オブジェクト指向システム用軽量ケーパビリティプロテクション方式

山田茂樹† 丸山勝己†

オブジェクト指向システムの性能をできるだけ損なわずに強固なメモリプロテクションを行う軽量ケーパビリティプロテクション方式 (Lightweight Capability-Based Protection Scheme: LCP) を提案する．LCP は従来のケーパビリティ方式に対して以下の特徴を持つ．(1)仮想記憶方式をベースとするケーパビリティの限定適用：メモリプロテクション機構として効率の良い仮想記憶方式をベースに，バグ等により不正アクセスを行う可能性の高いアプリケーション (AP) スレッドと，不正アクセスを受ける可能性が高いメッセージバッファ (MB) にケーパビリティを限定適用する．(2)スレッド ID によるチェック：AP スレッドと MB との対応付けを，ケーパビリティ内に含ませたスレッド ID で行う．これにより，AP スレッドが切り替わる度にケーパビリティを入れ替えるだけで対応がとれる．(3)メモリアクセスとチェックの並行実行：メモリアクセスと並行してケーパビリティチェックを行い，チェックによるメモリアクセスタイムの増加を回避する．以上の処理オーバヘッドの少ない LCP について実現アルゴリズム，ハードウェア／ソフトウェアインプリメンテーション方法を述べる．また，性能および信頼性の評価を行い，実用的な一例では LCP を使用しない方式 (仮想記憶のみの方式) に比べて実行時オーバヘッドは2％増程度で，MB への不正アクセスはほぼ 100% 近く検出できることを述べている．

A Lightweight Capability-Based Protection Mechanism for Object-Oriented Systems

Shigeki Yamada† and Katsumi Maruyama†

This paper describes a Lightweight Capability-Based Protection Scheme, LCP, which needs only small execution overhead for object-oriented systems. The LCP is based on a virtual addressing mechanism, instead of a capability-based addressing mechanism, to reduce the overhead. In LCP, capabilities are applied only to memory accesses from application threads to message buffers in a shared memory space. The reason for the limited use of capabilities is that the application threads may sometimes include potential errors, leading to illegal memory accesses, while the message buffers in a shared memory space are vulnerable to illegal accesses, therefore, both necessitating a robust memory protection mechanism. Capabilities in LCP are assigned to both application threads and message buffers. An application thread's capability contains its own thread identifier, while a message buffer's capability contains the thread identifier of an application thread, given an access right to the message buffer. Every time an application thread makes an access to a message buffer, the two capabilities are compared to detect an illegal access. This capability comparison is performed in parallel with the memory access, resulting in little performance degradation. The LCP is implemented with 2% execution overhead increase while achieving almost 100% error detection under practical conditions.

1. はじめに

オブジェクト指向システムにおけるオブジェクト[1]は，一般にメソッドと呼ばれる手続きとデータの合併体として定義される．手続きは外部から送られてくるメッセージによって起動され，メッセージの内容とオブジェクトの記憶や状態に対応した計算を実行し，自分の記憶の更新，他のオブジェクトへのメッセージの送信を行う．オブジェクト指向システムは，オブジェクト間でメッセージを交換しながら処理を進めるシステムで，その実現技術としてハードウェアレベルからプログラミングレベルまでさまざまな手法が提案されている．

本論文では，特にメッセージを記憶するメモリに着目して，少ないオーバヘッドで強固なメモリプロテク

† NTT 交換システム研究所伝達ソフトウェア研究部
Transport Switching Software Laboratory, NTT Communication Switching Laboratories

ションを行う軽量ケーパビリティプロテクション方式(Lightweight Capability-Based Protection Scheme : LCP) を提案する．LCP の特徴は，以下の点にある．(1)仮想記憶方式をベースとするケーパビリティの限定適用：メモリプロテクション機構として効率の良い仮想記憶方式をベースに，バグ等により不正アクセスを行う可能性の高いアプリケーション (AP) スレッドと，不正アクセスを受ける可能性が高いメッセージバッファ (MB) にケーパビリティを限定適用する．(2)スレッド ID によるチェック：AP スレッドと MB との対応付けを，ケーパビリティ内に含ませたスレッド ID で行う．これにより，AP スレッドが切り替わる度にケーパビリティを入れ替えるだけで対応がとれる．(3)メモリアクセスとチェックの並行実行：メモリアクセスと並行してケーパビリティチェックを行い，チェックによるメモリアクセスタイムの増加を回避する．

本稿では，まず従来のプロテクション方式の問題点を述べ，これらと異なるアプローチにより解決を図った LCP について，要求条件，ハードウェア，ソフトウェアの実現法を説明する．最後に性能および信頼性の評価を行い，LCP がほぼ仮想記憶方式並みのオーバヘッドで強固なメモリプロテクションを実現できることを示す．

2. 従来のメモリプロテクション方式の問題点

メモリプロテクション機能，特にメインメモリのアクセス保護方式は，大別[3]して(1)領域レジスタ方式，(2)保護キー方式，(3)リング保護方式，(4)仮想記憶方式，(5)ケーパビリティアドレシング方式などに分類される．このうち，(1)，(2)はハードウェア中心の単純な手法であり，高度な情報処理システムに適用するには不十分な技術である．(3)〜(5)はソフトウェア，ハードウェアを組み合わせた高度なプロテクション方式であり，(3)は(4)と組み合わせて使用される場合が多い．ここでは特に最近のフォンノイマン型プロセッサで最も広く用いられている仮想記憶ページディスクリプタ方式と，高度なプロテクション機能を追求したケーパビリティアドレシング方式について考察する．

仮想記憶ページディスクリプタ方式は**図1**(1)に示すようにメモリの論理アドレスを物理アドレスに変換する際に，ページテーブルに記憶されたディスクリプタのアクセス権情報をチェックする方式である．高頻度に参照されるページテーブルはプロセッサ内の TLB (Translation Lookaside Buffer) に収容されるので，高速で処理オーバヘッドが少ないという利点がある．しかし，プロテクト単位がページ（例えば1〜4K バイト）という物理的単位に固定され，オブジェクト指向プログラムにとって望ましい小さな論理的単位（例えば 300〜500 バイト程度[4]）でのプロテクションを実現するのが困難である．

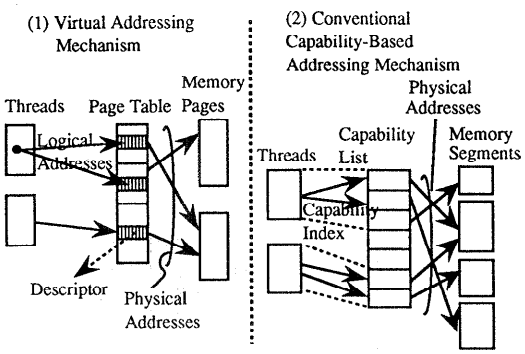

図 1 従来のメモリプロテクション方式の概念図
Fig. 1 Conventional memory protection mechanisms.

ケーパビリティアドレシング方式[7]は，図1(2)のように論理名（ケーパビリティインデクス）を物理アドレスに変換する点はページディスクリプタ方式と似ているが，ページ単位ではなく任意サイズのメモリセグメントを保護の単位にできる点，ケーパビリティ（アクセス権を表示したチケット情報）をスレッド（軽量プロセス）間でコピー/移譲できる点が異なり，仮想記憶ページディスクリプタ方式に比べてさらに高機能で柔軟なプロテクションが実現できる．

ケーパビリティアドレシング方式は，情報隠蔽やモジュール性を高めるオブジェクト指向の考え方と共通点が多い．そのためオブジェクト指向プロセッサ iAPX 432[6]のように，単にケーパビリティアドレシング機構のみならずオブジェクト指向に要求される各種のオペレーティングシステムの機能（例えばプロセス間通信，プロセススケジューリング機能，多種多様なオブジェクトのサポートなど）をハードウェアでサポートした斬新的なプロセッサアーキテクチャで実現される場合が多かった．このような高機能化，汎用化の代償として iAPX 432 の性能が VAX 11/780 の平均 1/20 に低下したり[6]，ケーパビリティを使ったプロシジャコールが VAX 11/780 の 12 倍のクロック数を要した[7]例等が報告されている．

そこで本論文では，汎用的なオブジェクト指向アーキテクチャにケーパビリティを組み込むアプローチではなく，ケーパビリティの思想を生かしながら機能・適用範囲を限定したアプローチをとることによって，ほぼ仮想記憶方式並みの処理オーバヘッドで強固なメモリプロテクションを実現する．

3. 軽量ケーパビリティプロテクション方式

3.1 メモリ空間構造

送信スレッドから受信スレッドへのメッセージ通信は，メッセージバッファ（MB）の記憶エリアを確保し，そこに送信スレッドが書き込み，次に受信スレッドが読み出すことで実現される．アプリケーションにも依存するが，MB サイズは例えば256バイト(B)程度を想定する．もし送信スレッドが直接アクセスできる MB と受信スレッドが直接アクセスできる MB とが異なるならば，2つの MB 間で何らかの形でテッセージの物理転送が必要となり，物理転送に伴うソフトウェア，ハードウェアのオーバヘッドを無視できなくなる．そこで送受スレッド間で物理的な転送をともなわず，MB ポインタの受け渡しのみでメッセージ通信が実現できるように構成する．具体的には**図2**に示すように，アプリケーション（AP）空間（ドメイン)/共有空間/カーネル空間の3階層論理空間制御を用いる[8]．図2においてドメインはサービス種類等に応じて複数個存在し，各種の AP スレッド（AP モードで走行する並列処理単位の軽量プロセス）が配置される．共有空間は各ドメインに共通の空間で，ここに MB を配置することによって，ドメイン上の各 AP スレッドから MB に直接読み書きすることができる．その結果，ドメイン内/ドメイン間いずれのメッセージ通信においても，カーネル空間を経由したメッセージのコピーが不要で効率良いメッセージ通信を実現できる．カーネル空間はプロセッサのカーネルモードでのみアクセス可能な空間で，PLATINA[8] のカーネル等が配置される．

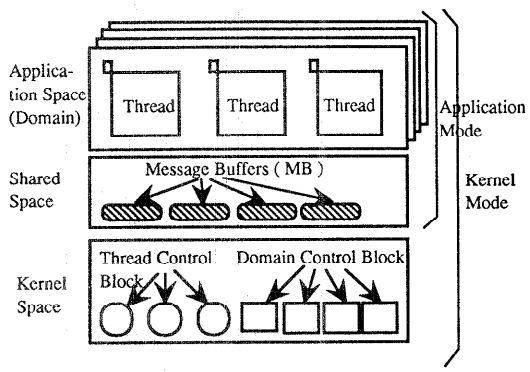

図 2 PLATINA の論理空間の構造
Fig. 2 Logical address space structure of PLATINA.

この論理空間構造では，共有空間に配置されたすべての MB が各ドメイン内の全 AP スレッドからのアクセスを物理的に許しているので，プログラムの潜在バグ等により，AP スレッドが，論理的にアクセスを許されていない MB に誤ってアクセスする可能性がある．そこで，共有空間上の MB を対象に以下の LCP 方式を適用する．

3.2 LCP の特徴

LCP の特徴は以下のとおりである．

（1）仮想記憶方式をベースとするケーパビリティの限定適用
（2）スレッド ID によるアクセス側と被アクセス側との対応付け
（3）メモリアクセスとケーパビリティチェックの並行実行

一般にケーパビリティプロテクションの処理オーバヘッドは，(a)スレッド切替えごとにケーパビリティのロード，ストア，複製，移譲等を行うための［ケーパビリティ操作オーバヘッド］と，(b)メモリセグメントへのアクセスごとにケーパビリティの内容をチェックする［ケーパビリティチェックオーバヘッド］とに分類される．(1)，(2)はケーパビリティ操作オーバヘッドを軽くし，(3)はケーパビリティチェックオーバヘッドを軽減する．

3.3 仮想記憶方式をベースとするケーパビリティの限定適用

従来のようにプロセッサアーキテクチャにケーパビリティアドレシング方式を取り入れると，オーバヘッドが大きくなるので，元来オーバヘッドの少ない仮想記憶ページディスクリプタ方式をベースに，重点的保護が必要な部分にのみケーパビリティを限定適用する．ケーパビリティ適用対象として，(a)不正アクセスを行う可能性が高いスレッド（=AP スレッド：PLATINA が想定する実行環境において，頻繁な追加，変更が行われて潜在バグが混入しやすいスレッド）と，(b)不正アクセスを受けやすい保護対象メモリセグメント（=MB：共有空間に配置され，AP スレッドから直接アクセス可能なメモリセグメント）の組み合わせに対してのみ，3.4節/3.5節で述べるケーパビリティの操作/チェックを実行する．

オブジェクト指向のメッセージ通信では送信側と受信側との間で1対1のメッセージ通信が多い．そこで，各 MB へのアクセス権は，同時にはただ1つのスレッドが持つように制限する．すなわち送信スレッドと受信スレッドは同時に同じ MB へのアクセス権を持つことはなく，排他的に持つ．また1つのスレッドは複数の MB のアクセス権を同時に所有できるが，各 MB は他のスレッドに同時に所有されることはない．

3.4 スレッド ID によるアクセス側と被アクセス側との対応付け

図2の仮想記憶方式の共用空間のページテーブルは，各ドメインに共用され，各ドメイン内のどのスレッドからもアクセスしてもページテーブルのディスクリプタのアクセス権（読み/書き/実行権）チェックを通り抜けてしまう．したがって仮想記憶方式のみでは，共有空間内の特定の MB が特定のスレッドからのアクセスを許すというような選択的制御を行うことができない．これに対処するために，どの AP スレッドがどの MB をアクセスできるかという対応付けを，アクセスする側の識別情報（スレッド ID）を用いて行う．このスレッド ID とアクセス権を合わせたものをケーパビリティと呼ぶ．処理の流れを**図3**に示す．

[1]実行すべき AP スレッドが決まると，そのスレッド ID が Current Capability Register(CCR) に記憶される．一方，それとは独立に，ある MB へのアクセス権が，ある AP スレッドに与えられると，そのスレッドの ID が MB 対応の Memory Capability Register(MCR) に登録される．

[2]この状態で，AP スレッドが MB をアクセスすると，図3のようにまず論理アドレスがページテーブルで物理アドレスに変換される．これが共有空間内のアドレスならば，このアドレスを検索キーに，MB に対応する MCR 内のケーパビリティを取り出す．（物理アドレスをもとに複数の MCR の中から，所望の MCR を取り出すアルゴリズムは4章で説明する．）

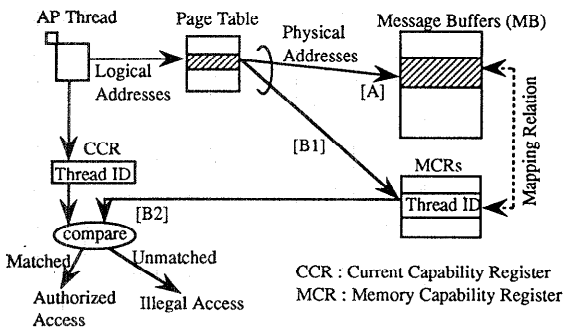

図3 軽量ケーパビリティプロテクション方式 (LCP)
Fig. 3 Lightweight capability protecion scheme (LCP).

[3]MCR 内のケーパビリティを CCR 内のケーパビリティと比較し，両者のスレッド ID が一致すれば，正しいアクセス，不一致であれば不正なアクセスとみなす．

[4]物理アドレスが共有空間外であれば通常の仮想記憶方式によるページ単位のアクセス権チェックを行う．

この方式によれば，スレッドの切り替え時に CCR のスレッド ID の値を更新するだけで，AP スレッドからアクセスできる MB のすべてを1個のスレッド ID で一括指定できるので，ケーパビリティ操作オーバヘッドを軽減することができる．

3.5 メモリアクセスとケーパビリティチェックの並行実行

図3で物理アドレスにより MB へアクセスするメモリアクセス操作[A]と，MB に対応する MCR を求め，CCR と比較する操作 [B1] および [B2] とを並列実行する．これらの実行時間をそれぞれ T_A, T_{B1}, T_{B2} とした場合，$T_A \geq T_{B1}+T_{B2}$ ならば，ケーパビリティチェックオーバヘッド＝0 となり，性能低下は生じない．また，$T_A < T_{B1}+T_{B2}$ の場合にも，並列実行により，性能低下を最小限に抑えることができる．

4. LCP のインプリメンテーション

4.1 MB の前提条件

MB 物理アドレスをもとに複数の MCR から，目的とする MCR を見つけ出す操作を容易化するため，MB に以下のような制限を与える．メモリ空間はバイトアドレシング構成で，アドレスサイズは a ビットとする．

（1） **図4**（1）に示すように，MB 群を収容する共有空間は全体で 2^a バイトで，共有空間はその先頭アドレスの下位 x ビットが0となるように配置される．

（2） MB サイズ＝2^n バイト（$n=u,u+1,\cdots,s$; u,s はそれぞれ最小 MB サイズ，最大 MB サイズに対応する値）に規格化し，MB の先頭アドレスは下位 n ビットが0となるように配置する．この制限内で各種のサイズの MB が共有空間内に混在することを許す．

MB の物理アドレスから，対応する MCR を見つけ

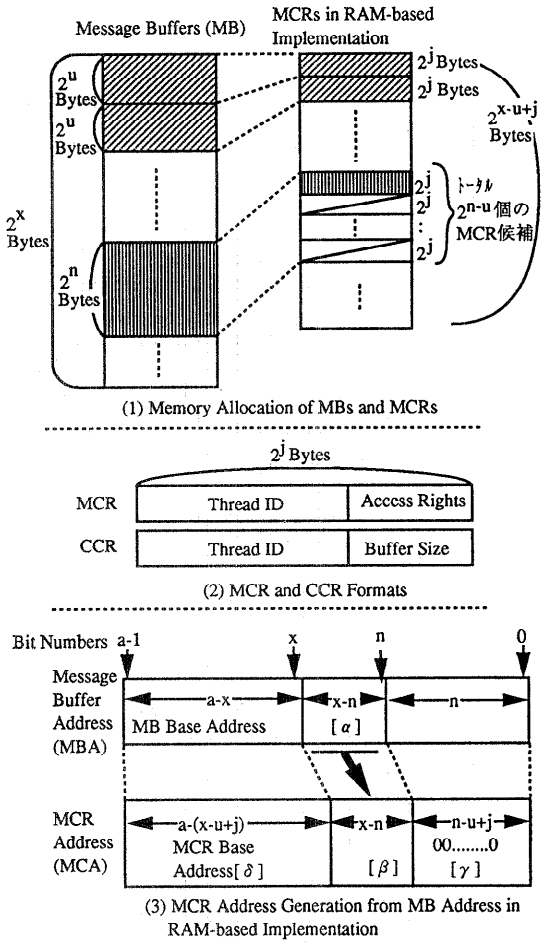

図 4 RAM 方式による MCR 選択方法
Fig. 4 RAM-based scheme to select an MCR corresponding to a message buffer.

出す方法はハードウェアインプリメンテーブションに依存するので，以下に2つのインプリメンテーションを説明する．なお，上記の制限条件(1)，(2)は2つのインプリメンテーションに共通である．

4.2 RAM (Random Access Memory) 型のインプリメンテーション

MCR 空間（MCR をアドレスが連続するように並べて構成した空間）をメインメモリと同様，RAM で構成する方法である．図4(1)に示すように MCR 1個のサイズを 2^j バイトとし，各 MCR は MB 最小サイズ（$=2^u$ バイト）に対応してメモリを用意するので，MCR 空間のサイズは 2^{x-u+j} バイトとなる．また，MCR 空間は先頭アドレスの下位 $(x-u+j)$ ビットが0となるように配置する．ここで 2^n バイト $(n \geq u)$ の MB では 2^{n-u} 個の MCR 候補エリアがあるが，このうち最若番地（＝下位 $n-u+j$ ビットが0）の MCR 候補のみを用い，残りの MCR 候補は使用しない．

MCR は図4(2)のように，(1)MB へのアクセスが許された AP スレッドのスレッド ID と(2)アクセス権情報（読み出し権，書込み権等の表示ビット）とを含む．CCR は(1)実行中の AP スレッドのスレッド ID と，(2)各種のサイズの MB に対処するためのバッファサイズ情報（n）とを含む．MCR，CCR に値が設定された状態で 2^n バイトの MB へのアクセスが生じると，図4(3)並びに以下の手順で MB アドレス（MBA）から MCR アドレス（MCA）が生成される．（表記法として，$A_{i:j}$ はアドレスAの第 i ビット目から第 j ビット目まで $(j \geq i)$ を表す．）

(1) $MBA_{n,x-1} \rightarrow MCA_{n-u+j,x-u+j-1}$：図4(3)の，MBA のフィールド[$\alpha$]を切り出し，MCA のフィールド[$\beta$]に埋め込む．4.1節の MB 配置条件により，$2^n$ バイトの MB 内の各エリアのアドレスはフィールド[α]の値が同じになるので，これを MCR アドレスの一部に流用することを意味する．

(2) all $0 \rightarrow MCA_{0,n-u+j-1}$：図4(3)のフィールド[$\gamma$]に all 0 を埋め込む．フィールド[$\gamma$]は $(n-u+j)$ ビットからなるが，その上位 $n-u$ ビットの all 0 は，2^{n-u} 個の MCR 候補のうち最若番地の MCR を選択することを意味し，残りの下位 j ビットの all 0は，2^j バイトのサイズの MCR の先頭番地を指定することを意味する．

(3) MCR Base Address $\rightarrow MCA_{x-u+j,x-1}$：図4(3)のフィールド[$\delta$]の部分に MCR ベースアドレス（固定値）を埋め込む．

以上で 2^n バイトの MB に対応する MCR アドレスを一意に求めることができる．ここで a, x, u, j は設計時に決まる定数，n は扱う MB に依存する変数で，その値は CCR で与えられる．上記のフィールド切り出し/埋め込み操作をハードウェアで行えば，MCR アドレスを高速に生成することができる．そのハードウェア構成を**図5**に示す．CCR，MCR およびそれらの比較回路一式がメインメモリの外付け回路として構成される．なおシングルプロセッサの場合，同時に実行される AP スレッドは1つなので図5に示すように CCR は1個のみ用意し，AP スレッドの実行に合わせてメモリライト命令で値を入れ替える．

RAM 型では低コストと高速性を実現できるが，1

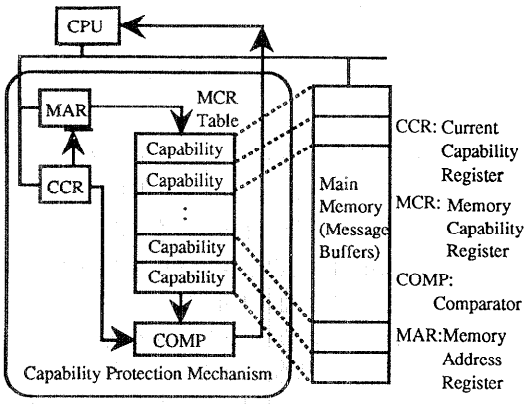

図 5 RAM 型の実現法
Fig. 5 RAM-based implementation.

つの MB が一時に1個の AP スレッドに所有されるという制限がつく．しかし，オブジェクト指向システムのメッセージ通信では，1つの MB が Sender または Receiver のいずれか一方のスレッドに所有される使い方が多いので，RAM 型はオブジェクト指向システムに適したインプリメンテーションといえる．

4.3 CAM (Content Addressable Memory) 型のインプリメンテーション

CAM 型は図6に示すように MCR 群を CAM（連想メモリ）で実現する方法である．CAM は検索キーデータと各 CAM ワード（CAM セル）とを比較して一致したワードのアドレスを出力する．検索キーデータ（KD）としては，MB 先頭アドレスおよびスレッド ID を用いる．MB 先頭アドレスを KD に用いた理由は MB 内のどのエリアにアクセスしても，以下のように MB アドレスから MB 先頭アドレスを求

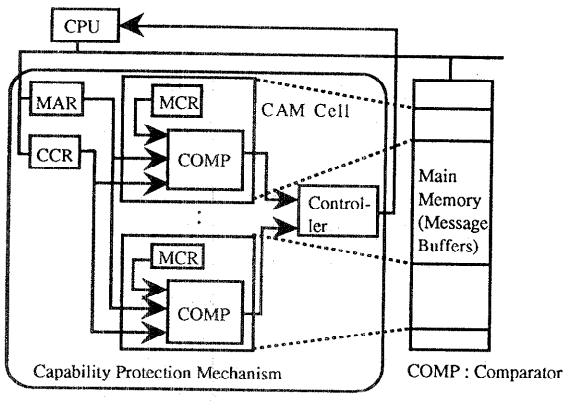

図 6 CAM 型の実現法
Fig. 6 CAM-based implementation.

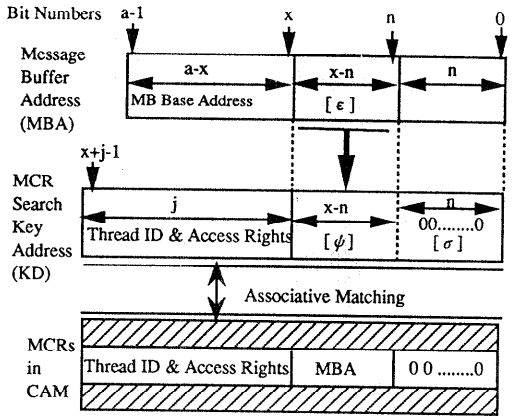

図 7 CAM 方式で Message Buffer Address から対応する MCR を求める方法
Fgi. 7 CAM-based scheme to select an MCR corresponding to a message buffer.

めるのが容易なためである．また，KD として MB 先頭アドレスのほかにスレッド ID を用いる理由は，MB が複数のスレッドに共有されている場合，どのスレッド対応の MCR かを識別するためである．KD の生成手順を図7ならびに以下に示す．

（1）$MBA_{n,x-1} \rightarrow KD_{n,x-1}$：図7のフィールド［ε］を KD のフィールド［ψ］に埋め込む．これは図4のフィールド［β］と同様，2^n バイトの MB の共通なアドレス部を抜き出すことを意味する．

（2）all $0 \rightarrow KD_{0,n-1}$：図7のフィールド［σ］に0を埋め込む．2^n のサイズの MB 内のどのエリアにアクセスしても，その MB の先頭アドレス（下 n ビットが0）の値が生成されることを意味する．

（3）CCR の内容（スレッド ID，アクセス権ビット）$\rightarrow KD_{x,x+j-1}$（j＝スレッド ID 長＋アクセス権ビット長）

KD を CAM に入力して検索し，一致した MCR があれば，正しいアクセスであると判断する．CAM 型では複数の MCR との一致も検出できるので1個の MB に複数の MCR（すなわち，複数のスレッド）を対応させるなど，将来の機能拡張にも柔軟に対処できる．

4.4 ソフトウェアのインプリメンテーション

図8のソフトウェア制御シーケンスは，送信側 AP スレッド（図8の Sender）から受信側 AP スレッド（Receiver）に1対1通信でケーパビリティを移譲するとともに共有空間上の1つの MB を介して非同期型メッセージ通信が行われる様子を示したもので，

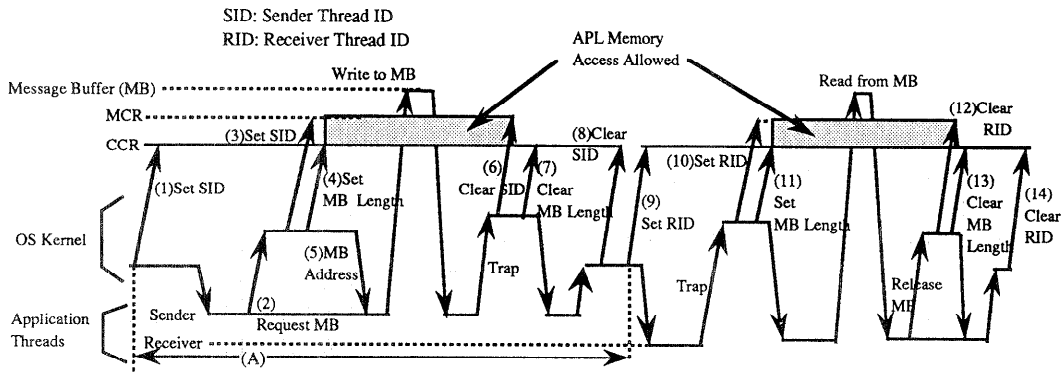

図 8 ソフトウェアから見た軽量ケーパビリティプロテクション (LCP) 制御シーケンス
Fig. 8 LCP program control sequence.

RAM 型/CAM 型共通のシーケンスである．(以下の ( ) 内の番号は図8の番号に対応する.)

(1)カーネルは実行すべきスレッドを決定し，そのスレッド ID(SID) を CCR に設定する．(2) Sender は送信したいメッセージの準備ができると，カーネルに MB 捕捉要求を出す．(3)，(4)カーネルは空き MB を捕捉し，MCR には SID を，CCR にはバッファサイズ (=n) を登録する．以上で CCR，MCR に制御情報が揃ったので，(4)の時点以降，プロテクションウィンドゥが開かれ，識別番号 SID を有する Sender だけが当該 MB へのアクセスを許される．(5)カーネルは Sender に MB アドレスを通知する．(6)，(7)，(8) Sender が MB へのメッセージ書込みを終了すると，カーネルにメッセージ転送を依頼する．カーネルは MCR，CCR をクリアしてプロテクションウィンドウを閉じるとともにメッセージのヘッダを参照して Receiver のメッセージキューにメッセージを接続する．(9) Receiver が実行され，メッセージ受信要求のトラップが発生するとカーネルは Receiver のメッセージキューにメッセージが届いているかどうかを確認し，届いていれば CCR にスレッド識別番号 RID を設定する．(10)カーネルが MCR に受信側のスレッド ID(RID) をセットする．(3)から(10)への状態変化は当該 MB のケーパビリティが Sender から Receiver に移譲されたことを意味する．以後の(11)～(14)では Sender の場合と同様のシーケンスでプロテクションウィンドゥの開閉を行う．プロテクションウィンドゥが閉じている期間は，すべての AP スレッドからのアクセスができず，カーネルからのアクセスのみ許されるので MB に対して非常に強固なプロテクションを実現することができる．

5. LCP の評価

5.1 MCR 容量

LCP のハードウェア量の大部分は MCR によって占められるので，必要な MCR 容量を評価する．パラ

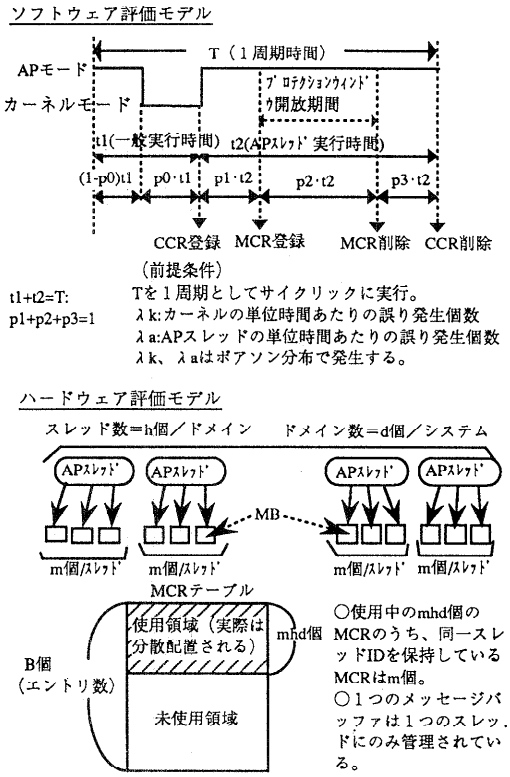

図 9 LCP 評価モデルと前提条件
Fig. 9 LCP hardware and software models.

メータは以下のとおりとする．

RAM 型の場合，スレッド ID とアクセス権ビットの合計で MCR サイズ＝4 バイト（$j=2$），MB 空間サイズ＝2 MB（$x=21$），MB 最小サイズ＝64 バイト（$u=6$），MB 最大サイズ＝1,024 バイト（$s=10$）とすれば，MCR 最大数＝2^{x-u}＝32 K（キロ）個となる．したがって必要な MCR 容量＝32 K×4 バイト＝128 Kバイト程度であり，十分経済的に実現できる範囲にある．

CAM 型の場合，MCR のサイズ＝RAM 型の MCR サイズ（4 バイト）＋MB 空間ビット長（$x=21$ ビット）＝53 ビットとして MCR 容量＝32 K×53＝1.7 M ビット程度となる．CAMLSI 容量を 20 kb/チップとすれば，85 チップ/システムで，十分実現可能な範囲にある．

5.2 処理オーバヘッド

LCP を採用した場合のメッセージ通信におけるケーパビリティ操作オーバヘッドを，採用しなかった場合と比較して評価する．LCP により純粋に増加するステップは図 8 のステップ（1），（3），（4），（6）～(14)である．このうち，Sender，Receiver はほぼ同じと見なして Sender 側（（1），（3），（4），（6）～（8））のみを考える．MC 68030 プロセッサでステップ（3）は 8 命令，残りのステップは各 2 命令程度で実現でき，合計 18 命令がケーパビリティ操作オーバヘッドとなる．ケーパビリティを採用しない場合，カーネルの実行も含めた 1 つのスレッドの実行（図 8 の区間（A））に平均 1,000 命令（LCP の適用を想定しているディジタル交換機における経験的な値）を要するとすると，提案した方式の採用による操作オーバヘッド増分は 18/1000＝1.8% と非常に少ない．

チェックオーバヘッドについては現在試作中の RAM 型で，メインメモリ用 RAM よりも高速な RAM チップを MCR として使用することにより，チェックオーバヘッドを 0 にする予定である．

5.3 システム信頼性の評価

LCP のシステム信頼性に対する効果を評価するために，不正アクセスが MB に対して行われた場合に，

表 1 エラー発生パタンとエラー検出個数との関係
Table 1 Error generating conditions and detectable errors.

区間	エラー検出可否（○：可，×：不可）	場合分け時の確率	区間内のエラー発生個数	区間内のエラー検出個数
$(1-p0)t1$	（1） ○（CCR 未設定により検出可）	1	$\lambda a(1-p0)t1$	$\lambda a(1-p0)t1$
$p0t1$	（2） ×（カーネル実行時はチェックしないため）	1	$\lambda k \cdot p0 \cdot t1$	0
$p1t2$	（3） IF アクセスアドレスが正しい THEN ○（MCR 未設定により検出可）	$1/B$	$\lambda a \cdot p1 \cdot t2$	$\lambda a \cdot p1 \cdot t2/B$
	IF アクセスアドレスが正しくない THEN			
	（4） IF MCR＝CCR THEN ×（エラーとして識別不可）	$(m-1)/B$		0
	IF MCR≠CCR または MCR not exist THEN （5） ○（スレッド ID 不一致または MCR 未登録で検出）	$(B-m)/B$		$\lambda a \cdot p1 \cdot t2(B-m)/B$
$p2t2$	（6） IF アクセスアドレスが正しい THEN ×（CCR，MCR 一致により検出不可）	$1/B$	$\lambda a \cdot p2 \cdot t2$	0
	IF アクセスアドレスが正しくない THEN			
	（7） IF MCR＝CCR THEN ×（エラーとして識別不可）	$(m-1)/B$		0
	（8） IF MCR≠CCR または MCR not exist THEN ○（スレッド ID 不一致または MCR 未登録で検出）	$(B-m)/B$		$\lambda a \cdot p2 \cdot t2(B-m)/B$
$p3t2$	（9） IF アクセスアドレスが正しい THEN ○（MCR 未設定により検出可）	$1/B$	$\lambda a \cdot p3 \cdot t2$	$\lambda a \cdot p3 \cdot t2/B$
	IF アクセスアドレスが正しくない THEN			
	（10） IF MCR＝CCR THEN ×（エラーとして識別不可）	$(m-1)/B$		0
	（11） IF MCR≠CCR または MCR not exist THEN ○（スレッド ID 不一致または MCR 未登録で検出）	$(B-m)/B$		$\lambda a \cdot p3 \cdot t2(B-m)/B$
合計 T	合 計		$G=\lambda a((1-p0)t1+t2)+\lambda k \cdot p0 \cdot t1$	$D=\lambda a((1-p0)t1+((p1+p3)(B-m+1)+(p2(B-m))t2/B)$

これがLCPによって検出できる条件付き確率（エラー検出率）を求める．**図9**にLCPのエラー検出率を求めるためのソフト，ハード評価モデルと前提条件を示す．ソフトウェアモデルは図9のシーケンスに基づく．付録1にその詳細を示す．MBにアクセスするAPスレッドの実行時間と，それ以外の一般実行時間との比を$t2:t1$に，一般実行時間の中でカーネルモードとAPモードの実行時間比を$p0:(1-p0)$に設定し，これらの比率を変えて種々の特性のプログラムをモデル化する．ソフトウェアモデルの各タイミング期間においてエラーが発生すると，それが不正なメモリアドレスを生成してMBの各領域を均等にアクセスすると仮定する．この不正アクセスをLCPで検出できるかどうかを分類したものが**表1**である．LCPで検出できないエラーは以下のようなケース（表1で×が表示されている区間）で発生する．

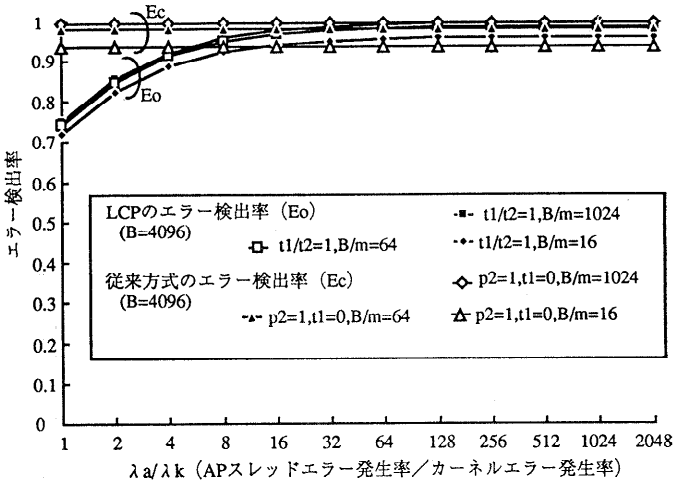

図10 APスレッドエラー発生率（相対値）とエラー検出率との関係
Fig. 10 AP thread error rate (relative value) vs. error detection probability.

（a） カーネルが暴走し不正なメモリアドレスのアクセスを行った場合：カーネルモードではチェックを行わないため，エラー検出ができない（表1(2)）．

（b） APスレッドが暴走し不正なメモリアクセス（他人のMBへのアクセス）を行ったが，それに対応するMCRとCCRのスレッドIDが偶然一致した場合：不正アクセスされたMBの所有者と本来アクセスされるべきMBの所有者が偶然同一だった場合に生ずる．LCPでは正常と判定してしまい検出不可（表1(4)，(7)，(10)）．

（c） APスレッドがプロテクションウィンドゥオープン中に暴走したが，不正メモリアクセス先が偶然，ウィンドゥオープン中のMB（自分のMB）であった場合：LCPでは正常と判定してしまい検出不可（表1(6)）．

表1の場合分けに基きづ，LCPのエラー検出確率$=D/G$（ただし$G=1$周期期間T中の総エラー発生個数，$D=$そのうちLCPで検出できるエラー個数，詳細は表1参照）として求める．また，付録2に示すように，図9と表1において$t1=p1=p3=0$，$p2=1$とした場合は従来のケーパビリティアドレシング方式と見なすことができるので，これらのパラメータ値を選択することにより，従来のケーパビリティアドレシング方式のエラー検出確率E_cも求めることができる．

なお，PLATINAでは共有空間以外にAP空間やカーネル空間を有し，AP空間とカーネル空間は従来の仮想記憶方式でプロテクトされる．したがってシステム全体のエラー検出率は，AP/共有/カーネルの各空間サイズに応じて，2つの方式（提案方式と仮想記憶方式）の重み付けにより求められる．

5.4 数値結果と考察

図10に数値結果の一例を示す．横軸にAPプログラムとカーネルのエラー発生率の比λ_a/λ_kを，縦軸にLCPのエラー検出率E_0と従来のケーパビリティアドレシング方式のエラー検出確率E_cをプロットした．共有空間内に収容されるMBの数（$=$MCRの数）は，$B=4,096$と仮定する．従来のケーパビリティアドレシング方式では，APモード，カーネルモードいずれの場合にもケーパビリティプロテクション機構が働くので両モードのエラー発生率の比λ_a/λ_kが変わっても，その値に関係なく，エラー検出確率E_cは一定である．

一方，LCPはAPモードでのみプロテクション機構が働くので，APモードでのエラー発生相対比率(λ_a/λ_k)が大きいとエラー検出確率も向上する．例えば図10で$\lambda_a/\lambda_k=128$すなわちAPスレッドがカーネルの128倍のエラーを発生するとLCPでは$E_0=99$％以上のエラー検出確率を有する．またMCRが大きくなると，スレッドIDの偶然の一致によるエラー見逃し頻度が低く抑えられるためにエラー検出確率も

向上する．

LCP の適用を想定しているディジタル交換機においては，通信サービスの頻繁な追加，変更により，AP プログラムの追加，変更量が非常に多く，AP スレッドのエラー発生率がカーネルのエラー発生率より圧倒的に高い．また，多重処理により MB の数も非常に大きいことから，ディジタル交換機に LCP を適用すると共有空間への不正アクセスを実用的なレベルでほぼ 100% 近く検出できるものと考えられる．

6. おわりに

PLATINA の内部構造の特徴の1つである共有空間に対して重点的にケーパビリティプロテクションの考えを取り入れ，処理オーバヘッドが少なく，仮想記憶方式と組み合わせて使用できる軽量ケーパビリティプロテクション方式 LCP を提案し，その実現方法と性能，信頼性の評価を行った．

提案した LCP ではスレッド切替時に CCR のスレッド ID の値を切り替えているが，スレッド ID の代わりにドメイン ID の値を CCR に設定することによりドメイン間のプロテクションに適用する等，LCP は各種のプロテクション対象に幅広く適用可能である．今後は，具体的なソフトウェア，ハードウェアの試作設計を通して LCP の有効性の確認を行うとともに，提案方式によるケーパビリティの分散環境上での管理法等，残された課題を検討する予定である．

謝辞 本研究の機会を与えてくださった当研究所伝達システム研究部富田修二部長，有益な助言をいただいた久保田稔主幹研究員，大崎憲嗣社員，性能評価にご協力いただいた田中聡研究主任に深謝いたします．

参考文献

1) Cox, B. J.: Message/Object Programming: An Evolutionary Change in Programming Technology, *IEEE Software*, Vol. 1, No. 1, pp. 50-61 (1984).
2) 山田，丸山：オブジェクト指向交換プログラム向きのケーパビリティプロテクション方式，信学技報，SSE 92-12, pp. 37-42 (1992).
3) 猪瀬 博ほか：コンピュータシステムの高信頼化，pp. 220-224，オーム社 (1977).
4) Kaiser, R.: MUTABOR, A Coprocessor Supporting Memory Management in an Object-Oriented Architecture, *IEEE Micro*, Vol. 21, No. 10, pp. 30-46 (1988).
5) Corsini, P., Frosini, G. and Lopriore, L.: The Implementation of Abstarct Objects in a Capability Based Addressing Architecture, *The Computer Journal*, Vol. 27, No. 2, pp. 127-134 (1984).
6) Hansen, P., Linton, M., Mayo, R., Murphy, M and Patterson, D.: A Performance Evaluation of The Intel iAPX 432, *Computer Architecture News*, Vol. 10, No. 4, pp. 17-26 (1982).
7) Gehringer, E. F. and Colwell, R. P.: Fast Object-Oriented Procedure Calls: Lessons from the Intel 432, *13th Intemational Symposium on Computer Architecture*, pp. 92-101 (1986).
8) Kubota, M., Maruyama, K., Tanaka, S., Osaki, K. and Yamada, S.: Distributed Processing Platform for Switching Systems: PLATINA, *Proceedings of ISS, Yokohama*, pp. 415-419 (1992).

付録1（図8と図9の相違点）

図8の送信側と受信側は基本的に同一シーケンスなので送信側のみ取り出す．次に図8のステップ(3)と(4)，(6)と(7)はそれぞれ隣接するステップなので，同じ時期として縮退させる．その結果，図9のソフトウェア評価モデルが得られる．

付録2（従来のケーパビリティアドレシング方式の評価モデル）

図9において従来方式で以下のパラメータ値を設定すればよい．

(1) AP モード，カーネルモードいずれの場合もプロテクション機構が働くので，プロテクション機構の動作しない期間 $t1=0$ となる．

(2) スレッドの実行前にケーパビリティの退避，実行後にケーパビリティの回復をそれぞれまとめて実行するので図9の $p1t2$, $p3t2$ のようにケーパビリティの値が一部未設定な期間が存在せず，$p1=p3=0$, $p2=1$ となる．

(3) B は LCP では MCR 最大エントリ数であったが，従来方式では共有空間の最大 MB 数となる．

なお，従来のケーパビリティアドレシング方式においても LCP と同様，ケーパビリティが指定している範囲のアドレスのメモリエリアをアクセスする限りエラーは検出されない．

(平成4年7月15日受付)
(平成5年6月17日採録)

山田 茂樹（正会員）

昭和47年北海道大学工学部電子工学科卒業．昭和49年同大大学院修士課程修了．平成3年工学博士．昭和49年 NTT 武蔵野電気通信研究所入所．以来，電子交換機用プロセッサ，データフロー制御交換システム，オブジェクト指向システム等の研究に従事．昭和56～57年にカリフォルニア大学ロサンゼルス校客員研究員．現在，NTT 交換システム研究所伝達ソフトウェア研究部主幹研究員．電子情報通信学会員．

丸山 勝己（正会員）

1944年生．1968年東京大学工学部電子工学科卒業．1970年同大大学院修士課程修了．同年日本電信電話公社入社．現在 NTT 交換システム研究所勤務．電子交換機の実時間増設方式，高水準言語と最適化コンパイラ，オブジェクト指向プログラミング，交換プログラム構造，通信網用分散処理などの研究実用化に従事．1985～88年 CCITT 第X研究委員会副議長．著書「交換用プログラミング言語 CHILL」（電気通信協会）．工学博士（東京大学）．1990年度情報処理学会論文賞受賞．電子情報通信学会員．