システムソフトウェアと
オペレーティング・システム 73－1
（1996．8．26）

リアルタイムOSにおける単一仮想空間のマッピング手法

河野通宗　　前沢敏行　　安西祐一郎

慶應義塾大学 計算機科学専攻

E-mail: {kohno,toshi,anzai}@aa.cs.keio.ac.jp

リアルタイムOSに適した、単一アドレス空間のマッピング機構を提案、実装する。このマッピング機構は、仮想ページの保護情報の変更に要するオーバヘッドが総物理メモリ量にもタスク数にも依存しないため、時間的オーバヘッドを$O(1)$に規定できる。さらにページテーブルを格納する空間的オーバヘッドを単純な線形マッピングに比べはるかに小さく抑えることができる。上記の提案を検証するため、ロボット用OS *μ-PULSER* に単一アドレス空間システムを実装した。評価の結果、実行中のタスク数に依存しないスレッド生成時間が得られたことがわかった。さらにスレッド間同期の処理時間が一定であり、ロボット用オペレーティングシステムに十分な時間予測可能性が得られたことがわかった。

Address Translation Scheme of Single Address Space System for Real-Time Operating Systems

Michimune Kohno　　Toshiyuki Maezawa　　Yuichiro Anzai

Department of Computer Science, Keio University

This paper proposes a new address translation mechanism that is suitable for real-time operating systems such as one for autonomous robots. This mechanism provides a fully-protected, time-boundable page translation scheme. The overhead for virtual address translation does not depend on neither total amount of physical memories nor the number of running tasks. This paper also describes the implementation of the single address space system on *μ-PULSER*, a real-time operating system which is to be used for autonomous mobile robots. In order to evaluate this system, several kinds of overhead were measured. Then the result proved better performance when the number of tasks was large. It was clear that the cost of updating page table is constant in this translation mechanism, and the predictability of entire system was improved significantly.

1　はじめに

パーソナルロボット用オペレーティングシステム *μ-PULSER*[1, 2] を単一アドレス空間システムに変更する。単一アドレス空間システムではアドレス空間の切り替えが発生しないため、コンテキスト切替えの際に、カーネルがキャッシュとTLBエントリをフラッシュする必要がない。そのためキャッシュ

ミスによる時間予測可能性の低下を抑えることができる。しかし、ソフトウェアエミュレーションではタスクごとのメモリ保護の実装が難しく、空間的・時間的オーバヘッドが生じる。

本稿ではリアルタイムシステムに適した単一アドレス変換のためのマッピング機構を提案する。このマッピング機構は、仮想ページの保護情報の変更に要するオーバヘッドが総物理メモリ量にもタスク数にも依存しないため、時間的オーバヘッドを規定できる。この機構により、タスクに割り当てられたメモリ領域を保護し、かつ通信時の時間予測可能性を向上させることができる。さらにページテーブルを格納する空間的オーバヘッドは、単純な線形マッピングに比べはるかに少ない。

上記の提案を検証するため、*μ-PULSER*に単一アドレス空間システムを実装した。実装対象がSPARCアーキテクチャであるため、レジスタウィンドウをシフトさせないフラットモデルを採用し、平均的な時間的オーバヘッドの増加よりもトラップによる予測可能性の低下を避けた。実装したシステムの性能を評価したところ、実行中のタスク数に依存しないスレッド生成時間が得られたことがわかった。さらにスレッド間同期の処理時間が一定であり、ロボット用オペレーティングシステムに十分な時間予測可能性が得られたことがわかった。

次節以降、本稿では次のような構成をとる。まず次節においてこれまでの*μ-PULSER*の機能を概観し、本稿で解決する問題点を明らかにする。そして第3節において単一アドレス空間システムの設計および実装について述べ、第4節では実装したシステムの、スレッド操作に関する基本的性能と、時間予測可能性がどの程度改善したかを評価する。最後にまとめと今後の課題を述べる。

2 リアルタイムOSのメモリ管理

2.1 パーソナルロボット用OS *μ-PULSER*

我々の開発しているオペレーティングシステム*μ-PULSER*は、外界からの入力に対して機敏に反応できるパーソナルロボット用ハードウェアアーキテクチャ*ASPIRE*[3]をサポートするオペレーティングシステムである。スレッド間の同期機構とコンテキスト切替え機構を統合したコンカレントインタラプト (Concurrent Interrupt — CI) 機構と、その信号により駆動されるCI駆動スケジューラを持つ。さらにリアルタイムスケジューラを実装しており、時間制約を持つスレッドを正しく実行できる。各タスクはそれぞれ別のアドレス空間上で実行される。各種サービス要求は、サーバに対するメッセージパッシングで行なわれる。

2.2 複数アドレス空間の問題点

複数アドレス空間によるタスク管理は、タスクごとのメモリを保護することに重要な役割を果たす反面、一般にリアルタイムシステムにおける予測可能性を低下させる。この要因として2点挙げる。1点目は、アドレス空間を切り替える時にTLBキャッシュをすべてフラッシュする必要があることである。TLBエントリは仮想アドレスから物理アドレスへの変換テーブルのキャッシュであるので、アドレス空間の切り替え時にはすべてのエントリをフラッシュしなければならない。このため、切り替わった後のタスクはアドレス変換テーブルの検索をすべてのアドレスについて行なうことになり、プロセッサはその度に実行をストールさせる。2点目は、ある一つのキャッシュエントリ (インストラクション、データ双方) に複数の仮想アドレスが割り当てられる可能性があるため、アドレス空間を切り替える時にはキャッシュをすべてフラッシュしなければならないことである。このことはキャッシュミスの割合を増大させ、実行時間の予測を困難にする。特にマイクロカーネル構造のOSでは、ユーザタスクのサービス要求の度にアドレス空間が切り替わる。このように、アドレス空間を越えるコンテキスト切り替えが、リアルタイムシステムにおいて重要な予測可能性を低下させている。

2.3 単一アドレス空間システム

近年、単一アドレス空間によるメモリ管理機構が盛んに議論されはじめている。その理由の一部を以下に挙げる。

1. 64ビット幅のアドレス空間を有効に活用するため
2. 分散システムにおけるアクセス透過性を得るため

単一アドレス空間システムは、単一のアドレス空間内ですべてのプロセスを実行するシステムである。この機構の利点・欠点を以下に挙げる。

- キャッシュ・TLB のフラッシュが不要
  ある仮想アドレスに 2 つ以上の物理アドレスが対応することはないので、コンテキスト切替えの際にキャッシュと TLB をフラッシュする必要がない。
- 空間的オーバヘッドの減少
  ページテーブル保持のための空間的オーバヘッドが小さい。
- ユーザレベルスケジューラの実装が容易
  ユーザレベルスケジューラは、特殊なメモリマッピングストラテジを用いることなく、すべての実行可能スレッドの優先度を取得できる。
- 実行形式の制約
  プログラムの実行開始アドレスが不定なので、リロケータブルな実行形式プログラムでなければならない。
- 新しい保護機構の必要性
  タスクに割り当てられた仮想空間を正しく保護するため、アドレス空間に代わる保護の単位が必要である。
- エミュレーションが困難
  現在主流のプロセッサは複数アドレス空間でのタスク管理を前提としており、単一アドレス空間システムの実装に適したハードウェアサポートは受けられない。しかもソフトウェアエミュレーションの実装は困難で、かつ時間的オーバヘッドが増加する。

単一アドレス空間システムをロボットのようなリアルタイムシステムのための OS に適用する際には、システムの予測可能性の低下を抑えるために 1 番目の項目が特に重要である。そこで本研究では、単一アドレス空間によるメモリ管理機構を *μ-PULSER* に採用し、以下に挙げる 2 点の両立を図る。

- タスクごとのメモリ空間を保護する
- システム全体の予測可能性の低下を防ぐ

2.4 実装時の問題点

本研究の実装対象は SPARC アーキテクチャのプロセッサ μ-SPARC である。このアーキテクチャは UNIX を強く意識したアーキテクチャであり、MMU は複数アドレス空間をサポートすることを前提とした機能を提供している。このため SPARC は単一アドレス空間システムに適しておらず [4]、実装のためにはエミュレーションを行なう必要がある。さらに SPARC アーキテクチャ特有のレジスタウィンドウを、関数呼び出しのネスト時のレジスタ退避に用いると、ウィンドウオーバフロー、アンダフローの発生が予測不可能となり、リアルタイムシステムには適さない。したがって、リアルタイムシステムに適したレジスタウィンドウの使用モデルを設計する必要がある。

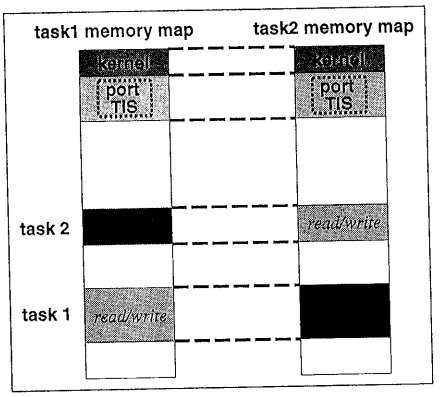

図 1: エミュレーションによる単一アドレス空間

3 単一アドレス空間システムの設計・実装

3.1 単一アドレス空間システムの設計

まず本稿では、*μ-PULSER* におけるタスクの定義を変更する。タスクは、これまでプロセッサ以外の資源の割り当て単位であった。しかし単一アドレス空間システムではアドレス空間が全タスクを通して 1 つになるため、この定義を変更する必要がある。本稿において、「タスク」を以下のように定義する。

タスク ＝ メモリ空間の保護の単位

すなわち、1 タスク ＝ 1 保護ドメインとする。

3.1.1 メモリ割り当て

上記の定義に基づき、タスクへのメモリの割り当て機構をエミュレートする。メモリマップの例を図 1に示す。エミュレーションなので、実際にはアドレス空間はタスクごとに分離している。存在するアドレス空間すべてに対し、全タスクのメモリ空間を同じアドレスにマップする。図 1の例では、たとえ

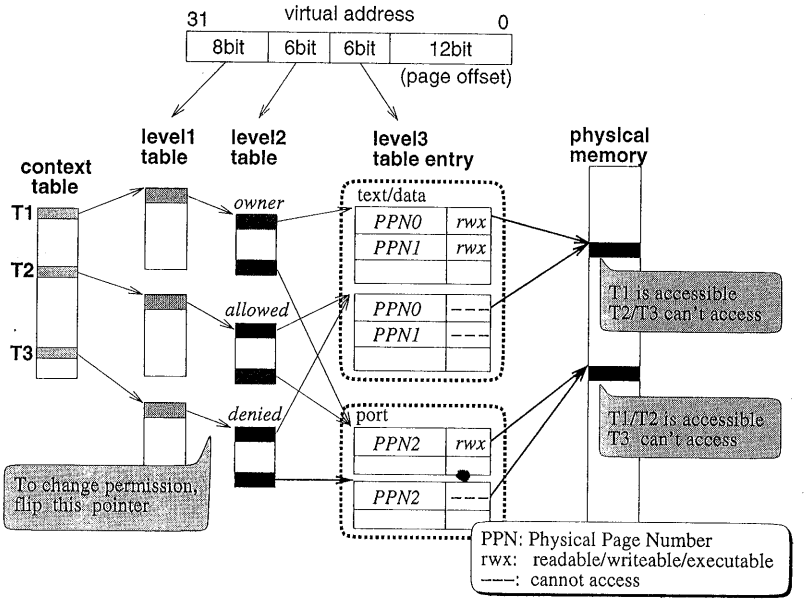

図 2: リアルタイムシステムのための単一アドレス空間ページテーブル

ばタスク 1 のアドレス空間にはカーネル、タスク 1、タスク 2 がマップされているが、タスク 2 のメモリ空間にはアクセスできない。タスク 2 のアドレス空間はその逆である。このメモリ空間を構成するためのページテーブルについて、次の節で述べる。

3.1.2 リアルタイムシステムに適したページテーブル

本稿では図 2のようにページテーブルを構築することを提案する。各タスク (保護ドメイン) はそれぞれ 1 つのコンテキスト ID を持つ。コンテキスト ID はハードウェアによるメモリ保護を受けるために必要であり、コンテキストテーブル内の 1 つのエントリを指し示す (このエントリの内容を ptp_1 とする)。レベル 1 ページテーブルは保護ドメインごとに 1 つ存在し、ptp_1 によって指し示される。レベル 1 ページテーブルの各エントリはレベル 2 ページテーブルのアドレスを指し示す (これを ptp_2 とする)。レベル 2 ページテーブルはレベル 3 ページテーブル (物理ページ番号と保護情報を格納している) へのポインタを格納しているので、複数の ptp_2 が同じレベル 2 ページテーブルを指し示す時、複数の保護ドメインがその保護情報を共有できる。

そこで、同一のアドレス変換を行なうが、保護情報のみが異なるレベル 3 テーブルを複数用意しておき、同数のレベル 2 テーブルを用意してそれぞれ別々のレベル 3 テーブルを指し示すように初期化しておく。すると、ある仮想アドレスの保護情報を変更する際、ptp_2、すなわちレベル 1 テーブル内の 1 つのエントリを変更するだけで切り替えることができる。こうすることにより、タスク数が増えたときでも余分なテーブル更新を行なわずに済む。本研究で提案する保護情報の種別は以下の 3 つである。なお、以下の項目中の「ポート」は、スレッド間通信のための共有バッファを指す。

- owner: 保護ドメインの所有タスクであり、メモリ空間内・ポート共に読み書き可能
- allowed: 保護ドメインの所有タスクへの通信が許可されており、ポートに対してのみ読み書き可能
- denied: その保護ドメインへのアクセスは一切禁止

このモデルならば、ある保護ドメインが、自分に属するページを別のあるドメインに対して公開する場合、そのページに該当する 3 つの保護情報を変更するだけでよい。この数はタスク数に依存しないので、リアルタイムシステムに適していると考える。仮想アドレス情報を変更するために更新すべきエントリ数を、単純に複数のアドレス空間を同じアドレス変換させる機構と比較した結果を、表 1に示す。

表 1: 更新すべきエントリ数の比較

処理内容	提案する変換機構	従来のアドレス変換機構
新規タスク生成	$3n_p$	$n_p x$
1ページ割当て	3	x
1ページ解放	3	x
あるページに対する保護情報の変更	1	x

(n_p: タスクの持つページ数, x: タスク数)

3.2 単一アドレス空間システムの実装

実装対象について説明する。プロセッサは Texas Instruments の TMS390S10 (μ-SPARC) 50MHz である。メインメモリは 8MB であり、SBus、シリアル、パラレル、イーサネットの各インタフェースを持つ。実装は、既に MC68030 上で動作している *μ-PULSER* の移植と、単一アドレス空間システムへの変更を並行して行なった。その後複数アドレス空間システムとの比較を行なうため、複数アドレス空間の *μ-PULSER* も実装した。

3.2.1 ポート

スレッドの持つポートへのアクセスは、ポートのアドレスを取得し、その領域にメッセージを直接書き込むことで行なう。ケイパビリティチェックに要するオーバヘッドを省くため、今回はポートのケイパビリティは実装せず、ポートのメモリ領域を各タスクに配置して全タスクに公開するように実装した。

3.2.2 CI 信号処理

CI 処理ルーチンと割込み処理ルーチンは割込み禁止状態で実行される。*μ-PULSER* における割込み処理 (CI 処理を含む) は、割込みのネストを禁止するかわりに処理時間を短く抑え、割込み処理時間を予測可能にするという方針に基づいて設計されている [5]。本実装でもこの方針に従った。このためカーネルコンテキストウィンドウの退避処理は実装せず、特権状態でトラップが発生したときはカーネルの実行を停止するようにした。

3.2.3 レジスタウィンドウ

レジスタウィンドウの割り当てを図 3に示す。関数呼び出し時にレジスタウィンドウをシフトさせないフラットモデルを採用し、関数呼び出しの実行時間を一定にした。特権モードとユーザモードで使用するウィンドウを分け、割込みレベル変更、CI 信号発生などのシステムコール発行時のオーバヘッドの軽減を図った。

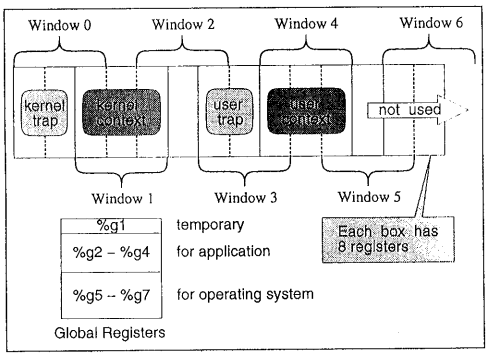

図 3: レジスタウィンドウの割り当て

表 2: 基本性能の測定結果 (単位: μs)

	単一空間		複数空間	
	平均	最大	平均	最大
ヌルシステムコール	12.7	13	12.8	13
スレッド生成	2813	2856	3043	3100
コンテキスト切替え	35.3	37	36.0	37

4 評価

スレッドの実行に関する基本的な性能評価を表 2に示す。単一アドレス空間と複数アドレス空間のカーネルで、それぞれ 1000 回ずつ計測した。測定は、実行中のスレッドが 5 つ、タスクが 2 つの状態で行なった。表を見ると、両者にほとんど差はない。平均時間で比較すると、単一アドレス空間の方がスレッドの生成において 10%、コンテキスト切り替えでは 2.3%改善されているのみである。この結果は、SPARC のキャッシュミス処理が命令実行に隠蔽されており、予想よりフラッシュの影響が小さかったことを示している。

次に、タスク数の変化に対するメモリ操作のオーバヘッドを測定するため、スレッドのポートを確保する処理を、並行実行させるタスク数を変化させて測定した。本稿で提案した owner/allowed/denied モデルと比較するため、保護機構を持たないアドレス変換機構におけるコストも測定した。図 4にそ

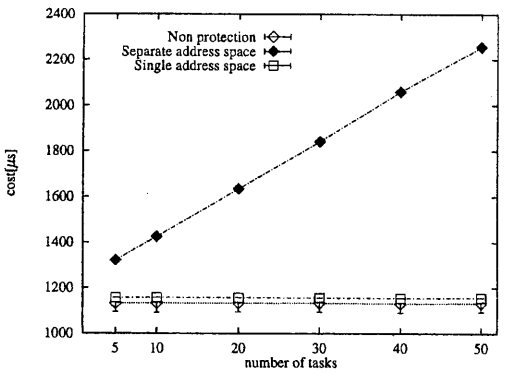

図 4: 実行タスク数に対するポート確保時間の変化

の結果を示す。図中、コストが線形に増加しているのが複数アドレス空間によるメモリ管理である。線形に増加している理由は、複数のアドレス空間が存在する場合にすべてのテーブルツリーを更新する必要があるためである。これに対して保護機構を設けない場合と単一アドレス空間システムでは、並行実行中のタスク数に依存せず、それぞれ 1131μs と 1156μs であった。保護機構のない単一アドレス空間では 1 つのエントリの更新だけなのでコストは一番少なく、かつタスク数に依存しない。owner/allowed/denied モデルは、ページエントリの更新箇所が 3 箇所のみであり、タスク数に依存しない。このためページ情報更新のコストは、保護機構のないアドレス変換機構のコスト 1131μs に 25μs のオーバヘッドが加わった 1156μs になっている。リアルタイム OS において各サーバのスレッド生成時間がタスク数に依存しないことは極めて重要であり、提案したページテーブル構造がタスク実行時間の予測に有効であることがわかった。

5 まとめ

リアルタイムシステムに適した単一アドレス空間によるメモリ保護機構を提案して*μ-PULSER* に実装し、その性能を評価した。その結果、スレッドの生成とポート領域の確保に関して、実行中のタスク数に依存しない処理性能を得られたことがわかった。しかしキャッシュのフラッシュが不要であることによるオーバヘッドの改善に関しては、大きな性能向上を得ることができなかった。

今後の課題としては、レジスタウィンドウの使用法の検討、およびμ-SPARC 上の*μ-PULSER* の速度向上を考えている。

参考文献

[1] 矢向高弘, 菅原智義, 安西祐一郎. μ-PULSER: パーソナルロボットを構築するためのオペレーティングシステム. 電子情報通信学会論文誌 D-I, Vol. J77-D-I, No. 2, pp. 207–214, February 1994.

[2] Takahiro Yakoh and Yuichiro Anzai. A new reactive operating system for human-robot interaction. *Advanced Robotics*, Vol. 8, No. 4, pp. 371–383, 1994.

[3] 山崎信行, 安西祐一郎. パーソナルロボット用機能別並列計算機アーキテクチャ: ASPIRE. 情報処理学会論文誌, Vol. 37, No. 1, January 1996.

[4] Eric J. Koldinger, Jeffrey S. Chase, and Susan J. Eggers. Architectural support for single address space operating systems. In *Proceedings of 5th International Conference on Architectural Support for Programming Language and Operating Systems*, pp. 175–186. Vol. 26 of ACM SIGPLAN Notices, October 1992.

[5] 矢向高弘, 菅原智義, 安西祐一郎. Pulser: リアクティブシステムの構築に適したオペレーティ ングシステム. コンピュータ・ソフトウェア, Vol. 11, No. 1, pp. 24–35, January 1994.

[6] Yuichiro Anzai. Towards a new paradigm of human-robot-computer interaction. In *Proceedings of IEEE International Workshop on Robot and Human Communication*, pp. 11–17, September 1992.

[7] 河野通宗, 前沢敏行, 安西祐一郎. 単一仮想記憶空間を提供するリアルタイム OS の設計. 情報処理学会システムソフトウェアとオペレーティングシステム研究会報告 71–7, pp. 33–38, November 1995.

[8] Richard Gerber and Seongsoo Hong. Compiling real-time programs with timing constraint refinement and structural code motion. *IEEE Transactions on Software Engineering*, Vol. 21, No. 5, pp. 389–404, May 1995.