汎用並列オペレーティングシステムにおける資源保護と仮想化

松本 尚　平木 敬

汎用並列オペレーティングシステムでは、汎用性の確保と効率の良い並列実行環境の実現という同時に成立させることが困難な目標を達成する必要がある。効率の良い並列実行環境の実現するためにはノード間の高速かつ保護され仮想化されたユーザレベル通信および同期のサポートが不可欠である。我々は汎用並列オペレーティングシステムに適した高速なユーザレベル通信同期として、他ノードのメモリ空間内のデータを直接読み書きするソフトウェアメモリベース通信を考案し開発している。しかし、メモリベース通信を許すことによって、不当なジョブ間の相互干渉が避けられないのでは、汎用オペレーティングシステムとは呼べない。本稿では、メモリベース通信を採用した場合の汎用並列オペレーティングシステムにおける資源保護と仮想化方式に関して議論を行なう。結論として、メモリ管理機構を活用した保護方式によって、関連するアクティビティ間のメモリ保護が不要な並列実行はもとより、サーバクライアント実行のような不当な相互干渉の排除が不可欠なアプリケーションにもメモリベース通信が適用可能である。

Protection and Virtualization of Resources in a General-Purpose Parallel Operating System

Takashi MATSUMOTO and Kei HIRAKI

A general-purpose operating system for parallel systems must satisfy two capabilities that contradict each other: realizing protected and time-shared execution environment, and providing efficient parallel-execution environment. In parallel executions with a general-purpose operating system, performance of the protected and virtualized user-level communications/synchronizations is the most crucial issue. We proposed a novel high-speed user-level communications/synchronizations scheme "Memory-Based Communication Facilities (MBCF)" suitable for the general-purpose parallel operating system with off-the-shelf communication-hardware. For achieving high-performance, MBCF adopts the direct remote-accesses to destination user-level memory-space without address checks. In this paper, we discuss aspects of protection and security on MBCF. We conclude MBCF is qualified for not only parallel processing but also server-client distributed-computations which require strict protection and security.

1. はじめに

本稿において対象とする汎用並列オペレーティングシステムは、既存オペレーティングシステムに実現されているマルチユーザ / マルチジョブの汎用性と、効率の良い並列実行環境の実現を両立させるものである。並列処理の最大の特徴はプロセッサおよびノード間の通信と同期であり、高速なユーザレベル通信同期なしには効率の良い並列実行環境はあり得ない。本稿ではユーザレベルの通信同期に焦点を当て、保護と仮想化の要件十分に満足しつつユーザレベル通信同期を高速に実装する方式について議論を行う。

集中共有メモリを持つ並列計算機や従来のページ管理機構を遠隔メモリアクセスに拡張した Memory-Based Processor (**MBP**) [1] を持つ並列計算機では、プロセッサは共有メモリ領域への通常のメモリアクセスで通信同期を行う。ユーザプログラムはマッピングされたページにしかアクセスできないため、ページ管理機構によってジョブ間の不当干渉を排除することが可能である。つまり、これらの計算機ではユーザレベルの通信 (同期) を通常のメモリの load/store で実現しており、保護に関してはプロセッサのメモリ保護機構の方式がそのまま流用可能である。しかし、集中共有メモリ型並列計算機は集中共有メモリへのアクセスがボトルネックとなり、プロセッサ台数の大規模なものを製造することが困難である。分散メモリ実装の並列計算機やワークステーションクラスタ (**NOW**: Network of Workstations) においては、MBP と同様なハードウェアを付加することによって、メモリアクセスとして高速かつ保護され仮想化されたユーザレベル通信同期が実現できる。けれども、MBP タイプのハードウェア付加機構はまだ一般的でない。また、MBP は主要素プロセッサのメモリアクセス動作と密に協調して働くため、

† 東京大学 大学院理学系研究科 情報科学専攻
Department of Information Science, University of Tokyo

MBP の実装はプロセッサのメモリ周りの実装に依存してしまう可能性がある。

これらの理由から、汎用並列オペレーティングシステムの開発に当たって、我々は集中共有メモリや MBP と同様な通信同期ハードウェアを仮定しない分散メモリ実装の並列計算機環境 (NOW を含む) においても実現可能な、高速かつ保護され仮想化されたユーザ通信 / ユーザ同期を考案する必要にせまられた。これに対して我々が出した回答がソフトウェアメモリベース通信機能 (**MBCF**: Memory-Based Communication Facilities) である。以下本稿では、 MBCF の保護および仮想化の設計方針と能力についての議論を行なう。

2. プログラム実行モデルとアドレス空間

本稿では、我々が開発中の汎用超並列オペレーティングシステム SSS–CORE [2] の用語に従って説明を行うため、本節において必要な用語を定義する。 SSS–CORE はワークステーションクラスタ (NOW) および分散メモリ型並列計算機のためのオペレーティングシステムであり、各計算ノードには同一タイプのプロセッサが使用されていることを仮定している。計算ノードは単一プロセッサのマシンである必要はなく、 UMA 型のクラスタ (例えば、密結合マルチプロセッサ型ワークステーション) であることを許している。

一台のプロセッサに割り当てられるプログラムの実行の軌跡 (命令流) のことをスレッドと呼ぶ。複数のスレッドで 1 つのプロセスを構成し、プロセスの内で同一ノード (クラスタ) に属するスレッドがサブプロセスを構成する。さらに、プロセス内の論理的にも物理的にも完全にメモリ空間を共有するスレッドがタスクを構成する。タスクはメモリ空間 (コンテクスト識別子) の割り当て単位であり、従来の OS ではこのタスクがプロセスと呼ばれることが多い。タスクはサブプロセスと一致する場合もあり、複数のタスクが一つのサブプロセスに包含される場合もある。しかし、タスクが複数のサブプロセス (つまりノード) に跨って存在することはない。プロセスの識別子 (プロセス ID) は並列システム全体で一意であり、プロセス ID が一致するサブプロセス / タスク / スレッドは同一プロセスに属している。タスクの識別子 (タスク **ID**) はノード (サブプロセス) 内で一意であり、ノード識別子 (ノード **ID**) とタスク ID の組合せでシステム内の一つのメモリ空間を特定することができる。

SSS–CORE のカーネルプログラムは各ノード (クラスタ) に常駐しており、ノード間で協調動作することでシステム全体の管理を行う。プロセスからカーネルレベルの資源スケジューラにスケジューリングヒントが提示され、資源割り当ての対象を通信距離的に近接したノードのみに限定したり、同一プロセスに属するタスクに対する実プロセッサ割り当てを複数のノードに跨って同一タイムスライスに実現させることが可能である [2]。同様にプロセスレベルのスケジューリングヒントによって、 1 タスクに同時に割り当てられるプロセッサ数 (ただし最大値はノード内の実プロセッサ数) を変化させることも可能である。 1 タスクに複数の実プロセッサが同時割り当てされた場合は、メモリ空間は完全に重なっており、スタック領域の管理までユーザレベルで行う必要がある。逆に言えば、ユーザレベルで複数台のプロセッサに自由にスレッドを割り付けてスケジューリングの制御ができる。

3. ソフトウェアメモリベース通信

MBCF の保護および仮想化の側面についての議論を行なう前準備として、 MBCF の簡単な解説と MBCF の汎用性について述べる。つまり、 MBCF が単に我々が開発中の並列オペレーティングシステム SSS–CORE に特化した通信方式ではなく、高性能マイクロプロセッサのページ管理機構を活用し、十分に一般性をもった高速かつ保護され仮想化されたユーザ通信 / ユーザ同期を実現する方式であることを説明する。

3.1 メモリベース通信の概要

マルチユーザ / マルチジョブの汎用システムを目標とするため、並列アプリケーションはユーザレベルでは保護と仮想化の下で実行される必要がある。この保護と仮想化の制約を守るために従来型オペレーティングシステムの TCP/IP や UDP/IP による通信を行うと、システムコールのオーバヘッドやプロトコルの限界のために大きなコストがかかる。我々は特殊な通信ハードウェアがなくても実装可能なソフトウェアメモリベース通信機能 (**MBCF**) を考案し実装を行った。 MBCF ではノード間の通信を通信相手先ノードの論理的メモリ空間への直接アクセスと捉えることで、高性能マイクロプロセッサの高メモリアクセスバンド幅とメモリ管理機構を最大限活かしたユーザレベル通信を実現する。 MBCF は機能的には MBP [1] で提案された各種メモリベースの通信同期機能を純粋にソフトウェアによって実現している。しかし、各種機能実現のための通信以外の付加的操作は、ノード内のローカルオペレーションに過ぎず、ワーキングセットも大きくないため、高性能マイクロプロセッサによって高速に処理可能である。保護の観点から、既存の通信ハードウェアがユーザに直接操作させられないために、 MBCF の実装でもシステムコールは必要とされる。しかし、この MBCF 用のシステムコールは専用システムコールとして実装され、コピーや無駄なスケジューリング動作等を撤廃することにより非常に高速になっている。

MBCF の定性的な側面からの説明は他の文献 [3] に紹介済みであり、本稿では保護や仮想化が議論可能な具体性を持たせるために、現状の実装に即した説明を行う。 MBCF にとってノードの通信ハードウェアは高速高性能であるに越したことはないが、通信方式や種別は基本的に問わない。現在、 MBCF は Ethernet (ISO/IEC 8802-3) および Fast Ethernet (IEEE 802.3u 100Base-T) 上で実現されている。ただし、

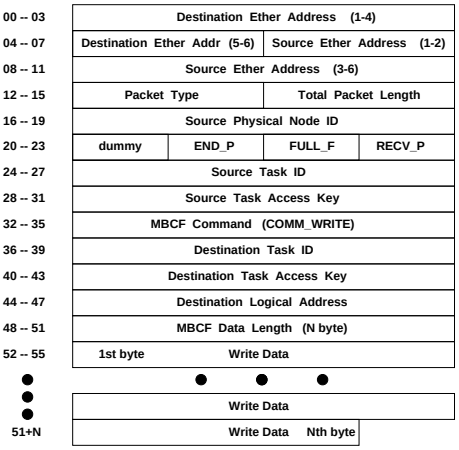

図1　遠隔書き込みの MBCF パケット

Ethernet はパケットの到着がハードウェアレベルで保証されない通信規格であるため、パケットの転送到着保証およびパケットの到着順序保証（FIFO 性保証）のプロトコルが MBCF の現実装においては付加され、MBCF の処理を若干複雑にしている。

3.2　MBCF パケットの構造

Ethernet のパケットは最大 1500byte のデータを運べるため、レイテンシ削減重視の MBCF の現実装では 1 パケットを Ethernet の 1 パケットと一致させている。

図 1に遠隔書き込みを行う MBCF パケットを示す。便宜上、パケットの先頭を 0 番地として byte 単位のアドレスによってパケット内の位置を示すことにする。0 番地から 19 番地までは Ether 上の IP パケットと同形式をしている。ただし、Packet Type として IP 等の他のプロトコルと衝突しないものを使用し、Source Physical Node ID は IP アドレスではなく MBCF 用の独自のノード ID を採用している。20 番地は空いている。21 番地から 23 番地までの 3 個の 1byte 長の制御データは通信到着保証と FIFO 性保証と Acknowledge 制御のために使用されている。24 番地、28 番地にはそれぞれパケット送信元のタスク ID とアクセスキーが格納される。0 番地から 31 番地までのパケットの内容は必ずカーネルレベルのルーチンによってセットアップされる。32 番地から 51 番地までのデータはユーザによるパケット発信時にはユーザレベルでセットアップされる。ただし、32 番地の MBCF Command の上位ビットはパケット再送制御等に使用されるため、カーネルルーチンで正しく再設定される。32 番地には MBCF の機能を示すコマンドが書かれ、36 番地にはパケットの宛先のタスク ID が格納され、40 番地には宛先タスクのアドレス空間へのアクセスキーが格納される。後述するようにアクセスキーが間違っている場合は単に通信（この場合は書き込み）が失敗するだけではなく、セキュリティ保持のためのペナルティが発信元に課される場合もある。44 番地に宛先タスク内の書き込みを開始する論理アドレスが格納され、48 番地に書き込みデータ長が格納される。52 番地からは実際の書き込みデータ

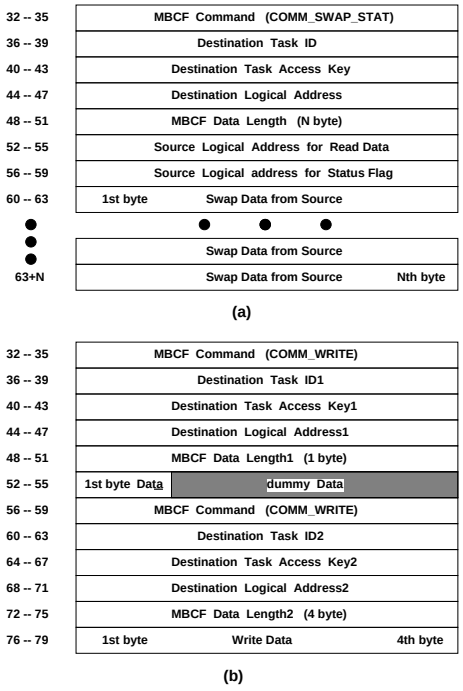

図2　MBCF パケットのバリエーション

続く。ただし、ユーザがシステムコールで MBCF 送信ルーチンを呼ぶ際には書き込みデータはユーザ空間内のポインタで渡され、カーネル内でパケット作成のために一度だけコピーされる。

図 2に MBCF パケットの他の実例を示す。0 番地から 31 番地までは図 1の遠隔書き込みパケットと同一であるため省略してある。図 2(a) には遠隔ノード上の複数データを不可分に SWAP するコマンド「COMM_SWAP」のバリエーションの一つである「COMM_SWAP_STAT」を示す。COMM_SWAP_STAT はこの処理が終了したことを発信元タスクのフラグ変数（56 番地にポインタ）に 1 を書き込むことで通知する。また、SWAP で遠隔操作対象のタスク内のデータを返送するためのポインタ（送信元タスク内の論理アドレス）が 52 番地に含まれている。コマンドバリエーションを示すことは本稿の主旨ではないので、全コマンドの詳細な説明は他稿に譲る。図 2(b) には同一タスクから発信された複数の MBCF コマンドが 1 パケットのまとめられた形式のパケットを示す。この図では二つのタスク（同一の可能性もある）への 1byte と 4byte の遠隔書き込みを 1 パケットで行っている。複数の MBCF Command が含まれていても再送制御やフロー制御等は 32 番地の最初の MBCF Command の上位ビットのみで行うため、任意の MBCF コマンドが一つのパケットにまとめられるわけではない。複数の MBCF コマンドがまとめられる（コンバイニング可能）かどうかは、返送パケットの有無や返送パケットのサイズおよびマルチキャスト動作の経路等で決まる。

3.3 MBCF の通信処理概要

MBCF の発信時:

(1) ユーザが通信(同期)用のパケットを組み立てる。(宛先情報は論理タスク番号)

(2) ユーザ組み立てたパケットを引数にして MBCF 発信専用システムコールを呼ぶ。

(3) カーネル内でセキュリティ、通信保証、FIFO 性保証のためのデータを付加し、論理タスク番号をノード ID とタスク ID の組に変換して通信ハードウェアに転送する。通信保証のためパケットの 1 コピーをカーネル内に保持し、通信の確認がなされた時点で領域を回収する。(Ethernet,Fast Ethernet のコントローラでは DMA によるパケット転送領域をそのままパケット保存領域に流用できるため、カーネル内のパケットコピーは最小の 1 回に抑えられている。)

(4) すぐにユーザルーチンに戻る。

MBCF の受信時: (データは必ずメモリ上で受け渡されるため受信用システムコールは存在しない。)

(1) 通信ハードウェアからプロセッサに受信割り込みがかかる。

(2) 受信割り込みルーチンでパケットのシーケンスチェックを行い、パケットの欠落なしに FIFO 性が保たれて通信が行われていることを確認する。欠落を発見した場合、到着したパケットを破棄し、欠落している番号のパケットの再送要求を行い、割り込み前のルーチンに復帰する。

(3) パケットが正しく到着した場合、アクセスキーの内容が正しいことをチェックして、カーネル内のまま一時的にアドレス空間を指定されたタスクに切替えてメモリ操作を行う。メモリ操作後リプライを要求するコマンドならば、返送用パケットがカーネル内で組み立てられて返送される。ただし、パケットのシーケンス管理とパケット保存領域の解放のため、ユーザが返送を要求しないパケットでも、MBCF プロトコルに基づいて Acknowledge を返送することがある。

(4) 割り込み前のルーチンに復帰する。

この他に、Acknowledge や再送要求メッセージ等の制御パケットが転送中に消失した場合に備えて、タイマ割り込み時にステートをチェックする割り込みルーチンも存在する。

3.4 割り込み処理による公平性の阻害について

MBCF では受信時の動作をノードのプロセッサに割り込みをかけて行わせるため、プロセスやタスクへの実プロセッサのスケジューリングと無関係にいつでも通信が可能である。その反面、関係のないプロセスが走行中にも受信処理が行われることになる☆。しかし、通常の TCP や UDP 通信であっても受信のためにカーネルの割り込みルーチンが走ることに変わりはない。違いは、MBCF ではこの割り込みルーチン内で目的タスクの操作対象メモリを直接操作してしまう点である。パケットの内容をカーネル内に保持する必要がないため、メモリコピーの回数が削減できる。また、専用受信割り込みルーチンでは MBCF の受信処理以外は何もしないため、数 μsec から数十 μsec 程度(コピーのためパケットのデータ長に依存、read,write では 1 回コピー、swap なら read&write で 2 回コピー、階層マルチキャストなら分裂数 +1 回コピー)の中断で済む。この最低限の中断を厭う向きには、受信時の処理を MBP のような通信専用のプロセッサに行わせるシステムを提供する必要がある。

☆ ただし、操作対象メモリがスワップアウトされている場合は、その操作対象タスクに関する MBCF パケットが後続分も含めてカーネル内にバッファリングされ、タスクが属するプロセスのスケジューリング時に対象メモリのスワップイン操作も含めて処理される。

3.5 アドレス空間の高速切替について

最近の高性能マイクロプロセッサの TLB にはコンテクスト識別子(アドレス空間の識別子)がタグとして含まれるようになり、複数のアドレス空間のページエントリが TLB 内に混在できるようになっている。そのため、アドレス空間の一時的な切替が TLB のエントリを小数本入れ換えるだけで高速に実行可能になった。また、受信割り込みルーチンから戻った時点で TLB のエントリはほとんど割り込み前と変わっていないため、処理スピードを落さずに割り込み前の処理が継続できる。

3.6 ユーザレベル受信ルーチン不採用理由

メモリベース通信には多種ではあるが固定された機能(コマンド)のみが用意されている。しかし、これを一般化してユーザルーチンを受信ルーチンとして使用することによる機能拡大は効率および保護と公平性の観点から行っていない。Active Message (AM) [4] はまさにユーザルーチンを受信ルーチンとして使用する通信方式である。しかし、AM を汎用環境で実現する場合、外部割り込み直後のプロセッサはカーネルモードであるため、直接ユーザレベルのルーチンを呼び出すことができない。モードを変更するルーチンを一段介したり、ユーザレベルのハードウェア割り込みを実装したりしても、ユーザレベルの受信ルーチンでは直接通信ハードウェアを操作することは保護の観点から難しいうえ、ユーザルーチンの処理時間の上限が抑えられない。このため、NOW 上で AM を実現した SSAM [5] では一旦パケットをカーネル内のバッファで受け取って、対象タスク(通常の意味のプロセス)が in core の状態になった時点でユーザルーチンを起動してバッファ内のパケットを処理させている。この方式ではタスクのスケジューリングによっては返送パケットの発送が大幅に遅れる場合がある。また、MBCF 方式よりも本質的に 1 回余分なパケットのコピー(カーネル内のバッファへのコピー)が生じてしまう。

さらに、通常のアプリケーションプログラムの通信同期では、write,read,swap, update,page_invalidate, データ queue 登録(メモリベース FIFO), barrier といった予めデフォルトのコマンドとして用意されている MBCF の機能で十分である。

3.7 MBCF と第一世代 MPP の通信性能比較

MBCF の詳細な性能評価は他稿に譲るが、MBCF の性能の一端を示すためにノード間通信用専用ハードウェアを持つ第一世代 MPP（通信ソフトウェア込み）との通信性能比較を表 1に示す。MBCF の測定環境は Axil 320 model8.1.2 (Sun SS20 互換機, 85MHz SuperSPARC CPU × 2) に Sun Microsystems 社製の Fast Ethernet SBus Adapter 2.0 を追加して、100baseTX のハブで FastEtherNet 接続したものである。表中の SSAM [5] は MBCF と同様にワークステーションクラスタベース（ただしネットワークは 156Mbps ATM）のソフトウェアによる通信機構である。ただし、SSAM はパケットの到着保証と順序保証のプロトコルを省略しているため、実用化時にはこの値よりも悪くなることが予想される。表中の MBCF 以外の性能値は文献 [5] から引用した。

表1　メモリベース通信の性能比較

Machine	Peak bandwidth (Mbytes/s)	Round-trip latency(μs)
SP–1 + MPL/p	8.3	56
Paragon + NX	7.3	44
CM–5 + Active Message	10.0	12
SS–20 cluster + SSAM	7.5	52
SS–20 cluster + MBCF	11.2	49

我々の MBCF は CM–5 上の Active Message (AM) に Round-trip タイムで劣っているが、MBCF が一切特殊ハードウェアを必要としないこと、AM/CM–5 が仮想化や保護に対応していないことを考慮すれば、我々の MBCF の性能が優れていることが判る。

4. アクセスキーによる保護およびセキュリティ

基本的に MBCF は機能的には MBP の動作をソフトウェアのみによってエミュレートしているが、MBP が内蔵 TLB を使って実現しているページ管理機構の動作をソフトウェア的にエミュレーションすることはオーバヘッドが大きい。このためにキーによるアクセス管理を行っている。タスク（アドレス空間）ごとにアクセスキーが登録してあり、パケット内のアクセスキーが一致した場合のみ対象タスクの操作対象論理アドレスのメモリ操作が行われる。並列アプリケーション等で、通信相手のタスクのアクセスキーを静的に知っている場合は、実行コード内に他タスクのキーを埋め込めむことで MBCF 通信が可能になる。しかし、MBCF をサーバクライアントのような分散処理の通信手段として使用する場合にはアクセスキーの取得方法が問題となる。もちろん、従来の TCP や UDP による通信と MBCF を共に実装して、アクセスキーの獲得は他の通信手段に頼るという解決方法もある。しかし、我々は MBCF のみで自己完結可能で、TCP や UDP を流用するよりも安全なインタフェースを用意している。

MBCF にはタスクごとにユニークな通信キュー（taskque）が用意されており、taskque の使用 / 不使用をその owner のタスクが宣言できる。taskque への登録メッセージ（パケット）のみが操作対象論理アドレスと対象タスクのアクセスキーを持たずに発行可能であり、到着順に taskque に格納される。taskque の使用が宣言されていない場合は、taskque 宛の通信は不正なものであるため、その発信元タスクを kill するカーネルレベルのメッセージが発信ノードに返送される。taskque への登録メッセージの発行には発行元のアクセスキーがカーネルによって必ず付与される。そして、受信側タスクは taskque に登録したタスクに関してのみ、そのアクセスキーを取り出すことができる。通常の MBCF パケットにも発信元のアクセスキーが付与されているが、これらはユーザレベルから読み出すことはできない仕様になっている。また、taskque にパケットを送りつけたタスクに対しては、taskque の owner は kill メッセージを発行する権利を持つ（kill メッセージの正当性は正しいアクセスキーで保証される）。つまり、サーバタスクの taskque にパケットを登録したクライアントタスクはそのサーバタスクに生殺与奪の権を与えたことになる。この taskque に自分の方から登録したという情報はサーバ側が明示的にクライアントデータを抹消しない限り保持され、その後の MBCF において起こったすべてのエラーは taskque に登録したクライアント側の責任となる。

クライアントタスクがサーバの taskque にパケットを登録するのは、通常メモリベース通信を開始するための情報（アクセスキーや操作対象論理アドレス等）を獲得するためである。この情報を得るためのプロトコルはユーザアプリケーション依存で、セキュリティの度合を高めたい場合には複雑な認証プロトコルやケイパビリティリストをアプリケーションレベルにおいて実装すればよい。しかし、一般に広く公開されたサーバプログラムを考えると、アクセスキー等を獲得するための認証プロトコルを公開する必要がある。公開サーバのアクセスキーが返送された場合、悪意を持ってサーバのデータ領域を破壊するクライアントプログラムが作成可能である。この問題を解決するのが同一ノード内のタスク間共有メモリであるクラスタ共有メモリ機構（**CSM**:Cluster Shared Memory）である☆。同一ノード（クラスタ）内ではページ管理機構を活用してページ単位で共有 / 非共有が大きなコスト増なしに設定可能である。そこで、クライアントごとにサーバはダミーのタスク（アドレス空間）を用意して、クライアントタスクにはそのダミータスクのアクセスキーを返答する。ダミータスクとサーバの間でサーバ内の該当クライアント用に用意したデータエリアのみを CSM でメモリ共有しておけば本来の MBCF とまったく同じ扱いが可能である。クライアントがカーネルの MBCF ルーチンまで偽造している可能性がある場合は、サーバ自体から発送する MBCF パケットからサーバのアクセスキーが洩れる

☆ CSM は UNIX System V 系の共有メモリ機構と同一である。

危険性がある。ここまで注意する必要がある場合は、クライアント宛の MBCF パケットの発送をサーバ自体は行わず、ダミーのタスクに肩代りさせてパケットを発送すればよい。

このダミータスクとクラスタ内メモリ共有によるサーバタスク保護方式のペナルティは、ダミーのタスクを生成することによりタスクの管理構造体のためのメモリを消費することと、TLB のエントリを区別するコンテクスト識別子を余分に消費することである。ただし、最近の計算機はメモリもコンテクスト識別子も十分にあり、実用に耐え得る。

5. タスク指定の仮想化に関して

汎用並列オペレーティングシステムでは、コンソールの使用やマシン特定の音源デバイスの使用といったタスクの位置が固定される要因がない場合、システム全体の負荷バランスをとるために、マイグレーション機能を持つことが望ましい。また、NOW 上の実行時間が長時間におよぶ並列数値計算シミュレーション等では実行途中で一部のマシンが shutdown されるかもしれない。この場合は shutdown 処理中に停止するマシン上のコンテクストをすべて他のマシンに移動 (マイグレーション) する必要がある。

このマイグレーションの必要性により、ユーザプログラム内の他タスクの指定方式は論理的な (仮想的な) ものにならざるを得ない。MBCF では論理タスク番号がユーザプログラム内で使用され、MBCF 専用システムコールにおいてタスクごとにカーネルが管理しいるタスク変換テーブルを参照して、ノード ID とタスク ID に変換される。プロセス全体や一部のタスクのマイグレーションが行われた (コンテクストが移送された) 場合、このタスク変換テーブルが更新されれば MBCF による通信を再開することができる。

プログラム実行ノードに依存がないアプリケーションではプログラムはすべて論理タスク番号のみで記述されている。並列プログラムで複数のタスクがプログラム起動時に OS によって用意される場合は、タスク変換テーブルの内容が OS によってセットされており、taskque への登録操作なしで MBCF による通信が可能である。サーバクライアントの分散処理やタスクを実行時に生成増殖させる並列処理の場合は、子タスク (クライアント) が親タスク (サーバ) の taskque にアクセスキー獲得メッセージを発行して、MBCF 通信を開始する。taskque への登録メッセージは、通常論理タスク番号が割り当てられる前に発行されるメッセージであり、ノード ID とタスク ID を明示的に指定する必要がある。taskque への登録メッセージ発行システムコールは引数として転送先タスク (taskque の owner) のノード ID とタスク ID を取り、転送先タスクの論理タスク番号を戻り値として返す。タスク変換テーブルに指定されたタスクが登録されていなければ、親タスク (サーバ) に対する論理タスク番号が新たに割り当てられる。この後の MBCF における親タスクの指定にはこの論理タスク番号が使用される。親タスク側 (受信側) では taskque にパケットが登録された時点で、パケットを送ってきたタスクに論理タスク番号を割り当て、taskque 読み出し時に、taskque への登録元の論理タスク番号を知る。

クライアントが適宜生成消滅する分散処理のようにタスク変換テーブルがアプリケーションによって、更新される必要がある場合は、マイグレーションはこのタスク変換テーブルの更新関連手続きを一時中断して行う必要がある。これを守らないと論理タスク番号とノード ID およびタスク ID の対応が正しくないエントリが登録される可能性がある。

6. おわりに

ソフトウェアメモリベース通信機能 (MBCF) は NOW 等の分散メモリ並列計算環境に特殊なハードウェア機構を付加せずに実装可能な高速かつ保護され仮想化されたユーザレベルの通信同期機構であり、しかも通信ハードウェアの方式や性能を問わない。MBCF は通信対象の論理アドレス空間のメモリ操作をアドレス範囲のチェックなしに行うために、危険度が高いように見える。しかし、使用方法の工夫により、同一アプリケーション内のアクティビティ間の保護が不要な並列処理のみならず、プログラムやデータの保護とセキュリティが重要になるサーバクライアントタイプの分散処理にも適用可能な能力を MBCF が持っていることを示した。これらの点から MBCF は汎用超並列超分散オペレーティングシステムがサポートするのに非常に適したユーザレベル通信同期方式である。

謝　辞

本研究は情報処理振興事業会 (IPA) が実施している独創的情報技術育成事業の一環として行なった。

参考文献

1) 松本 尚, 平木 敬: Memory-Based Processor による分散共有メモリ. 並列処理シンポジウム JSPP '93 論文集, pp.245–252 (May 1993).
2) 松本 尚, 平木 敬: 汎用並列オペレーティングシステム SSS–CORE の資源管理方式. 日本ソフトウェア科学会第 11 回大会論文集, pp.13–16 (October 1994).
3) 松本, 駒嵐, 渦原, 平木: メモリベース通信による非対称分散共有メモリ. コンピュータシステムシンポジウム論文集, 情報処理学会 pp.37–44 (November 1996).
4) T. von Eicken, D. E. Culler et al.: Active Messages: A Mechanism for Integrated Communication and Computation. *Proc. 19th Int. Symp. on Computer Architecture*, pp.256–266 (May 1992).
5) T. von Eicken, A. Basu, and V. Buch: Low-Latency Communication Over ATM Networks Using Active Messages. *IEEE Micro*, pp.46–53 (February 1995).