第１の実施例で、共用メモリ多重処理コンピュータ・システム２０は、複数のプロセッサ２２ないし２４と、１つまたは複数の大域メモリ装置２６、２７とを備えている。バス、クロスバー・スイッチ、多段ネットワーク、ローカル・エリア・ネットワークなどの相互接続通信媒体２８は、プロセッサ２２ないし２４と大域メモリ装置２６、２７を相互接続する。通信媒体、すなわち、相互接続媒体２８は、プロセッサ２２ないし２４と大域メモリ装置２６、２７の間でデータを転送するために使用される。
【００２８】
各大域メモリ装置２６、２７は、大域メモリ制御装置３０、３１と大域メモリ３２、３３とを有する。大域メモリ３２、３３は、プロセッサ２２ないし２４が共用すべきデータの記憶域を提供し、データ併合に関係するデータを記憶することもできる。大域メモリ制御装置３０、３１は、プロセッサからの大域メモリ要求に応答し、データ併合動作を実行する。大域メモリ制御装置３０、３１は状態マシンとして実施することができる。しかし、大域メモリ制御装置３０、３１は、本発明に従って動作するようにプログラムされた汎用処理回路で実施することが望ましい。
【００２９】
各プロセッサ２２ないし２４は、中央演算処理装置３６ないし３８と、ローカル・メモリ制御装置４０ないし４２と、ローカル・メモリまたはキャッシュ・メモリ４４ないし４６（以下、ローカルメモリとして説明を進める）とを有する。プロセッサ２２ないし２４は、共通プログラム化タスクの各部分を形成できるプロセスを個別に、プロセッサ自体のそれぞれの中央演算処理装置３６ないし３８で実行する。プロセッサの中央演算処理装置３６ないし３８で実行中のプロセスが、１つの大域メモリ装置２６、２７に記憶された共用データへのアクセスを必要とするとき、プロセッサのローカル・メモリ制御装置４０ないし４２はその大域メモリ装置にデータ要求を発行する。大域メモリ装置２６、２７から１つのプロセッサ２２ないし２４によって受け取られたデータは、前記プロセッサのローカル・メモリ４４ないし４６に記憶される。
【００３０】
大域メモリ３２、３３に記憶された共用データは、それぞれ、一般に多数のバイトから成る複数のデータ・ブロックを備えている。プロセッサ２２ないし２４は、共用データにアクセスするのに必要とされる平均時間を短縮するために、プロセッサ自体のローカル・メモリ４４ないし４６に共用データ・ブロックのコピーをキャッシュすることができる。ローカル・メモリ４４ないし４６は、中央演算処理装置３６ないし３８の命令実行速度と同程度の高速アクセス時間を提供できるランダム・アクセス半導体型メモリである。大域メモリ３２、３３は比較的高速の半導体型メモリでもよいが、光学媒体データ記憶装置や磁気媒体データ記憶装置など低速の大容量記憶装置を含むことができる。したがって、プロセッサ２２ないし２４の中央演算処理装置３６ないし３８が大域メモリ装置２６、２７中の共用データへのアクセスを必要とするとき、プロセッサのローカル・メモリ制御装置４０ないし４２は大域メモリ装置にデータ要求を発行する。大域メモリ装置中の大域メモリ制御装置３０、３１は、要求されたデータを含むデータ・ブロックのコピーをプロセッサに伝送することによって応答する。そうすると、プロセッサのローカル・メモリ制御装置４０ないし４２は、プロセッサのローカル・メモリ４４ないし４６にデータ・ブロック・コピーを記憶する。
【００３１】
共用メモリ多重処理コンピュータ・システム２０では、複数のプロセッサ２２ないし２４が、同じメモリ・データ・ブロックのコピーをローカル・メモリ４４ないし４６にキャッ

シュし、そのようなコピーの各部分を独立に修正することができる。キャッシュ・コピーを独立に修正する結果、コピーが異なるものになることがある。キャッシュ・コピー間のこのような違いは、以下で説明するデータ併合によって解決される。

【００３２】

共用メモリ多重処理コンピュータ・システム２０中の複数のプロセッサによって共用データ・ブロックのコピーをキャッシュして修正することができるので、読取り／書込みデータ・レースおよび書込み／書込みデータ・レースが発生することがある。これらのレースは、コンピュータ・システム２０において適切な同期によってプロセッサ２２ないし２４上で走るプロセスをプログラムすることによって妨げられる。書込み／書込みデータ・レースをなくすには、プログラムされた同期点間に２個のプロセッサが同じデータ要素を修正することがないように、プロセスを適切にプログラムしなければならない。プログラマは、どのプロセッサも、プログラムされた同期点間に他のプロセッサによって修正された（すなわち、書込みが行われた）共用ブロックのデータ要素を読み取らないようにすべきである。したがって、共用メモリ多重処理コンピュータ・システム２０で複数のプロセッサが同期の間に同じデータ要素を修正するのは、プログラマのエラーの場合だけである。共用データ要素へのアクセスの同期はたとえば、従来型のロック命令またはバリヤ命令や、その他の関連機構によって行うことができる。

【００３３】

図２を参照すると、本発明の第１の実施例によれば、データ併合は、キャッシュ・コピーのどのデータ要素が修正されたかを追跡することによって行われる。共用メモリ多重処理コンピュータ・システム２０では、ローカル・メモリ制御装置４０（図１）が大域メモリ装置２６からローカル・メモリ４４にデータ・ブロック（「発信側データ・ブロック」）のコピー５４を記憶するとき、ローカル・メモリ制御装置は書込みビットマスク５６もローカル・メモリに記憶する。書込みビットマスク５６は、コピー５４中の複数のデータ要素７０ないし７４のそれぞれに１つある複数のフラグ６０ないし６４から成る。フラグ６０ないし６４は、図２で矢印によって示したように、データ要素７０ないし７４と１対１の関係で関連付けられている。図２に示した実施例では、各フラグは単一ビットであるが、代替実施例では複数のビットを備えることができる。また、図の実施例では、ビットマスク・フラグと１対１で関連付けられたデータ要素はバイトである。しかし、図３に示し、かつ以下で説明する実施例で示されるように、ビットマスク・フラグと１対１で関連付けられたデータ要素は、ワードでも、共にキャッシュ・コピーを形成する他のどんなサイズのデータ単位でもよい。

【００３４】

ローカル・メモリ制御装置４０は、修正されたデータ要素に関連するフラグ６０ないし６４をセットすることによって、キャッシュ・コピー５４のデータ要素７０ないし７４が修正されたかどうかを追跡する。ローカル・メモリ制御装置４０は最初に、フラグ６０ないし６４を非書込み値（すなわち、フラグに関連するデータ要素７０ないし７４が修正されていないことを示す値）にセットする。図の実施例では、単一ビット・フラグ６０ないし６４の非書込み値は０である。中央演算処理装置３６が後でデータ要素７０ないし７４のうちのどれかを修正すると、ローカル・メモリ制御装置４０は、そのような修正されたデータ要素に関連するフラグを、修正値（たとえば、図の実施例中のもの）にセットする。たとえば、中央演算処理装置３６がデータ要素７１および７２に書込みを行うことによってこれらのデータ要素を修正した場合、関連するフラグ６１、６２はそれぞれ、１にセットされる。その結果、ビットマスク・フラグ６０ないし６４は、どのキャッシュ・コピー・データ要素７０ないし７４が修正されたかを示す。

【００３５】

キャッシュ・コピー５４がローカル・メモリからフラッシュされると、ローカル・メモリ制御装置４０は、ビットマスク５６およびキャッシュ・コピー５４を大域メモリ装置２６に伝送する。キャッシュ・コピー５４のフラッシュはたとえば、中央演算処理装置３６が他のデータ・ブロック・コピー中のデータにアクセスできるように、ローカル・メモリ制

御装置４０がキャッシュ・コピーを他のデータ・ブロック・コピーと置換するときに行われる。いくつかの同期機構は、ローカル・メモリ制御装置４０に無効化信号を送ることによって、キャッシュ・コピー５４のフラッシュをトリガすることもできる。また、実施例によっては、ローカル・メモリ制御装置４０は、データ併合に必要とされる情報しか伝送せず、それによって、通信オーバヘッドを低減する。たとえば、ローカル・メモリ制御装置は、ビットマスク５６と、修正されたキャッシュ・コピー５４のデータ要素だけを伝送する。

【００３６】

大域メモリ装置２６では、大域メモリ制御装置３０（図１）は、キャッシュ・コピー５４をその発信側データ・ブロックと併合する際にビットマスク５６を使用する。大域メモリ制御装置３０は、修正値にセットされるビットマスク５４中の各フラグ６０ないし６４ごとに、キャッシュ・コピーの関連するデータ要素７０ないし７４を発信側データ・ブロック中に記憶する。たとえば、図２の図中のフラグ６１、６２が１（すなわち、修正値）にセットされているので、関連するデータ要素７１、７２は発信側データ・ブロックの対応する位置に記憶される。これによって、複数のキャッシュ・コピーの大域メモリ・データ・ブロックで一貫した併合が行われる。というのは、キャッシュ・コピーの修正された部分だけがデータ・ブロックを更新し、かつプログラミングの制約によって、複数のプロセッサがデータ・ブロックの同じ部分を修正することが禁止されるからである。

【００３７】

上記の従来の技術で説明したように、従来型のKarp-Sarkar データ併合法の１つの欠点は、プロセッサが、キャッシュ・データ・ブロックの要素を修正し、このブロックを併合し、ブロックのすべてのコピーが併合される前に再びこのデータ・ブロックをキャッシュすることが妨げられることである。これは、Karp-Sarkar 型のデータ併合では、プロセッサがデータ・ブロック中の同じ要素を再び修正することができ、そのような以後の修正が大域メモリに記憶されないためである。Karp-Sarkar データ併合では、大域メモリ中のビットマスクが、大域ブロックのどの要素が修正されたかを追跡する。キャッシュ・コピーの要素は、それが大域ブロックの対応する位置にある未修正の要素と異なる場合だけ記憶される。特に、プロセッサが、そのデータ・ブロックの第１のキャッシュ・コピー中のデータ要素を修正すると、大域ブロックの対応する位置にある要素が更新され、関連するフラグがセットされる。フラグがセットされた後、ブロックのキャッシュ・コピーがすべて併合され、フラグがリセットされるまで、要素に対する以後の修正によって大域ブロックを更新することはできない。このため、Karp-Sarkar データ併合システムは、データ・ブロックが初めて併合された後、ブロックの以前にキャッシュされたコピーもすべて併合されるまで、プロセッサがデータ・ブロックをキャッシュすることを妨げる中断機構を使用する。

【００３８】

しかし、遅延一貫性を使用するシステムでは、同期の間の同じプロセッサによるデータ要素の複数回の修正が可能である。複数のプロセッサによる同じデータ要素の修正だけがエラーとみなされる。その場合、Karp-Sarkar データ併合法は、望ましくなく不当に長い遅延をプロセッサに課し、プロセッサは、そのブロックの第１のキャッシュ・コピーをフラッシュした後に、再びデータ・ブロックの同じ要素を修正しなければならない。

【００３９】

本発明のデータ併合法では、プロセッサはデータ・ブロックを何度でもキャッシュして修正することができる。キャッシュ・コピーの修正された要素はすべて、常に記憶される。したがって、プロセッサが、併合が始まってから完了するまでにデータ・ブロックのコピーをさらにキャッシュするのを妨げる必要はない。したがって、中断機構は必要とされない。

【００４０】

図３を参照すると、本発明の第２の実施例によれば、共用メモリ多重処理コンピュータ・システム２０中のローカル・メモリ制御装置４０ないし４２は、フラグ９０ないし９２が

ワード８０ないし８２と１対１の関係で関連付けられた「ワード」書込みビットマスク８８を使用して、キャッシュ・コピー８６のどのワード８０ないし８２が修正されたかを追跡する。各ワード８０ないし８２は、整数個のバイト９６－１０７から成る（たとえば、図３に示した実施例ではワード当たり４バイト）。どのワードが修正されたかを追跡するために、フラグ９０ないし９２は最初、非書込み値にセットされる。次いで、キャッシュ・コピー８６中のワード８０ないし８２が修正されると、これらのワードに関連するフラグ９０ないし９２がそれぞれ、書込み値にセットされる。たとえば、図３に示したように、フラグ９０ないし９２が最初０にセットされている場合、フラグ９１は、ワード８１が修正されたときに、それが修正されたことを示す１にセットされる。

【００４１】

第２の実施例では、フラグ９０ないし９２をキャッシュ・コピーのワードと関連付けることによって、キャッシュ・コピーへの修正を追跡することに関連する記憶域オーバヘッドが低減される。これに対して、キャッシュ・コピーの各バイトごとにビットマスク・フラグが維持される図２に示した第１の実施例では、ビットマスク・フラグ用に約４倍の記憶域が必要である。ワードごとにフラグを維持すると特に、中央演算処理装置３６ないし３８がワード・サイズの増分単位でデータにアクセスする場合に好都合である。現在では極めて一般的なそのような中央演算処理装置では、キャッシュ・コピー８６中のどのバイト９６ないし１０７が修正されたかをビットマスク８８中のバイトごとのフラグによって追跡することは冗長である。

【００４２】

引き続き図３を参照すると、ワード書込みビットマスク８８を維持する本発明の代替（「第３の」）実施例では、サブワード記憶動作に適応するためにサブワード記憶フラグ１１０もキャッシュ・コピー８６に関連付けられる。通常、データにワードごとにアクセスするいくつかの中央演算処理装置３６ないし３８は、１ワードよりも少ない量の増分単位でデータを修正する（たとえば、一度に１バイトだけ修正する）動作（以下、「サブワード記憶動作」と呼ぶ）を実行することができる。そのような中央演算処理装置を使用するコンピュータ・システム２０では、通常、データ併合をワード・レベルで実行して記憶域および伝送のオーバヘッドを低減し、しかもサブワード記憶動作が使用されるときにバイト・レベル・データ併合を行うことが望ましい。

【００４３】

したがって、キャッシュ・コピー８６がローカル・メモリ４４に記憶されるとき、ローカル・メモリ制御装置４０はサブワード記憶フラグ１１０もローカル・メモリ４４に記憶する。サブワード記憶フラグ１１０は、記憶されると、最初は、キャッシュ・コピー８６に対してサブワード記憶動作が実行されていないことを示す「no subword stores occurred（サブワード記憶を実行せず）」値にセットされる。そのようなとき、中央演算処理装置がサブワード記憶動作を実行すると、ローカル・メモリ制御装置４０が、そのような動作が実行されたことを示す「subword store occurred（サブワード記憶を実行する）値にサブワード記憶フラグをセットする。また、ローカル・メモリ制御装置４０は、サブワード記憶動作によって修正されたワードに関連するワード書込みビットマスク・フラグを、書込み値にセットする。図の実施例では、サブワード記憶フラグは単一ビットであり、「no subword stores occurred」値は「０」であり、「subword store occurred」値は「１」である。たとえば、中央演算処理装置３６が、ワード８１の一部であるバイト１０２にデータを書き込むサブワード記憶動作を実行する場合、ローカル・メモリ制御装置はサブワード記憶フラグ１１０を１にセットし、ワード８１に関連する書込みビットマスク・フラグ９１も１にセットする。サブワード記憶動作によってバイト１０２が修正された後のワード・ビットマスクおよびサブワード記憶フラグの状態を図３に示す。

【００４４】

続いてキャッシュ・コピー８６がローカル・メモリ４４からフラッシュされると、ローカル・メモリ制御装置４０は、キャッシュ・コピー８６、キャッシュ・コピー書込みビットマスク８８、およびサブワード記憶フラグ１１０を、発信側データ・ブロック１１８が記

憶されている大域メモリ装置２６に伝送する。サブワード記憶フラグ１１０が「no subword store occurred 」値にセットされた場合、ワード書込みビットマスク８８中の関連するフラグ９０ないし９２が、書込み値にセットされた各ワード８０ないし８２を、発信側データ・ブロック中の対応する位置に記憶することによって、キャッシュ・コピー８６が発信側データ・ブロック１１８と併合される。しかし、サブワード記憶フラグ１１０が「subword store occurred」値にセットされた場合、併合されるのはキャッシュ・コピー８６の修正されたバイトだけである。

【００４５】

大域メモリ制御装置３０は、修正されたバイトだけを併合するために、一般に、発信側データ・ブロックの各バイトごとのフラグから成る、サブワード書込みビットマスクを大域メモリに記憶する。各フラグは最初、非書込み値にセットされる。大域メモリ制御装置３０は、関連するワード書込みビットマスク・フラグ９０ないし９２が、書込み値にセットされたキャッシュ・コピー８６の各ワード８０ないし８２ごとに、発信側データ・ブロックの対応する位置にあるバイトとそのようなワードのバイトを比較する。バイトが異なり、発信側ブロック・バイトに関連するサブワード書込みビットマスク・フラグがまだ非書込み値にセットされている場合、大域メモリ制御装置３０は、発信側データ・ブロック・バイトの代わりにキャッシュ・コピー・バイトを記憶し、発信側データ・ブロック・バイトに関連する大域書込みビットマスク中のフラグを書込み値にセットする。続いて、他のプロセッサ２３ないし２４にキャッシュされたデータ・ブロックの追加コピーも同様に併合される。

【００４６】

たとえば、図４に示したように、キャッシュ・コピー８６（図３）の発信側データ・ブロック１１８はワード８０ないし８２（図３）の位置に対応する位置にあるワード１２０ないし１２２で構成されている。サブワード記憶フラグ１１０（図３）が、コピー８６に対してサブワード記憶動作が実行されたことを示す１にセットされると、大域書込みビットマスク１２４を使用してキャシュウ・コピー８６とブロック１１８が併合される。ビットマスク１２４は、発信側データ・ブロック１１８の各バイト１４６ないし１５７ごとのフラグ１２６ないし１３７を備えている。併合は、修正された各キャッシュ・コピー・ワード８１（関連するフラグ９１が１にセットされることで示される）のバイト１００ないし１０３（図３）を、関連するフラグ１３０ないし１３３がまだ０にセットされている（未修正であることを示す）発信側データ・ブロック１１８の対応する位置にあるバイト１５０ないし１５３と比較することによって行われる。関連するフラグ１３０ないし１３３が０にセットされたバイト１５０ないし１５３と異なるバイト１００ないし１０３は記憶され、関連するフラグは１にセットされる。

【００４７】

大域メモリ制御装置３０は、サブワード書込みビットマスク用の記憶空間を大域メモリの空きメモリ部分から動的に割り振ることが好ましい。これによって、データ・ブロック用のサブワード書込みビットマスクはデータ併合に必要とされるときにしか大域メモリに記憶されないので、記憶域オーバヘッドが最小限に抑えられる。

【００４８】

この種のサブワード・データ併合では、キャッシュ・コピー中の修正されたバイトは、対応する位置を有する発信側データ・ブロック・バイトが以前に修正されなかった場合にしか記憶されない。このため、併合が完了するまで、同じプロセッサが再び発信側データ・ブロックのコピーをキャッシュし、再び同じバイトを修正するのを妨げなければならない。併合が完了するまで発信側データ・ブロックを求める次の要求を中断することによって、前記発信側データ・ブロックのサブワード・データ併合が開始した後にプロセッサが再び前記発信側データ・ブロックをキャッシュするのを妨げることが好ましい。

【００４９】

図４も参照すると、大域メモリ制御装置３０は、データ要求の中断を行うために、各発信側データ・ブロック１１８ごとの中断フラグ１６０を大域メモリに記憶する。最初、中断

フラグ１６０は非中断値にセットされる。非中断値は、１ビット中断フラグを使用する本発明の実施例では「０」であることが好ましい。発信側データ・ブロック１１８のサブワード・データ併合が開始すると、この発信側データ・ブロックに関連する中断フラグ１６０が、好ましくは１である中断値にセットされる。データ要求が受け取られたときに中断フラグ１６０が中断値にセットされている場合、大域メモリ制御装置３０は中断待ち行列１６２に要求を記憶する。中断待ち行列１６２とは、大域メモリ３２に記憶されている中断要求１６４ないし１６８のリストである。たとえば、図４に示したように、記憶されている中断された要求１６４は、プロセッサ識別子１７０と、要求タイプ識別子１７２と、ブロック・アドレス１７４とを備えている。中断された要求１６４ないし１６８用の他のフォーマットも使用することができる。中断フラグ１６０が後で、サブワード・データ併合が完了した後に非中断値にリセットされると、大域メモリ制御装置３０は、好ましくは先入れ先出し順序（すなわち、受け取られた順序）で待ち行列に記憶されている中断された要求１６４ないし１６８を実行する。

【００５０】

併合がいつ完了したかを判定するには、発信側データ・ブロック１１８のコピーがいくつキャッシュされたかを追跡する必要がある。大域メモリ制御装置３０は、この数を追跡するために、大域メモリ３２中の各発信側データ・ブロック１１８ごとのカウント１７８を記憶する。最初、大域メモリ制御装置３０は各カウントを「０」にセットする。次いで、大域メモリ制御装置３０は、データ・ブロック１１８のコピーをプロセッサに伝送するたびに、カウント１７８を増分する。コピーが再び発信側データ・ブロック１１８と併合されるたびに、大域メモリ制御装置３０はブロックに関連するカウント１７８を減分する。また、キャッシュ・コピーが修正されずにプロセッサのローカル・メモリからフラッシュされた場合、プロセッサのローカル・メモリ制御装置は、大域メモリ制御装置がカウント１７８を減分できるように前記大域メモリ制御装置に通知する。このように、データ・ブロックの未処理キャッシュ・コピーの数のカウント１７８は正確に維持される。カウント１７８が減分されて再び「０」になると、大域メモリ制御装置３０は、併合が完了したと判定し、このデータ・ブロック用の中断されているデータ要求があればそれを実行する。

【００５１】

図１、図３、および図４を参照すると、本発明の第４の実施例では、プロセッサがデータ・ブロックに専用にアクセスできるようにする「ロック」として中断フラグ１６０、中断待ち行列１６２、およびカウント１７８も使用される。そのようなロックは、メモリ・アクセスの同期やその他の目的に有用である。発信側データ・ブロック１１８に専用にアクセスするために、ロック側プロセッサ（たとえば、プロセッサ２２）は、発信側データ・ブロック１１８が記憶された大域メモリ装置２６にロック要求を伝送する。発信側データ・ブロック１１８用のカウント１７８が「０」である場合、大域メモリ装置２６中の大域メモリ制御装置３０は、前記ブロック用の中断フラグ１６０（「ロック」フラグとしても働く）をロック値にセットし、データ・ブロックのコピーをキャッシュできるようにロック側プロセッサ２２に送り、カウントを「１」に増分する。図４に示したものなどの１ビット中断／ロック・フラグ１６０を有する実施例の場合、ロック値は「１」でよく、最初の非ロック値は「０」でよい。中断／ロック・フラグ１６０は、プロセッサ２２がロックを解除するまで、他のすべてのプロセッサが発信側データ・ブロック１１８のコピーをキャッシュするのを妨げる。中断／ロック・フラグ１６０がロック値にセットされている間の他のプロセッサ２３、２４による発信側データ・ブロック１１８を求めるデータ要求およびロック要求は、中断待ち行列１６２に入れられる。

【００５２】

しかし、ロック側プロセッサ２２がロック要求を送ったときに発信側データ・ブロック１１８用のカウント１７８が非ゼロであるときは、すでに他のプロセッサが前記データ・ブロックのコピーをキャッシュしている。したがって、ロック側プロセッサ２２は、すべての未処理キャッシュ・コピーが発信側データ・ブロック１１８と併合され、カウント１７８が０に戻るまで、前記発信側データ・ブロックへの専用アクセスを得ることはできない

。そのような例では、大域メモリ制御装置３０が中断／ロック・フラグ１６０をセットしてロック要求をまず中断待ち行列１６２に記憶することによってロック要求に応答することが好ましい。これによって、発信側データ・ブロック１１８のコピーがさらにキャッシュされるのが妨げられ、ロック要求を実行する上での遅延が最小限に抑えられる。発信側データ・ブロック１１８の未処理キャッシュ・コピーが併合された後、カウント１７８は「０」に戻り、ロック要求を実行することができる。

【００５３】

ロックの解除は、ロック側プロセッサ２２が大域メモリ装置２６に通知することによってロックを明示的に解除したときに行われる。大域メモリ制御装置３０は、これに応答して、中断／ロック・フラグ１６０を非ロック値にセットする。

【００５４】

発信側データ・ブロック１１８がロックされている間、他のプロセッサによる前記発信側データ・ブロックを求めるどんなデータ要求も中断され、すなわち、後で実行できるように中断待ち行列に記憶される。ロックが解除された直後に、大域メモリ制御装置によって要求が順序正しく実行される。ロックは、データ要求を中断することによって、低速メモリへのアクセスに対しても同様に機能する。プロセッサがメモリにデータ要求を送り、要求されたデータは、利用可能になった時点で返される。

【００５５】

データ・ブロックがロックされている間にデータ要求を中断することは、プログラマが他のタイプのロックと共に使用する「スピンロック」を不要にする利点も有する。スピンロックとは、ロックが解除され、プロセスが継続可能になるまで、連続的にロックの状況を試験する前記プロセス中のプログラムされた手順である。スピンロックは一般的に、ロックできる特定のデータへのアクセスなしには進行できないプロセスで使用される。しかし、ロックの状況のこの連続試験では、プロセスを実行しているプロセッサと大域メモリの間でかなりの通信トラフィックが生成され、プロセッサの中央演算処理装置も停止する。本発明のロックで中断待ち行列を使用してデータ要求を中断することによって、通信トラフィックおよび中央演算処理装置のオーバヘッドなしで、スピンロックと同じ機能が提供される。

【００５６】

さらに、この発信側データ・ロックでは、大域メモリ制御装置２６、２７がプロセッサのデータ要求を中断するときに前記プロセッサに通知することによって、中断されたデータ要求をもつプロセッサ２２ないし２４が、ロックが解除されるまで遊休状態になるのを妨げることができる。場合によっては、プロセスは、高い優先順位のタスクに必要とされるデータのロックが解除されるのを待つ間に実行できる何らかの低い優先順位のタスクを有することができる。プロセッサ上で実行中のプロセスは、前記プロセッサのデータ要求が中断されることを前記プロセッサに通知することによって、要求されたデータを必要としない他の有用な作業を実行することができる。

【００５７】

本発明のこの実施例によるロックの他の利点は、ロックの機能がすべて大域メモリ装置２６、２７によって実行されるので、プロセッサ２２ないし２４を修正する必要がないことである。

【００５８】

次に、図１および図５を参照すると、本発明の第５の実施例では、データ・レース条件を識別するために、各データ・ブロック１８４ごとの大域書込みビットマスク１８０および大域読取りビットマスク１８２が、前記データ・ブロックを含む大域メモリ３２（図１）に記憶される（図示のためだけに、本明細書では、プロセッサ２２によってキャッシュされているものとしてコピー２２４を記述し、大域メモリ装置２６によって記憶されているものとしてデータ・ブロック１９６を記述する。共用メモリ多重処理コンピュータ・システム２０の他のプロセッサ２３ないし２４および大域メモリ装置２７も同様に動作することが好ましいことを理解されたい）。大域書込みビットマスク１８０は、併合中にブロッ

ク１８４のどの要素１９０ないし１９４が修正されるかを追跡するために使用され、要素と１対１の関係で関連付けられた複数のフラグ２００ないし２０４から成る。大域読取りビットマーク１８２は、要素１９０ないし１９４のどれか読み取られる分を追跡するために使用され、やはり要素と１対１の関係で関連付けられた複数のフラグ２１０ないし２１４から成る。フラグ２００ないし２０４、２１０ないし２１４は最初、それぞれ非書込み値および非読取り値にセットされ、併合が完了したときにこれらの値にリセットされる。各フラグは単一ビットであり、０が非書込み値および非読取り値であることが好ましい。

【００５９】

本発明の第５の実施例でも、プロセッサ２２によってコピーがキャッシュされると、ローカル書込みビットマスク２２０およびローカル読取りビットマスク２２２がデータ・ブロック１８４のコピー２２４と共にプロセッサのローカル・メモリ４４に記憶される。ローカル・メモリ制御装置４０は、コピー２２４がローカル・メモリ４４にキャッシュされている間に、ローカル書込みビットマスク２２０によって、コピー２２４のどのデータ要素２３０ないし２３４が修正されているかを追跡する。ローカル書込みビットマスク２２０は、コピー２２４の各要素２３０ないし２３４ごとに１つあるフラグ２４０ないし２４４から成る。ローカル・メモリ制御装置４０は最初、要素２３０ないし２３４がまだ修正されていないことを示す非書込み値にフラグ２４０ないし２４４をセットする。ローカル読取りビットマスク２２２も、各要素２３０ないし２３４ごとに１つあり、最初は非読取り値にセットされる、フラグ２５０ないし２５４を有する。各フラグは単一ビットであり、「０」が非書込み値および非読取り値であることが好ましい。

【００６０】

中央演算処理装置３６がキャッシュ・コピー２２４の要素２３０ないし２３４に書込みを行うと、ローカル・メモリ制御装置４０は、書込みが行われた要素に関連するローカル書込みビットマスク２４０ないし２４４を書込み値にセットする。この場合も、書込み値が単一ビット・フラグの「１」であり、非書込み値が「０」であることが好ましい。同様に、中央演算処理装置によってキャッシュ・コピー２２４の要素２３０ないし２３４が読み取られると、ローカル・メモリ制御装置４０は、読み取られたそれらの要素に関連するローカル読取りビットマスク・フラグ２５０ないし２５４を読取り値（やはり、単一ビット・フラグの「１」であることが好ましい）にセットする。したがって、ローカル書込みビットマスクおよびローカル読取りビットマスク２２０、２２２はそれぞれ、キャッシュ・コピー２２４のどの要素に対して書込みおよび読取りが行われたかを示す。

【００６１】

キャッシュ・コピー２２４がローカル・メモリ４４からフラッシュされた後、ローカル・メモリ制御装置４０は、ローカル・ビットマスク２２０、２２２およびキャッシュ・コピー２２４をデータ・ブロック１９６と併合できるように大域メモリ装置２６に送る。大域メモリ制御装置３０は、ローカル書込みビットマスク２２０を使用して、図２に関して上記で説明したように、キャッシュ・コピー２２４とデータ・ブロック１９６を併合する。

【００６２】

ローカル書込みビットマスク２２０およびローカル読取りビットマスク２２２は、データ・レースが発生したかどうかを判定するためにも大域メモリ装置２６で使用される。上述のように、書込み／書込みデータ・レースは、複数のプロセッサが同期の間に、データ・ブロックの同じデータ要素に書込みを行ったときに発生する。書込み／書込みデータ・レースが発生したかどうかを判定するために、ローカル書込みビットマスク２２０と大域書込みビットマスク１８０が比較され、ビットマスクの対応する位置にあるフラグが共に書込み値にセットされているかどうかが判定される。大域書込みビットマスク・フラグ２００ないし２０４は、他のキャッシュ・コピーをデータ・ブロック１９６と併合することによってデータ・ブロック１９６のどの要素１９０ないし１９４がすでに修正されているかを示す。ローカル書込みビットマスク・フラグ２４０ないし２４４は、コピー２２４のどの要素２３０ないし２３４が修正されたかを示す。したがって、大域書込みビットマスク１８０、ローカル書込みビットマスク２２０の対応する位置にあるフラグが共に書込み値

にセットされている場合、現在併合中のキャッシュ・コピーと以前に併合されたコピーのデータ・ブロックとで同じデータ要素が修正されている。

【００６３】

大域書込みビットマスク１８０、ローカル書込みビットマスク２２０の対応する位置にあるフラグが共に書込み値にセットされているとの判定は、フラグが単一ビットであり、書込み値が１であり、非書込み値が「０」である場合、迅速に行うことができる。これは、大域書込みビットマスク１８０、ローカル書込みビットマスク２２０のビット単位論理積演算を実行することによって行われる。論理積演算の結果におけるビットのどれかが非ゼロである場合、キャッシュ・コピーとデータ・ブロックの両方で同じデータ要素が修正されている。さらに、論理積演算結果におけるそのような非ゼロ・ビットの位置は、キャッシュ・コピーとデータ・ブロックの両方でどのデータ要素が修正されたを示す。この情報は、書込み／書込みデータ・レースを発生させるプロセッサ２２上で動作するプロセスでバグを見つける上で極めて有用である。

【００６４】

さらに、上述のように、読取り／書込みデータ・レースは、１個のプロセッサが同期の間に、他のプロセッサによって修正されたデータ要素を読み取るために発生する。読取り／書込みデータ・レースが発生したかどうかを判定するために、ローカル読取りビットマスク２２２と大域書込みビットマスク１８０が比較される。ローカル書込みビットマスク２２０と大域読取りビットマスク１８２も比較される。ローカル読取りビットマスク２２２と大域書込みビットマスク１８０の対応する位置にあるフラグがそれぞれ、読取り値および書込み値にセットされている場合、同じ要素が、現在併合中のキャッシュ・コピー２２４では読み取られ、以前に併合されたコピーのデータ・ブロック１９６では修正されている。また、ローカル書込みビットマスク２２０および大域読取りビットマスク１８２の対応する位置にあるフラグがそれぞれ、書込み値および読取り値にセットされている場合、同じデータ要素が、現在併合中のキャッシュ・コピー２２４では修正され、データ・ブロック１９６の以前に併合されたコピーでは読み取られている。ローカル書込みビットマスク２２０と大域書込みビットマスク１８０の比較のように、これらの判定はビットマスクに対するビット単位論理積演算を使用して行うことが好ましい。

【００６５】

キャッシュ・コピー２２４がデータ・ブロック１９６と併合された後、ローカル書込みビットマスク２２０が大域書込みビットマスク１８０と併合される。大域メモリ制御装置３０は、書込み値にセットされたキャッシュ・コピー書込みビットマスク２２０中の各フラグ２４０ないし２４４ごとに、対応する位置にある大域書込みビットマスク・フラグ２００ないし２０４を書込み値にセットする。また、すでに書込み値にセットされている大域書込みビットマスク・フラグ２００ないし２０４は書込み値にセットされたままである。書込みビットマスクの併合は、書込みビットマスク・フラグ２００ないし２０４、２４０ないし２４４が単一ビット値であり、書込み値が「１」であり、非書込み値が「０」であるビット単位論理積演算によって迅速に実行することができる。大域メモリ制御装置は、大域書込みビットマスク１８０、ローカル書込みビットマスク２２０に対してビット単位論理和演算を実行して、大域書込みビットマスク１８０に結果を記憶する。

【００６６】

ローカル読取りビットマスク２２２も同様に、大域読取りビットマスク１８２と併合される。大域メモリ制御装置３０は、ローカル読取りビットマスク２２２中の各フラグ２５０ないし２５４ごとに、対応する位置にある大域読取りビットマスク・フラグ２１０ないし２１４を読取り値にセットする。すでに読取り値にセットされている大域読取りビットマスク・フラグ２１０ないし２１４は読取り値にセットされたままである。各読取りビットマスク・フラグ２１０ないし２１４が単一ビット・フラグであり、読取り値が１であり、非読取り値が０であり、ビット単位論理和演算によって大域読取りビットマスク１８２、ローカル読取りビットマスク２２２を併合できるようにすることが好ましい。

【００６７】

キャッシュ・コピー２２９とデータ・ブロック１９６が、図２に関して上記で説明したように併合されると、ローカル書込みビットマスク２２０、ローカル読取りビットマスク２２２、大域書込みビットマスク１８０、大域読取りビットマスク１８２の対応する位置にあるフラグは共に、単一のプロセッサが、複数回キャッシュされたデータ・ブロック１９６のキャッシュ・コピー中の関連するデータを読み取り、あるいは修正した結果として、読取り値または書込み値にセットすることができる。したがって、ビットマスクは、データ・レースを発生させる複数のプロセッサによるデータ要素１９０ないし１９４の読取り／修正と、データ・レースを発生させない再キャッシュ・コピーでの単一のプロセッサによる読取り／修正を共に識別する。ビットマスクが実際のデータ・レースしか識別しないように、上述の中断フラグ１６０（図４）および中断待ち行列１６２を使用して、プロセッサ２２ないし２４が、併合が開始した後にデータ・ブロック１９６の次のコピーをキャッシュするのを妨げることもできる。この代替法では、併合が開始した後にデータ・ブロックを求める次の要求を中断することによって、データ・ブロック１９６の複数の再キャッシュ・コピーにおいて単一のプロセッサがデータ要素を修正し、あるいは読み取って修正するのを妨げる。しかし、要求を中断すると、ブロックの未処理コピーがすべて併合されるまでこれらのプロセッサに対して任意の遅延が課される。

【００６８】

再び図１を参照すると、本発明の第６の実施例では、大域メモリ装置である種の「データ・リダクション」演算を実行することによって、プロセッサ２２ないし２４と大域メモリ装置２６、２７の間の通信トラフィックが低減される（本明細書では、そのような動作を大域リダクション演算と呼ぶ）。大域リダクション演算はいくつかの形をとることができ、データ値とロック値の両方に対して実行することができる。共用メモリ多重処理コンピュータ・システム２０上で実行される多数のプロセスでは、後でデータ・ブロック中の特定の位置に記憶される単一の値が得られる１組のデータに対して第１の形の大域リダクション演算を実行することが望ましい。たとえば、プロセスは、データ・ブロックの要素中の値のアレイの最大値または最小値を記憶する必要があることがある。この種の他の共通大域リダクション演算には、１組のデータ要素の和または積を記憶することや、データ・ブロックの特定の位置にあるセット中のすべての要素に対して実行された論理積演算または論理和演算の結果を記憶することが含まれる。

【００６９】

本発明の第６の実施例によれば、プロセッサ２２ないし２４によるデータ・リダクション要求に応答して大域メモリ制御装置３０、３１によってある種のデータ・リダクション演算が実行される。たとえば、プロセッサ２２は、１組のデータ要素の最大値を大域メモリ３２中の特定の位置に記憶するデータ・リダクション演算を実行するために、「reduce to maximum （最大値まで減少する）」演算、対象データ要素セット、および結果を記憶すべき位置を指定した要求を大域メモリ制御装置３０に送る（指定された１組のデータ要素は大域メモリ３２中のどこにでも記憶することができる）。大域メモリ制御装置３０は、これに応答して、要求された演算を実行する。大域メモリ制御装置３０は、指定されたセット中の最大値を見つけて、指定された位置にその値を記憶する。

【００７０】

第２の形の大域リダクション演算では、大域リダクション演算を要求するプロセッサ（たとえば、プロセッサ２２）が、オペランドを提供し、第２のオペランドの大域メモリ位置を指定する。要求側するプロセッサ２２は、これらのオペランドに対して実行すべき演算も指定する。この場合も、演算自体は、大域メモリ制御装置３０によって、指定されたメモリ位置に対して実行される。大域メモリ制御装置３０は、オペランドに対して演算を実行し、第２のオペランドの代わりに大域メモリ３２に結果を記憶し、あるいはプロセッサ２２によって指定された大域メモリ３２中の他の位置に結果を記憶する。

【００７１】

大域メモリ装置２６、２７でデータ・リダクション演算を実行すると、プロセッサ２２ないし２４と大域メモリ装置の間のトラフィックが低減される。データ・リダクション演算

がプロセッサ２２ないし２４自体によって実行されるとき、かなりのデータを交換しなければならない。データ・リダクション演算を実行するプロセッサは、対象データ要素セットを求めるデータ要求を大域メモリ装置に送らなければならない。大域メモリ装置は、これに応答して、対象セットを含むデータ・ブロックをキャッシュできるように要求側プロセッサに送る。前記プロセッサは、キャッシュ・コピーに対してデータ・リダクション演算を実行し、ターゲット位置（他のデータ・ブロック中にあってよく、そのような他のブロックのキャッシュを要求することができる）に結果を書き込む。プロセッサは次いで、ターゲット位置と関連する書込みビットマスクとを含む現在修正されているコピーを、データ・ブロックと併合できるように大域メモリ装置に送り返す。さらに、これらの様々なデータ・ブロック・アクセスはそれぞれ、一般に、そのデータ・ブロックに関連するロックにアクセスし（それによって、併合プロセスが完了するのを待つことが必要になることもある）、その後ロックを解除することを必要とする。

【００７２】

これに対して、大域メモリ装置２６、２７でデータ・リダクション演算が実行されるとき、要求側プロセッサから演算を実行する大域メモリ装置に送られるのは、データ・リダクション要求で使用するためのデータ・リダクション・データ値だけ（および場合によっては、オペランド・データ値）である。プロセッサが様々なデータ・ブロックに対するロックにアクセスするオーバヘッドもなくなる。

【００７３】

再び図１を参照すると、共用メモリ処理コンピュータ・システム２０中の多数のプロセッサ２２ないし２４が、１つの大域メモリ装置２６、２７に記憶された単一のデータ・ブロックへのアクセスを必要とすることがある（３個のプロセッサ２２ないし２４および２個の大域メモリ装置２６および２７が、共用メモリ処理コンピュータ・システム２０に含めることができる多数のプロセッサおよび大域メモリ装置を示すことが認識されよう）。そのような状況では、データ・ブロックを記憶する大域メモリ装置は多数のデータ要求を実行し、かつ後でデータのキャッシュ・コピーを併合しなければならない。したがって、大域メモリ装置がこのような要求実行動作および併合動作を実行できる速度が不十分なものになる恐れがある。その結果、ある種の要求実行動作および併合動作が遅延され、いわゆる「ボトルネック」が発生する。同時に、他の大域メモリ装置が遊休状態になる恐れがある。

【００７４】

本発明の第７の実施例では、多数のプロセッサが１個のプロセッサに記憶された単一のデータ・ブロックにアクセスすることによるボトルネックが、いくつかの大域メモリ装置間でそのようなデータ・ブロックを階層的に共用して併合することによってなくなる。たとえば、大域メモリ装置２６は、同じデータ・ブロックを求める多数のデータ・ブロック要求を受け取るようになると（たとえば、データ・ブロック要求の数が何らかの所定の限界を超えたとき）、他の大域メモリ装置２７にデータ・ブロックのコピーを送る。大域メモリ制御装置３１は、大域メモリ装置２７でコピーが受け取られると、大域メモリ３３にそのコピー（以下「大域コピー」と呼ぶ）を記憶する。

【００７５】

図１および図６を参照すると、ブロックを求める次のデータ・ブロック要求の実行と、ブロックのキャッシュ・コピーの併合は次いで、大域メモリ装置２６、２７間で分割される。この分割は、プロセッサと大域メモリ装置の所定の階層順序付けに従って行うことが好ましい。プロセッサ２２ないし２４と大域メモリ装置２６および２７のそのような階層順序付け２７０の１つを図６に示す。階層順序付けが、大域メモリ装置に対するプロセッサの論理順序付けまたは割当てであり、図１に示した共用メモリ処理コンピュータ・システムの構造に対する変化を反映しないことが認識されよう。また、本発明のこの実施例による階層順序付けが、図６に示したものよりもずっと多くのプロセッサ、大域メモリ装置、および階層レベルを含むことができることを理解されたい。具体的には、共用メモリ処理コンピュータ・システム２０は追加大域メモリ装置を含むことができ、発信側データ・ブ

ロックの大域コピーはこれらの大域メモリ装置のうちの複数に記憶することができる。また、多くのプロセッサは通常、発信側データ・ブロックの大域コピーを記憶する各大域メモリ装置に割り当てられる。

【００７６】

所定の階層順序２７０で、あるプロセッサは大域メモリ装置２６に割り当てられ、他のプロセッサは大域メモリ装置２７に割り当てられる。大域メモリ装置２６に割り当てられたプロセッサ２２によって送られるデータ・ブロック要求は、その装置によって実行される。また、大域メモリ装置２６に割り当てられたプロセッサ２２によってキャッシュされたブロックのコピーは、図２に関して上記で説明したように発信側データ・ブロックと併合される。しかし、大域メモリ装置２７に割り当てられたプロセッサ２３、２４に関するデータ・ブロック要求の実行とキャッシュ・コピーの併合は、装置２７によって実行される。

【００７７】

大域メモリ装置２７において、プロセッサ２３、２４にキャッシュされたコピーが大域メモリ３３に記憶された大域コピーと併合される。図２に関して上記で説明した併合と同様に、ローカル書込みビットマスク中の関連するフラグが書込み値にセットされたキャッシュ・コピーの要素を大域コピーに記憶することによって、各キャッシュ・コピーが大域コピーと併合される。また、大域メモリ装置２６に記憶された発信側データ・ブロックと大域コピーを後で併合できるように、大域書込みビットマスクは大域メモリ装置２７に記憶される。この大域書込みビットマスクは、大域コピーの要素と１対１の関係で関連付けられたフラグを有する。これらのフラグは最初、非書込み値にセットされる。次いで、キャッシュ・コピーを併合している間に大域コピーの要素が修正されると、そのような要素に関連する大域書込みビットマスク・フラグが書込み値にセットされる。

【００７８】

大域メモリ装置２７に割り当てられたプロセッサから得たすべてのキャッシュ・コピーが大域コピーと併合された後、大域コピーは発信側データ・ブロックと併合される。大域コピーを発信側データ・ブロックと併合するには、関連する大域書込みビットマスク・フラグが書込み値にセットされた大域コピーの要素が、発信側データ・ブロックの対応する位置に記憶される。

【００７９】

図７を参照すると、本発明の第８の実施例では、データ・ブロック２８０に関連する特殊併合フラグ２８２および特殊併合テーブル２８４を使用する代替併合方式によって、併合が開始した後に、大域メモリ３２（図１）に記憶されたデータ・ブロック２８０を求めるプロセッサ２２ないし２４（図１）による要求を中断する必要がなくなる。これに対して、上述の本発明の第１の実施例では、ブロックのキャッシュ・コピーに関連するローカル書込みビットマスク５６（図２）を使用する併合方式によって、部分的に併合されたデータ・ブロックを求める要求を中断する必要がなくなる。

【００８０】

第８の実施例の併合方式は、特殊併合フラグ２８２の状態によって制御される「通常」モードと「特殊」モードの２つの併合モードを有する。特殊併合フラグ２８２は、通常併合モードを示す第１の状態と特殊併合モードを示す第２の状態の２つの状態を有する。最初、特殊併合フラグ２８２は第１の状態にセットされ、通常併合モードが使用される。併合が開始して、現在部分的に併合されているデータ・ブロック２８０のコピーをキャッシュする要求が出された後、特殊併合モードが使用され、特殊併合フラグ２８２が第２の状態にセットされる。

【００８１】

最初の通常併合モードでは、データ・ブロック２８０のコピーをプロセッサ２２ないし２４によって要求し、それらのプロセッサのローカル・メモリ４４ないし４６（図１）に記憶またはキャッシュすることができる。キャッシュ・コピーは、ローカル・メモリ４４ないし４６からフラッシュされると、データ・ブロックに関連する大域書込みビットマスク

２８８の助けによって、大域メモリ装置２６（図１）中のデータ・ブロック２８０と併合される。大域書込みビットマスク２８８は、最後に併合が完了してから修正されていないデータ・ブロック２８０の要素２９６ないし２９９はどれかを示すためのフラグ２９０ないし２９３を含む。最初、ビットマスク・フラグ２９０ないし２９３はすべて、未修正値にセットされる。各キャッシュ・コピーは、データ・ブロックの対応する位置にある未修正データ要素２９６ないし２９９と異なるキャッシュ・コピーのデータ要素を記憶することによって併合される。そのように修正されたデータ・ブロック要素２９６ないし２９９用のビットマスク・フラグ２９０ないし２９３は修正値にセットされる。

【００８２】

カウンタ３０４は、未処理のデータ・ブロック２８０のキャッシュ・コピーの数を追跡する。カウンタ３０４は、プロセッサ２２ないし２４によってデータ・ブロック２８０のコピーがキャッシュされるたびに増分され、データ・ブロック２８０のキャッシュ・コピーがローカル・メモリ４４ないし４６からフラッシュされるたびに減分される。併合は、カウンタ３０４が０に戻ったときに完了する。その時点で、すべての大域書込みビットマスク・フラグ２９０ないし２９３が未修正値にリセットされる。

【００８３】

通常併合モードでの併合が開始した（すなわち、少なくとも１つのキャッシュ・コピーがデータ・ブロック２８０と併合され、少なくとも１つのデータ要素２９６ないし２９９が修正された）後、続いてプロセッサ２２ないし２４によってデータ・ブロック２８０のコピーが要求されると、特殊併合モードに入る（従来型のKarp-Sarkar 併合方法では、そのような次の要求は中断される）。併合が開始したかどうかを判定するには、大域書込みビットマスク２８８を試験することができる。ビットマスク・フラグ２９０ないし２９３のどれかが修正値にセットされている場合、併合が開始している。したがって、大域メモリ制御装置３０（図１）がデータ・ブロック２８０のコピーを求める要求を受け取り、大域書込みビットマスクが非ゼロであるとき（ビットマスク・フラグの修正値が１ビットである場合）、大域メモリ制御装置３０は特殊併合フラグを第２の状態にセットして特殊併合モードに入る。

【００８４】

特殊併合モードのとき、大域メモリ制御装置３０は、要求側プロセッサにコピーを送ることによって、１個のプロセッサ２２ないし２４からのデータ・ブロック２８０のコピーを求める要求（特殊併合モードを起動させた要求を含む）に応答する（特殊併合モードのときに得られるそのようなコピーは、データ・ブロック２８０の併合が開始した後に得られるので、「部分的に併合された」コピーである）。また、大域メモリ制御装置３０は、特殊併合テーブル２８４を使用して、部分的に併合されたコピーを追跡する。大域メモリ制御装置は、プロセッサ識別子３１０（要求側プロセッサを制御するためのもの）およびブロック・アドレス３１２（複製コピーのコピー元のデータ・ブロックを識別するためのもの）と共に、部分的に併合されたコピー３０８（要求側プロセッサに送られるものと同じもの）の複製を特殊併合テーブル２８４の未占有スロット３０９に記憶する。複製コピー３０８と、プロセッサ識別子３１０と、ブロック・アドレス３１２とは、特殊併合テーブル中の「項目」を構成する。ブロック・アドレス３１２によって、特殊併合テーブル２８４を複数の大域メモリ・データ・ブロックに使用することができる。各データ・ブロックごとに１つある特殊併合テーブルは、大域メモリ３２（図１）の未使用部分の必要に応じて割り振ることもできる。そのような代替特殊併合テーブルでは、ブロック・アドレス３１２を記憶する必要がなくなる。

【００８５】

大域メモリ制御装置は、特殊併合モード中にキャッシュ・コピーをデータ・ブロック２８０と併合するとき、コピーをキャッシュしたプロセッサのプロセッサ識別子３１０と、（キャッシュされたコピーがそこから発する）データ・ブロック２８０用のブロック・アドレス３１２とを含む項目を特殊併合テーブル２８４で探索し、キャッシュ・コピーが部分的に併合されたコピーであるかどうかを判定する。特殊併合テーブルにそのような項目が

含まれていない場合（キャッシュ・コピーが、部分的に併合されたコピーではないことを示す）、キャッシュ・コピーとデータ・ブロックは、大域ビットマスク２８８を使用して通常併合モードと同様に併合される。しかし、そのような一致する項目（たとえば、項目３０９の占有スロット）が存在する場合（キャッシュ・コピーが、部分的に併合されたコピーであることを示す）、キャッシュ・コピーは更新ビットマスク３１６を使用して併合される。

【００８６】

大域メモリ制御装置３０は、部分的に併合されたコピーを併合する際、部分的に併合されたコピー（図示せず）のデータ要素に関連するビットマスク・フラグ３２０ないし３２３を含む更新ビットマスク３１６を生成する。大域メモリ制御装置３０は、一致する項目３０９中の複製コピー３０８の対応する位置にあるデータ要素と異なる、部分的に併合されたコピーのデータ要素に関連するビットマスク・フラグ３２０ないし３２３を、修正値にセットする。この更新ビットマスク３１６は併合を制御する。大域メモリ制御装置３０は、関連するビットマスク３２０ないし３２３が修正値にセットされている部分的に併合されたコピー・データ要素を大域データ・ブロック２８０に記憶する。次いで、一致する待ち行列項目３０９がクリアされる。

【００８７】

大域メモリ制御装置３０は、部分的に併合されたコピーを大域データ・ブロック２８０と併合した後、更新ビットマスク３１６も大域ビットマスク２８８と併合する。大域メモリ制御装置３０は、修正値にセットされた更新ビットマスク３１６の各ビットマスク・フラグ３２０ないし３２３ごとに、大域ビットマスク２８８の対応する位置にあるビットマスク・フラグ２９０ないし２９３を修正値にセットする。修正値が単一の１ビットであり、未修正値が単一の「０」ビットである場合、大域ビットマスク２８８と更新ビットマスク３１６のビット単位論理和の結果を大域ビットマスク２８８に記憶することによって併合を行うことができる。

【００８８】

上述の本発明の第８の実施例の代替併合方式は、サブワード併合が開始した後に発生するプロセッサ要求を中断する必要をなくすために上記の第３の実施例のサブワード併合方法に適用することもできる。サブワード記憶併合方法が代替併合方法で補足される本発明のそのような実施例では、サブワード記憶動作によって修正されたキャッシュ・コピーを大域メモリ・データ・ブロックと併合した後に発生するどんな要求も中断されない。その代わり、大域メモリ制御装置３０（図１）は、部分的に併合されたデータ・ブロックの要求されたコピーを要求側プロセッサに送り、特殊併合テーブルに対応する項目を作成し、特殊併合フラグをセットする。その後、一致するテーブル項目を有するキャッシュ・コピーは更新ビットマスクと併合され、第８の実施例と同様に、一致する項目のないキャッシュ・コピーは大域ビットマスクと併合される。特殊併合フラグがセットされる前に、キャッシュ・コピーは第３の実施例と同様に併合される。

【００８９】

図１および図３を参照すると、本発明の第９の実施例は大域メモリ装置２６、２７（図１）でのデータ併合をビット・レベルで行う。この機能では、２個のプロセッサが同期の間に同じビットを修正しない限り、複数のプロセッサ２２ないし２４（図１）は共用データ・ブロックの同じバイト中のビットを修正することができる。各プロセッサのキャッシュ・コピー中の修正された別々のビットは後で、発信側大域メモリ・データ・ブロックにビット・レベルで併合される。複数のプロセッサ間でデータを共用することによって発生するデータ・レースは、ビット・レベルで検出することができる。

【００９０】

図３を参照すると、（第３の実施例と同様に）本発明の第９の実施例では、サブワード記憶フラグ１１０およびキャッシュ・コピー書込みビットマスク８８が、プロセッサのローカル・メモリ４４ないし４６に記憶された各キャッシュ・コピー８６に関連付けられている。第９の実施例のサブワード記憶フラグ１１０は、キャッシュ・コピー８６中の最小サ

イズの修正されたデータ要素を示すための２ビット値から成ることが好ましい。２ビット値では、４つの最小サイズを示すことができる。たとえば、値「１１」を使用して、キャッシュ・コピーの個別のビットが修正されたことを示すことができる。値「１０」、「０１」、「００」を使用して、バイト、ハーフワード（一般に１６ビット・データ要素）、またはワードがそれぞれ、最小サイズの修正されたデータ要素であったことを示すことができる。

【００９１】

第９の実施例で、ワード、ハーフワード、またはバイトが、最小サイズの修正されたデータ要素であるとき、キャッシュ・コピー８６は第３の実施例と同様に、キャッシュ・コピー自体の発信側データ・ブロックと併合される。具体的には、ワードが、最小サイズの修正されたデータ要素であるとき、キャッシュ・コピーはキャッシュ・コピー書込みビットマスク８８を使用して併合される。最小サイズ修正データ要素がハーフワードまたはバイトであるとき、キャッシュ・コピーは、それぞれ、発信側データ・ブロック１１８（図４）の各ハーフワードまたはバイトごとにビットマスク・フラグがあるサブワード書込みビットマスク１２４（図４）を使用して併合される。ビット・レベルで修正されたキャッシュ・コピーは、以下で説明するように併合される。発信側データ・ブロック１１８がハーフワード・レベル、バイト・レベル、またはビット・レベルで併合された後、下位レベルの併合が必要になり、あるいは併合が完了するまで、データ・ブロックのどんな残りのキャッシュ・コピーでもその同じレベルで併合される。

【００９２】

一般に、大部分のキャッシュ・コピーでは、最小サイズ修正要素はワードである。そのようなキャッシュ・コピーに対する併合プロセスは最も高速であり、処理作業が少なくて済み、ビットマスクが小さくて済む。バイトまたはハーフワードの最小サイズ修正要素を有するキャッシュ・コピーはそれほど頻繁には発生しない。このようなキャッシュ・コピーの併合には、より多くの処理作業および処理時間が必要とされる。ビット・レベルで修正されたキャッシュ・コピーの併合はこれよりもずっとまれであり、さらに多くの処理時間および処理作業を必要とする。したがって、最小サイズ修正データ要素のレベルで併合することの利点は、所与のキャッシュ・コピーのための処理時間および処理作業が最小限に抑えられ、同時に、必要に応じてビット・レベルまでのプロセッサ間のデータ共用が可能になることである。

【００９３】

図１、図３、および図８を参照すると、ビット・レベルでの併合の場合、キャッシュ・コピー８６のビット・レベル修正がいつ行われたかを検出することが必要である。従来型のＣＰＵ３６ないし３８では一般に、ローカル・メモリ４４ないし４６に記憶されたキャッシュ・コピーを直接修正するのは、ＣＰＵによって実行される（ワードまたはバイト・サイズの増分単位の）記憶動作だけである。したがって、そのようなキャッシュ・コピー中のワードまたはバイトの修正は、ＣＰＵ３６ないし３８が、ワード、ハーフワード、またはバイトをＣＰＵの内部レジスタからローカル・メモリ４４ないし４６に移動するための記憶動作をいつ実行するかから検出することができる。ＣＰＵ３６ないし３８とローカル・メモリ４４ないし４６の間でデータを移動するのはロード動作および記憶動作だけなので、キャッシュ・コピーの個別のビットは間接的に修正される。ＣＰＵ３６ないし３８は、個別のビットを修正するために、バイト・データまたはワード・データを内部レジスタにロードし、１つまたは複数のビット・レベル演算をデータに対して実行し、次いで、ローカル・メモリ４４ないし４６中のキャッシュ・コピーにそのワードまたはバイトを記憶する。

【００９４】

第９の実施例におけるＣＰＵ３６ないし３８は、ビット・レベル修正を検出するために、ＣＰＵ自体の各内部レジスタごとにビット修正フラグを維持する。たとえば、図８に示したように、ＣＰＵ３６は、その機能単位３３８によって処理しているデータを一時的に記憶するための８つの内部レジスタ３３０ないし３３７を有することができる。一般に、機

能単位３３８は、整数浮動小数点処理回路と命令復号回路とを含む。ＣＰＵ３６は、各内部レジスタ３３０ないし３３７ごとに１つある１ビットのビット修正フラグ３４０ないし３４７も有する。ＣＰＵ３６は、プロセッサ・バス３５０およびバス・インタフェース３５２を介して内部レジスタ３３０ないし３３７とローカル・メモリ４４（図１）の間でデータを転送する。

【００９５】

ビット修正フラグ３４０ないし３４７は、それらに関連する内部レジスタ３３０ないし３３７中のデータがビット・レベルで修正されたかどうかを示す。最初、ＣＰＵが内部レジスタ３３０ないし３３７のどれか１つにデータをロードすると、そのレジスタ用のビット修正フラグ３４０ないし３４７は未修正値（たとえば、０ビット）にセットされる。ＣＰＵ３６が、レジスタ中の１ビットまたは複数ビットのデータを修正するが、バイトやワード全体は修正しないことを求める命令をＣＰＵ自体の機能単位３３８によって実行するとき、機能単位３３８はビット修正フラグを修正値（たとえば、１ビット）にセットする。次いで、ＣＰＵ３６が１つの内部レジスタ３３０ないし３３７からローカル・メモリ４４中のキャッシュ・コピー８６にデータを記憶するとき、ローカル・メモリ制御装置４０は、そのレジスタに関連するビット修正フラグを検査して、キャッシュ・コピーをビット・レベルで併合すべきであるかどうかを判定し、それに応じてキャッシュ・コピー用のサブワード記憶フラグ１１０をセットする。

【００９６】

図４を参照すると、第９の実施例では、サブワード記憶フラグ１１０が、最小サイズ修正要素がビットであることを示す「１１」にセットされると、キャッシュ・コピー８６がその発信側データ・ブロック１１８とビット・レベルで併合される。ビット・レベル併合の場合、サブワード書込みビットマスク１２４のビットマスク・フラグ１２６ないし１３７は発信側データ・ブロック１１８のビットと１対１の関係で関連付けられている（図４は、ビットマスク・フラグ１２６ないし１３７が、バイト・レベル併合のためにデータ・ブロックのバイト１４６ないし１５７と１対１の関係で関連付けられている場合を示す）。各ビットマスク・フラグ１２６ないし１３７が単一ビット値であり、１ビットが、（前の併合と同様に）データ・ブロック１１８の関連するビットが修正されたことを示し、０ビットが修正なしを示すことが好ましい。このサブワード書込みビットマスク１２４を使用して、発信側データ・ブロック１１８の未修正ビットに対応する位置にあるキャッシュ・コピー８６の修正されたビットが前記データ・ブロックに記憶され、ビット・レベル併合が行われる。

【００９７】

好ましいビット・レベル併合では、発信側データ・ブロック１１８用の大域メモリ制御装置３０（図１）（データ・ブロック１１８は大域メモリ３２に記憶されていると仮定する）は、発信側データ・ブロック１１８の各ビットごとの単一ビット・フラグを含む一時ビットマスク（図示せず）をキャッシュ・コピー８６と発信側データ・ブロック１１８のビット単位排他的論理和から作成する。これによって、キャッシュ・コピー８６中の対応する位置にあるビットと異なる発信側データ・ブロック１１８のビットに関連する一時ビットマスク・フラグが「１」の値にセットされる。次に、大域メモリ制御装置３０は、以前に修正されなかった発信側データ・ブロック１１８の各ビットごとに、そのビットの代わりに、キャッシュ・コピー８６の対応する位置にあるビットを記憶する。これは、以下の（１）式のに示したビット単位論理積演算および論理和演算によって行うことができる。

【００９８】

【数１】

$$B = (B \ \mathrm{AND} \ \mathrm{NOT} \ M) \ \mathrm{OR} \ (C \ \mathrm{AND} \ M) \quad \cdots\cdots (1)$$

【００９９】
上記（１）式で、Ｂは発信側データ・ブロック１１８を、Ｍはサブワード書込み記憶ビットマスク１２４、Ｃはキャッシュ・コピー８６を、ＡＮＤはビット単位論理積演算を、ＯＲはビット単位論理和演算を、ＮＯＴは論理補演算を示す。その場合、大域メモリ制御装置３０は、一時ビットマスクによってサブワード書込みビットマスク１２４のビット単位論理積演算を実行してサブワード記憶ビットマスクを更新する。
【０１００】
ハーフワード・レベル、バイト・レベル、およびビット・レベルの併合が、データ・ブロックのどのデータ要素が未修正のままであるかに応じて制御されるので、単一のプロセッサが同期の間にデータ・ブロックを複数回キャッシュし修正する可能性に対処する必要がある。第３の実施例の中断方法または第８の実施例の代替併合方法をこの目的に使用することができる。
【０１０１】
本発明の原則を好ましい実施例に関して説明したが、そのような原則から逸脱することなく本発明の構成および細部を修正できることが認識されよう。本発明の原則を適用できる多数の可能な実施例にかんがみ、詳細な実施例が例示のためだけのものであり、本発明の範囲を限定するものとみなされるものではないことを認識されたい。本出願人はむしろ、特許請求の範囲およびその等価物の範囲および趣旨内に含まれるそのようなすべての実施例を本発明として請求する。
【０１０２】
以上本発明の各実施例について詳述したがここで各実施例の理解を容易にするために、各実施例を要約して以下に列挙する。
【０１０３】
１．　複数のデータ・ブロック（１１８、１９６、２８０）を大域メモリ（３２）に記憶するステップと、
各プロセッサ（２２ないし２４）の要求に応じて、データ・ブロックのうちの発信側データ・ブロックのコピー（５４、８６、２２４）を要求側プロセッサがアクセスできるように前記要求側プロセッサに対してローカルなキャッシュ・メモリ（４４ないし４６）に記憶するステップと、
コピーのどの要素（７０ないし７４、８０ないし８２、２３０ないし２３４）が、その要求側プロセッサによって修正されるかを追跡するステップと、
修正されたコピーの要素をデータ・ブロックに記憶することによってコピーを併合するステップとを含むことを特徴とするコンピュータ・システムのメモリ共用方法である。
【０１０４】
２．　追跡ステップが、
それぞれ、最初は第１の値にセットされる、コピー（５４、８６、２２４）中の各データ要素（７０ないし７４、８０ないし８２、２３０ないし２３４）に関連するフラグ（６０ないし６４、９０ないし９２、２４０ないし２４４）を備えるビットマスク（５６、８８、２２０）をキャッシュ・メモリ（４４ないし４６）に記憶するステップと、
要求側プロセッサによって書込みを行われた各データ要素に関連するビットマスク中のフラグを第２の値にセットするステップとを含み、
コピーと発信側データ・ブロックを併合するステップが、関連するフラグが第２の値にセットされているコピーの各データ要素を発信側データ・ブロック（１１８、１９６、２８

０）に記憶するステップを含むことを特徴とする前記１に記載のコンピュータ・システムのメモリ共用方法である。
【０１０５】
３．　複数のデータ・ブロック（１１８、１９６、２８０）を備える共用データを大域メモリ（１３２、３３）に記憶するステップと、
１つのデータ・ブロック中のデータにアクセスする各プロセッサ（２２ないし２４）ごとに、
前記データ・ブロックのローカル・コピー（５４、８６、２２４）をプロセッサのローカル・メモリに記憶するステップと、
それぞれ、最初は非書込み値にセットされる、ローカル・メモリ中の複数のローカル・コピー書込みフラグ（６０～６４、９０～９２、２４０～２４４）を、ローカル・コピー中の複数の第１のサイズのデータ要素（７０～７４、８０～８２、２３０～２３４）と、１対１の関係で関連付けるステップと、
プロセッサによって修正されたローカル・コピーの第１のサイズのデータ要素ごとに、前記データ要素に関連するローカル・コピー書込みフラグを書込み値にセットするステップと、
ローカル・コピーがローカル・メモリからフラッシュされたとき、ローカル・コピーとローカル・コピー書込みフラグを大域メモリに送るステップと、
書込み値にセットされた各ローカル・コピー書込みフラグごとに、そのようなローカル・コピー書込みフラグに関連する第１のサイズのデータ要素をデータ・ブロックに記憶することにより、ローカル・コピーとデータ・ブロックを大域メモリ中で併合するステップと、
複数のプロセッサが同時にデータ・ブロックのアクセスしたときに、
少なくとも２個のプロセッサが、共に書込み値にセットされ、かつプロセッサ自体のそれぞれのローカル・コピーの対応する位置にある第１のサイズのデータ要素に関連するローカル・コピー書込みフラグを送る書込み／書込みデータ・レースの発生を識別するステップと、
を含むことを特徴とする複数のプロセッサ（２２ないし２４）間でメモリ（３２、３３）をコンピュータ・システムのメモリ共用方法である。
【０１０６】
４．　ローカル・コピー（５４、８６、２２４）とデータ・ブロック（１１８、１９６、２８８）を大域メモリ（３２、３３）中で併合するステップが、
それぞれ、最初は非書込み値にセットされる、大域メモリ中の複数のデータ・ブロック書込みフラグ（２９０ないし２９３）を、データ・ブロック中の複数の第１のサイズのデータ要素（２９６ないし２９９）と関連付けるステップと、
最初は第１の値にセットされる特殊併合フラグ（２８２）を、大域メモリ中のデータ要素と関連付けるステップと、
データ・ブロックが部分的に併合される間に記憶される各ローカル・コピーごとに、
部分的に併合されたデータ・ブロックの複製コピー（３０８）をローカル・コピー用のテーブル項目（３０９）に記憶するステップと、
特殊併合フラグを第２の値にセットするステップと、
特殊併合フラグが第１の値にセットされている場合、関連するデータ・ブロック書込みフラグが非書込み値にセットされている、データ・ブロックの対応する位置にある第１のサイズのデータ要素と異なるローカル・コピーのデータ要素を、データ・ブロックに記憶するステップと、
特殊併合フラグが第２の値にセットされており、ローカル・コピーは記憶されたが、データ・ブロックは部分的に併合されなかった場合、関連するデータ・ブロック書込みフラグが非書込み値にセットされている、データ・ブロックの対応する位置にある第１のサイズのデータ要素と異なるローカル・コピーのデータ要素を、データ・ブロックに記憶するステップと、

特殊併合フラグが第２の値にセットされており、ローカル・コピーが記憶され、同時にデータ・ブロックが部分的に併合された場合、ローカル・コピー用のテーブル項目中の複製コピーの対応する位置にある要素と異なるローカル・コピーのデータ要素をデータ・ブロックに記憶するステップとを含むことを特徴とする前記３に記載のコンピュータ・システムのメモリ共用方法である。

【０１０７】

５．　さらに、１つのデータ・ブロック中のデータにアクセスする各プロセッサ（２２ないし２４）ごとに、

最小サイズの修正されたデータ要素を示し、最初は、第１のサイズのデータ要素（８０ないし８２）のサイズである第１のサイズを示すようにセットされる、サブワード記憶フラグを、プロセッサのローカル・メモリ（４４ないし４６）で、ローカル・コピー（８６）と関連付けるステップと、

プロセッサがサブワード記憶動作をローカル・コピーに対して実行するとき、サブワード記憶フラグを第２のサイズを示すようにセットするステップと、

ローカル・コピーがローカル・メモリからフラッシュされたとき、ローカル・コピーおよびローカル・コピー書込みビットと共にサブワード記憶フラグを大域メモリに送るステップと、

サブワード記憶フラグが第２のサイズを示すようにセットされているとき、

最初は非書込み値に設定される大域メモリ（３２、３３）中の複数のデータ・ブロック書込みフラグ（１２６ないし１３７）を、データ・ブロック（１１８）中の複数の第２のサイズのデータ要素（１４６ないし１５７）と１対１の関係で関連付けるステップと、

関連するフラグが非書込み値にセットされているデータ・ブロックの第２のサイズの各データ要素ごとに、ローカル・コピーの対応する位置にあるデータ要素（９６ないし１０７）とそのような第２のサイズのデータ要素が異なる場合、そのような第２のサイズのデータ要素の代わりにローカル・コピー・データ要素を記憶し、そのような第２のサイズのデータ要素に関連するデータ・ブロック書込みフラグを書込み値にセットするステップと、

サブワード記憶フラグが第１のサイズを示すようにセットされているとき、

書込み値にセットされた各ローカル・コピー書込みフラグごとに、そのようなローカル・コピー書込みフラグに関連する第１のサイズのデータ要素をデータ・ブロックに記憶するステップとを含むことを特徴とする前記３に記載のコンピュータ・システムのメモリ共用方法である。

【０１０８】

６．　さらに、１つのデータ・ブロック（１９６）中のデータにアクセスする各プロセッサ（２２ないし２４）ごとに、

最初は非読取り値にセットされるローカル・メモリ（４４ないし４６）中の複数の読取りフラグ（２５０ないし２５４）を、ローカル・コピー（２２４）の第１のサイズのデータ要素（２３０ないし２３４）と、１対１の関係で関連付けるステップと、

第１のサイズのデータ要素がプロセッサによって読み取られたとき、前記データ要素に関連する読取りフラグを読取り値にセットするステップと、

ローカル・コピーがローカル・メモリからフラッシュされたとき、ローカル・コピーおよびローカル・コピー書込みフラグ（２４０ないし２４４）と共に読取りフラグを大域メモリ（３２、３３）に送るステップと、

複数のプロセッサ（２２ないし２４）が同時に同じデータ・ブロック（１９６）にアクセスしたとき、

１個のプロセッサが書込み値にセットされたローカル・コピー書込みフラグを送り、他のプロセッサが読取り値にセットされた読取りフラグを送る読取り／書込みデータ・レースの発生を識別するステップとを含み、前記書込みフラグおよび読取りフラグが、前記プロセッサのそれぞれのローカル・コピーの対応する位置にある第１のサイズのデータ要素に関連付けられることを特徴とする前記３に記載のコンピュータ・システムのメモリ共用方法である。

【０１０９】
７．　さらに、最初はロック解除値にセットされるロック・フラグ（１６０）を大域メモリ（３２、３３）中の各データ・ブロック（１１８）と関連付けるステップと、
第１のプロセッサ（２２ないし２４）の要求に応じてロック・フラグをロック値にセットするステップと、
ロック・フラグがロック値にセットされている間、どのプロセッサによるデータ・ブロックへのアクセスも禁止し、そのようなプロセッサによるデータ・ブロックへのアクセスまたはロックを求める要求を中断アクセス要求待ち行列（１６２）に入れるステップと、
第１のプロセッサの要求に応じてロック・フラグをロック解除値にリセットするステップと、
ロック・フラグがロック解除値にリセットされたとき、中断アクセス要求待ち行列中の要求を処理するステップとを含むことを特徴とする前記３に記載のコンピュータ・システムのメモリ共用方法である。
【０１１０】
８．　さらに、プロセッサ（２２ないし２４）の要求に応じて、大域メモリ（３２、３３）中のデータに対して大域リダクション演算を実行し、大域リダクション演算の結果を大域メモリの指定された位置に記憶するステップを含むことを特徴とする前記３に記載のコンピュータ・システムのメモリ共用方法である。
【０１１１】
９．　さらに、データ・ブロック（１１８、１９６、２８０）の複数の大域コピーを、複数の大域メモリ（３２、３３）のそれぞれに１つずつ記憶するステップと、
複数の大域メモリのそれぞれに複数のプロセッサ（２２ないし２４）を関連付けるステップと、
プロセッサ（２２ないし２４）要求がデータ・ブロックにアクセスするとき、そのようなプロセッサに関連する大域メモリ中のデータ・ブロックの大域コピーにアクセスするステップとを含み、
大域コピーにアクセスするステップが、
アクセス側プロセッサのローカル・メモリ（４４ないし４６）に大域コピー（５４、８６、２２４）のローカル・コピーを記憶するステップと、
ローカル・コピーがローカル・メモリからフラッシュされたとき、アクセス側プロセッサに関連する大域メモリ中の大域コピーとローカル・コピーを併合するステップと、
１つの大域コピーのローカル・コピーがすべてそのような大域コピーと併合された後に、そのような大域コピーを他の大域コピーと併合するステップとを含むことを特徴とする前記３に記載のコンピュータ・システムのメモリ共用方法である。
【０１１２】
１０．　共用メモリ多重処理コンピュータ・システム（２０）において、
それぞれ複数のマルチビット・データ要素（１２０ないし１２２、１９０ないし１９４、２９６ないし２９９）を備える共用データ（３２、３３）の複数のデータ・ブロック（１１８、１９６、２８０）を記憶するための大域メモリを備える、１つまたは複数の大域メモリ装置（２６、２７）と、
それぞれ、最初、ローカル・コピーがローカル・メモリに記憶されるときに非書込み値に設定される、ローカル・コピーのデータ要素（７０ないし７４、８０ないし８２、２３０ないし２３４）と１対１の関係で関連付けられた複数のフラグ（６０ないし６４、９０ないし９２、２４０ないし２４４）を備える書込みビットマスク（５６、８８、２２０）と、１つのデータ・ブロックのローカル・コピー（５４、８６、２２４）とを、データ・ブロック中のデータへのアクセスがプロセッサによって要求されたときに記憶するためのローカル・メモリ（４４ないし４６）と、前記プロセッサ（３６ないし３８）とを備える１つまたは複数の大域メモリ装置と相互接続された複数の処理装置（２２ないし２４）とを備え、
各処理装置がさらに、それぞれ、プロセッサによって書込みが行われたデータ要素に関連

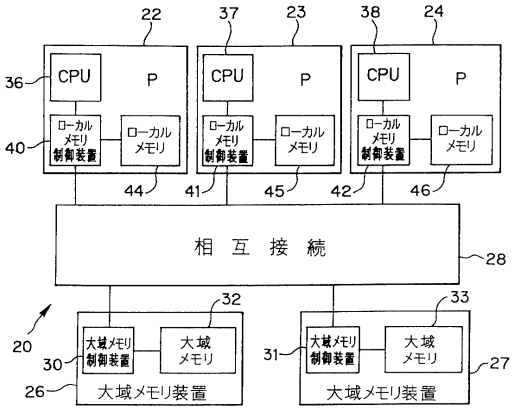
する書込みビットマスク・フラグを書込み値にセットし、ローカル・コピーがローカル・メモリ中の他のデータと置換されたとき、プロセッサによってデータ要素のどれかに書込みが行われている場合に書込みビットマスクおよび書込みが行われたデータ要素を大域メモリに送るように動作するローカル・メモリ制御装置（４０ないし４２）を備え、前記コンピュータ・システムがさらに、
関連する書込みビットマスク・フラグが書込み値にセットされた各データ要素をデータ・ブロックに記憶することによって、１個の処理装置から受け取ったローカル・コピーを併合するように動作する大域メモリ制御装置（３０、３１）を大域メモリ装置中に備えることを特徴とするコンピュータ・システムのメモリ共用方法に適用する共用メモリ多重処理コンピュータ・システムである。

【０１１３】
【発明の効果】
以上のように、本発明によれば、複数のデータ・ブロックを大域メモリに記憶し、各プロセッサの要求に応じてデータ・ブロックのうちの発信側データ・ブロックのコピーを要求側プロセッサにアクセスできるように、要求側プロセッサに対してローカル・メモリまたはキャシュ・メモリに記憶し、コピーのどの要素が要求側プロセッサによって修正要求されたかを追跡し、追跡されたコピーの要素をデータ・ブロックに記憶することにより、コピーを併合するようにしたので、プロセッサがデータ・ブロックを何度でもキャシュして修正することができるとともに、プロセッサが併合の開始から完了までにデータ・ブロックのコピーをさらにキャシュするのを妨げる必要がなく、中断機構を必要としなくなる。

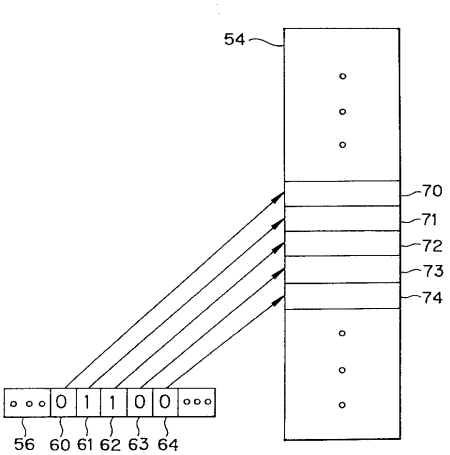
【図面の簡単な説明】
【図１】本発明によるコンピュータシステムのメモリ共用に適用される共用メモリ多重処理コンピュータ・システムの概略図である。
【図２】書込みビットマスク中のフラグと、本発明の第１の実施例による図１の共用メモリ多重処理コンピュータ・システムのローカル・メモリに記憶されたキャッシュされたデータ・ブロックのデータ要素との関係を示す説明図である。
【図３】書込みビットマスク・フラグと、本発明の第２、第３、第９の実施例による図１の共用メモリ多重処理コンピュータ・システムのローカル・メモリに記憶されたキャッシュ・データ・ブロックとの関係を示す説明図である。
【図４】データ・ブロックに関連し、本発明の第３および第４の実施例による図１の共用メモリ多重処理コンピュータ・システムの大域メモリに記憶された、大域書込みビットマスク、中断／ロック・フラグ、中断待ち行列、およびカウントを示す説明図である。
【図５】本発明の第５の実施例によってデータ・エラーを識別するために図１の共用メモリ多重処理コンピュータ・システムのローカル・メモリおよび大域メモリに記憶された、ローカル書込みビットマスクおよびローカル読取りビットマスクならびに大域書込みビットマスクおよび大域読取りビットマスクを示す説明図である。
【図６】本発明の第７の実施例による図１の共用メモリ多重処理コンピュータ・システム中のプロセッサおよび大域メモリ装置の階層順序付けを示す説明図である。
【図７】データ・ブロックに関連し、本発明の第８の実施例による図１の共用メモリ多重処理コンピュータ・システムに記憶された、大域書込みビットマスク、特殊併合フラグ、およびテーブルを示す説明図である。
【図８】本発明の第９の実施例による図１の共用メモリ多重処理コンピュータ・システム中のＣＰＵの内部レジスタおよびビット修正フラグを示す概略ブロック図である。
【符号の説明】
２０　　共用メモリ多重処理コンピュータ・システム
２２～２４　プロセッサ
２６、２７　大域メモリ装置
２８　　相互接続通信媒体
３０、３１　大域メモリ制御装置
３２、３３　大域メモリ

３６～３８　中央演算処理装置
４０～４２　ローカル・メモリ制御装置
４４～４６　ローカル・メモリまたはキャッシュ・メモリ
５４、８６、２２４　コピー
５６、８８　書込みビットマスク
６０～６４、１２６～１３６、２００～２０４、２１０～２１４、２４０～２４４、２５０～２５４　フラグ
７０～７４、１９０～１９４、２３０～２３４　データ要素
８０～８４、１２０～１２２　ワード
１１８　発信側データ・ブロック
１２４　サブワード書込みビットマスク
１６０　中断フラグ
１６２　中断待ち行列
１８０、２２８　大域書込みビットマスク
１８２　大域読取りビットマスク
１８４、２８０　データ・ブロック
２２０　ローカル書込みビットマスク
２２２　ローカル読取りビットマスク
２７０　階層順序付け
２８２　特殊フラグ
２８４　特殊併合テーブル
３０９　項目
３１０　プロセッサ識別子
３３８　機能単位
３４０～３４７　ビット修正フラグ

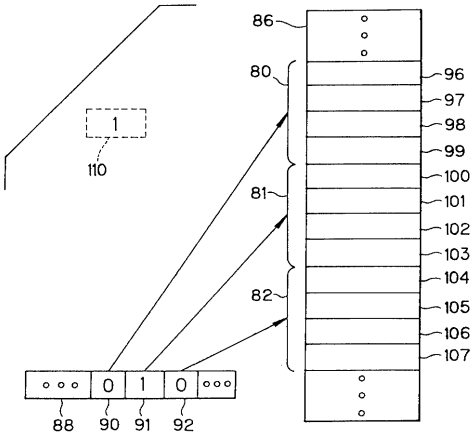
【図１】



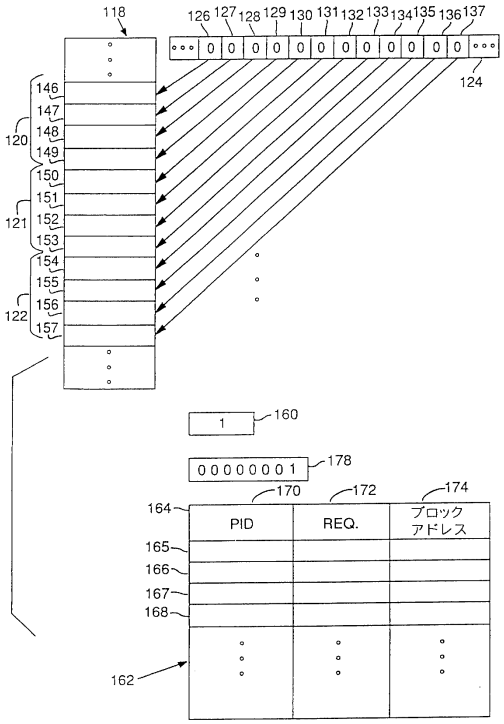
【図２】



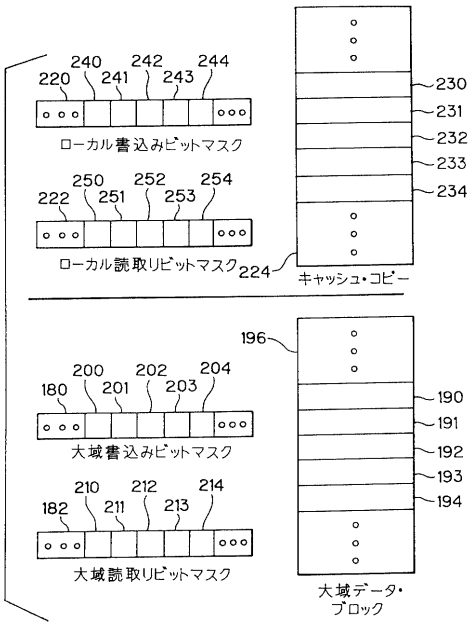
【図３】


86
80
81
82
96
97
98
99
100
101
102
103
104
105
106
107
1
110
0
1
0
88
90
91
92

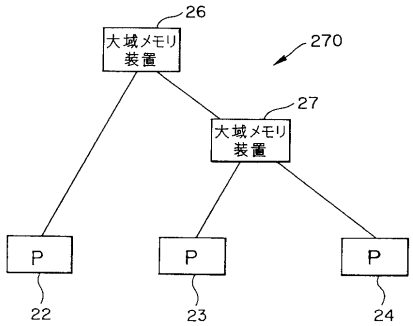
【図４】


118
126 127 128 129 130 131 132 133 134 135 136 137
0 0 0 0 0 0 0 0 0 0 0 0
124
120
121
122
146 147 148 149 150 151 152 153 154 155 156 157
1
160
00000001
178
170
172
174
164
PID
REQ.
ブロック
アドレス
165
166
167
168
162

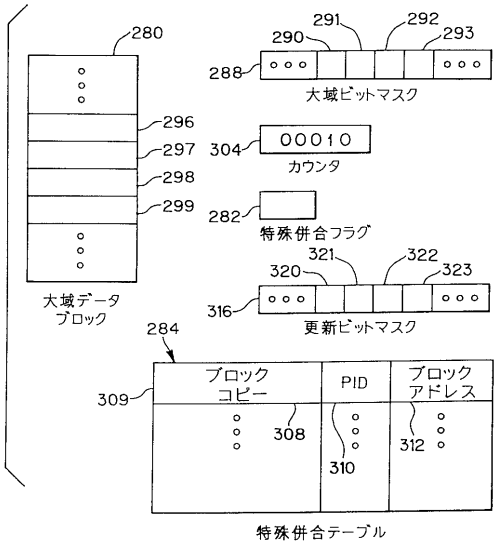
【図５】


240 242 244
220 241 243
ローカル書込みビットマスク
250 252 254
222 251 253
ローカル読取りビットマスク
224
キャッシュ・コピー
230
231
232
233
234
196
200 202 204
180 201 203
大域書込みビットマスク
210 212 214
182 211 213
大域読取りビットマスク
大域データ・
ブロック
190
191
192
193
194

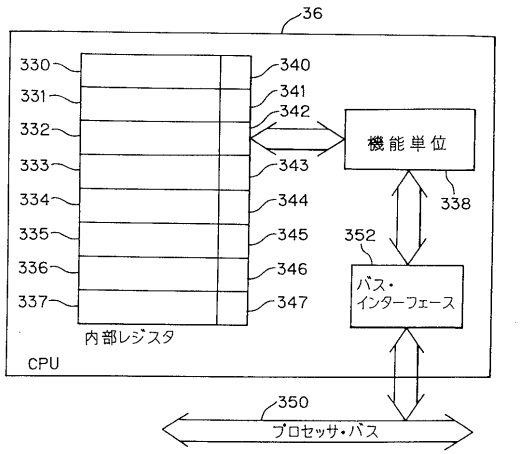
【図６】


26
大域メモリ
装置
270
27
大域メモリ
装置
P
P
P
22
23
24

【図７】


280
291
290
292
293
288
大域ビットマスク
296
297
304
00010
カウンタ
298
299
282
特殊併合フラグ
321
320
322
323
316
更新ビットマスク
大域データ
ブロック
284
ブロック
コピー
PID
ブロック
アドレス
309
308
310
312
特殊併合テーブル

【図８】


36
330
331
332
333
334
335
336
337
340
341
342
343
344
345
346
347
機能単位
338
352
バス・
インターフェース
内部レジスタ
CPU
350
プロセッサ・バス

フロントページの続き

(51)Int.Cl.[7]	FI	
	G06F 12/08	565
	G06F 12/08	583
	G06F 12/00	560F
	G06F 12/00	572A
	G06F 15/167	610A
	G06F 9/46	475C
	G06F 9/46	472B

審査官 清木 泰

(56)参考文献 特開平4－260146（JP，A）
特開平4－195563（JP，A）
特開平4－279960（JP，A）
特開平4－71046（JP，A）
特開平5－233531（JP，A）
特開平3－25541（JP，A）
特開平5－233421（JP，A）
Brian W. O'Krafka, A. Richard Newton, An Empirical Evaluation of Two Memory-Efficient Methods, Proceedings of 17th Annual International Symposium on Computer Architecture, IEEE, 1990年 5月28日, pages:138-147
Alan H. Karp, Vivek Sarkar, Data Merging for Shared-Memory Multiprocessors, Proceedings of Hawaii International Conference on System Sciences, 1993年 1月, pages:244-256
細見岳生、外3名，ソフトウェアとハードウェアのコヒーレンス制御を融合したキャッシュ・コヒーレンス制御，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，1993年 3月12日，第92巻，第497号，(CPSY92-80～92)，p.73-80
寺澤卓也、外2名，マルチプロセッサの記憶システム（1），情報処理，日本，社団法人情報処理学会，1993年 1月15日，第34巻，第1号，p.96-105
小口正人、外2名，複製型共有メモリにおけるメモリ内容の一貫性についての検討，情報処理学会研究報告，日本，社団法人情報処理学会，1992年11月20日，第92巻，第91号，(92-DPS-58)，p.181-188
徳永尚哉、外2名，高性能プラットフォーム要素マイクロプロセッサ・アーキテクチャー通信モデルに関する検討，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，1992年 8月21日，第92巻，第173号，(CPSY92-20～33)，p.1-8
Bill Nitzberg, Virginia Lo, Distributed Shared Memory: A Survey of Issues and Algorithms, COMPUTER, IEEE, 1992年 8月, Vol.24, Issue.8, pages:52-60
Yung-Syau Chen, Michel Dubois, Cache Protocols with Partial Block Invalidations, Proceedings of Seventh International Parallel Processing Symposium, IEEE, 1993年 4月13日, pages:16-23

(58)調査した分野(Int.Cl.[7]，DB名)
G06F12/08-12/12
G06F15/16-15/177
G06F 9/46- 9/54
G06F12/00,550-12/06
G06F13/16-13/18