システムソフトウェアと
オペレーティング・システム　89－１９
シ　ス　テ　ム　評　価　2－１９
（２００２．２．１５）

組込向けマイクロカーネルOS Lambdaの メモリ管理機構の実装

久住憲嗣* 北須賀 輝明† 中西恒夫‡ 福田晃‡

{nel, kitasuka}@f.csce.kyushu-u.ac.jp, tun@is.aist-nara.ac.jp, fukuda@f.csce.kyushu-u.ac.jp

あらまし

本研究室ではOSの保守性や開発効率を、主に保護利用してを向上させるため, マイクロカーネル構成を採用した組込向けマイクロカーネルOS Lambdaの設計, 実装を行っている。組込システムでは様々なプロセッサが使用され、また、様々な条件をアプリケーションが要求するために、保護、とくにメモリ保護の実装に注意を払う必要がある。本稿ではこれらのことを考慮し、組込向けメモリ保護フレームワークの提案を行い、またLambdaのメモリ管理の実装について説明する。

The memory management system of the Lambda μ-kernel based operating system for embedded systems

Kenji Hisazumi* Teruaki Kitasuka‡ Tsuneo Nakanishi† Akira Fukuda‡

Abstract

We implement an embedded operating system, called the Lambda operating system, which improves the maintainability and development efficiency of the operating system by protection. Embedded systems employ various processors and embedded applications require characteristics of real-time. We propose the memory protection framework for these features, and we explain the memory management system of the Lambda.

*九州大学 システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†九州大学 システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University
‡奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute Science and Technology

1 はじめに

近年, 情報家電の登場などにより, 組込システムが大規模化してきている. 大規模化するにつれ開発工数が増加する反面, 競争の激化などにより製品を短期間に設計・開発し, 市場に出すことが求められている. このため組込システムの開発において, 開発工数の削減が必要不可欠であり, 開発効率の向上が重要な課題となっている. 組込システムの開発効率を向上させるために, 組込向けオペレーティングシステム (OS) の保守性や開発効率が重要であるにもかかわらず, この点に注目し OS を実装している例は少ない. そこで, 本研究では,OS の保守性や開発効率を向上させるため, マイクロカーネル構成を採用した組込向けマイクロカーネル OS Lambda の設計, 実装を行っている。

Lambda はマイクロカーネルに近い構成で開発、デバッグ等をおこない、保護を有効に利用し開発効率の向上をはかる。また、開発終了後には必要がない部分の保護を排除し、モノリシックなバイナリに変換することによって、性能の向上をはかる [1][2]。

組込みシステムは、さまざまなハードウエアに対応する必要があり、また、さまざまなアプリケーションの要求にも対応する必要がある。本稿では、これらのことに留意しつつ Lambda のメモリ管理機構について述べる。

2 保護の重要性

従来の組込システムでは保護を利用せずにシステムを構成することが多いが、現在ではシステムの規模が大きくなってきており、保護が無い状態で開発を行うことが大変難しくなってきている。また、従来は少なかった外部システムとの通信が必要なので、セキュリティという面でも保護が重要になってきている。

そこで、Lambda では汎用システムなみのメモリ保護機構を提供するために、ページングを利用し柔軟なメモリ管理が行えるようにする。また、組込システムに多く存在するリアルタイム制約が厳しいソフトウエアに対応できるような機能の提供も行う。

3 組込向けマイクロカーネル OS Lambda

3.1 プロセスモデル

Lambda は Mach のようなプロセスモデルを採用する。プロセスは保護の単位であり、プロセスにはスレッドが 1 つ以上存在する。Lambda はカーネルといくつかのシステムサーバから構成される。システムサーバはプロセスの特別なもので、OS の機能の一部の提供を行う。たとえばメモリ管理サーバなどである。

3.2 プロセス間通信

Lambda ではメモリ管理とプロセス間通信 (IPC) は密接な関係がある。メモリ管理機構について議論する前に、Lambda における IPC の概要について説明する。

Lambda に IPC を実装するにあたり, 高速であること, 及びデータの受け渡しが柔軟にできることを目標とした. そこで, 遠隔手続き呼び出し (RPC) を使用することを前提とした IPC プリミティブの提供をおこない, また, メモリのコピー, マッピングを柔軟に行うことができるようなデータ転送モードの実装を行う.

IPC プリミティブとしては以下のようなものの実装を行う.

- call
  遠隔手続き呼び出しを行うために使用するモードである. 受信側スレッドに引数を渡し, 返答がくるのを待つ.
  リクエストの送信のために送信デスクリプタと送信引数を, 返り値の受信のために送信スレッドと受信ディスクリプタ, 受信引数を指定する. 返答を受信するか, タイムアウトするまでブロックされる.

- send_reply_and_recv
  RPC 等のサーバの性能を向上させるために, 値を返答する動作と, 引数を受信する動作を一つにまとめたモードである. 一つにまとめることにより, 通常だと send,recv と 2 回システムコールを発行しなくてはならないところを, 一度のシステムコール呼び出しで処理が完了し, コンテキストスイッチ回数が減り性能が向上する. 返答のために送信デスクリプタと送信引数を, リクエストを受信するために送信側スレッドと受信ディスクリプタ, 受信引数を指定する.
  受信が完了するか, タイムアウトされるまでブロックされる.

- send
  送信のみを行うモードである.
  送信デスクリプタ, 送信内容を指定する.
  受信側により受信されるか, またはタイムアウトするまでブロックされる.

- recv
  受信のみを行うモードである.
  送信側スレッドと受信デスクリプタ, 受信内容を格納する変数を指定する.
  送信側がメッセージを送信し受信が完了するか, またはタイムアウトするまでブロックされる.

また, データ転送モードには以下のようなモードがある.

- copy
  送信元から送信先へデータのコピーを行う.

- map read/write
  領域を共有メモリとして利用するために, 送信元のページを送信先のページに書き込み許可の状態でマップする. 送信元はページ境界に注意して転送を行わないと, 送信先にデータを破壊されてしまう可能性がある.

- map read only
  送信元のページを送信先のページに read only の状態でマップする. 書き込まれたくない場合や, 読み込みのみで使用する場合にこのモードで転送する. 送信元はページ境界に注意して転送を行わないと, 送信先に望まないデータを読み込まれてしまう可能性がある.

すべてのデータ転送モードを copy on write として実装しても良いのだが, 機構が複雑であり, またコピーする必要のない用途で使用することも多いので, 細かくデータ転送モードを指定できる方法を採用する.

4 メモリ管理機構

4.1 概要

MMU をもつプロセッサは、アドレス空間に関して以下のようなモードで動作する。

1. 物理アドレス空間
2. 単一仮想アドレス空間
3. 複数仮想アドレス空間

Lambda は 2, 3 で動作させることを前提に実装を行う。Lambda を 1 で動作させることも可能であるが、ここでは議論しない。

Lambda では、アドレス変換機構、メモリ管理機構に厳密にわけて実装を行う (図 1)。アドレス変換機構は、主にプロセッサ依存のアドレス変換機構のサポートを行う。ページのマップ、アンマップなどの操作を実現し、ページフォルトの捕捉、プロセッサに必要な情報の設定を行う。Lambda はどのようなアドレス変換機構を持っていても簡単にサポート可能なように、アドレス変換機構の抽象化を行う。メモリ管理機構は、主にプロセッサ非依存のメモリ管理を行う。具体的には、ユーザプロセスとカーネルに対してメモリを割り当てる。

メモリ管理機構は通常はシステムサーバとして実装する。ユーザプロセスに対しては、プロセス間通信 (IPC) のマップ機能を利用して、メモリ割り当ておよびマップを行う。このように、厳密にわけて実装することにより、移植性や再利用性の向上をはかる。

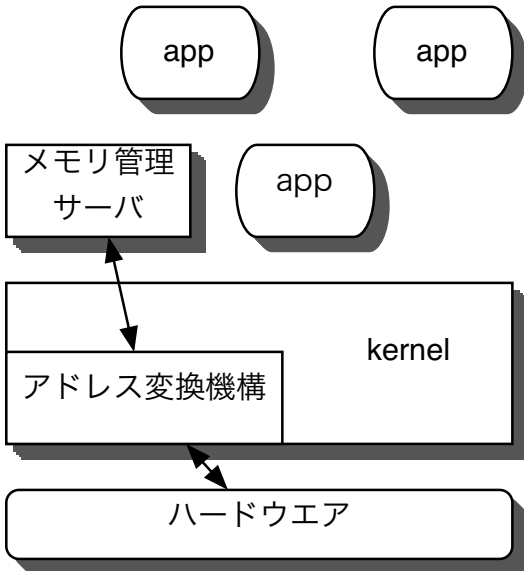

図 1: Lambda のメモリ管理機構の構成

4.2 メモリ管理機構の目標

Lambda のメモリ管理機構を実装するに当たって、目標として以下のようなものを挙げる。

- 様々なアーキテクチャに対応
- リアルタイム性を要求される用途への対応
- メモリ割り当てポリシーを簡単に変更可能
- 保護を効率よく利用できるようにするためのフレームワークの提供

4.3 メモリ管理機構の概要

Lambda の IPC は、データの転送方法として、コピーによる転送と、マップによる転送の 2 種類をサポートしている。このうちの、マップ機能を利用して、メモリ管理サーバがユーザプロセスにメモリ割り当てを行う。起動時にカーネルはすべての物理メモリをメモリ管理サーバに割り当てる。メモリ管理サーバは、この中からユーザの要求に応じてメモリの割り当てをおこない、IPC を利用して、ユーザプロセスが実際に利用できるようにする。

ページフォルトがおこると、アドレス変換機構がよばれ処理される (図 2)。アドレス変換機構はおもに仮想アドレスの解決を行う。アドレス変換機構が処理できなかったページフォルトは、RPC 経由でメモリ管理サーバにつたえられる。メモリ管理サーバは適切な処理をおこなった後、アドレス変換機構に返答を返す。

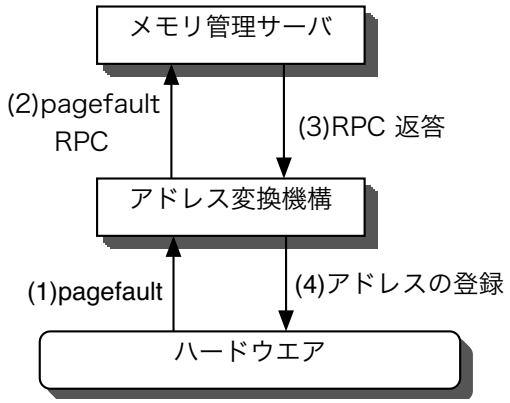

図 2: ページフォルトハンドリング

4.4 ユーザプログラム

ユーザプログラムは、遠隔手続き呼び出し (RPC) を用いてメモリの割り当て要求をメモリ管理サーバに対して行う。この RPC をメモリ割り当て RPC とよぶ。この RPC に対する返答により、実際にメモリの割り当てがおこなわれる。RPC を利用してメモリ割り当てを行うことにより、ユーザプログラムは簡単にメモリ割り当てを受けることが可能になる。

メモリ割り当て RPC を実現するために、通常の RPC でのデータ転送指定である IN, OUT 指定以外に MAP 指定をサポートする。MAP 指定は、送信側の指定されたページを受信側のアドレス空間にマップするものである。この機能は、カーネルがサポートする IPC のマップ指定を利用して実現される。

5 アドレス変換機構フレームワーク

5.1 概要

組込システムで採用されるプロセッサには様々なページング機構が存在し、それらに対応する必要がある。また、組込システムはリアルタイム性を要求するので、TLB ミスのペナルティが大きいページングを利用するときには、さまざまな注意が必要となる。これらを実装するにあたり、アーキテクチャごとに実装したり、個々のアプリケーション用にチューニングしたりしていては、工数がかかりすぎる。そこで、多くのプロセッサに共通して必要な実装を、アドレス変換機構フレームワークライブラリとして提供する。Lambda はアドレス変換機構フレームワークライブラリを利用し、アプリケーションにサービスを提供する。

5.2 対応すべきハードウエア

現在、組込みシステムでよく利用されるプロセッサには、以下のようなページングハードウエアが存在する。

- x レベルページング
  ページテーブルを x 段に分けて OS が管理し、プロセッサがページテーブルを自動的に引き、仮想アドレスの解決を行う方式。

- TLB 更新型
  対照的に、プロセッサはページテーブルを引かずに、TLB が miss したときには、ページフォルトが起こる。OS が自分でページテーブル等を検索して、TLB に登録することにより仮想アドレスを解決する方式。また、さらに、効率よくするために、TLB エントリにプロセス id を持たせたものも存在する。プロセス id を TLB エントリに持たせることにより、プロセス切り替えの際に TLB をフラッシュする必要がないので、多仮想アドレス空間を高速にスイッチすることが可能になる。

他の特別なメモリ管理ハードウエアを持つものとして、組込みシステムで多く採用されている ARM がある。ARM はドメインとよばれる特殊な保護機構をサポートしている [3]。この保護機構を利用すると、保護のオーバヘッドを最小限にとどめつつも、保護を利用することが可能になる。この場合には、保護単位がプロセスとはちがったものになるが、属性の設定と組み合わせることによって、サポートする。

5.3 OS へ提供する機能

Lambda のためにアドレス変換機構を記述するには、以下のような機能の実装を行う必要がある。

- アドレス空間の初期化と生成
- 物理ページのマップ
- ページのアンマップ
- 仮想アドレスを物理アドレスに変換
- ページ属性の設定
- ページフォルトハンドラ

Lambda は保護単位ごとに id を割り当て、その id を利用して以上の操作を行う。通常は、id とプロセスは一対一に対応する。

5.4 属性の設定

また、組込みシステムは、リアルタイム性能がとくに重視されるので、キャッシュの無効化やページを TLB からはずさないというような、特別な設定を必要とする場合が多い。これらをサポートするために、柔軟にページ属性の設定をおこなえるように実装を行う。

6　モノリシック化

アドレス管理機構とメモリ管理機構を厳密に分割して実装することにより、移植性が向上し、また実装も容易になるが、性能が良くない。メモリの割り当てはとくによく行われるので、この部分の性能が低いのは問題がある。そこで、モノリシック化の手法を用いる。モノリシック化とは RPC を関数呼び出しに置換し高速化をはかる手法のことである [2]。メモリ管理サーバをカーネル空間で動作させ、さらにメモリ割り当てを行う RPC を関数化することにより性能を向上させる。これにより、カーネルとカーネル空間で動作するプログラムに対して、高速にメモリ割り当てをおこなえるようになる。

また、RPC で伝えられるページフォルト処理も、モノリシック化することにより高速になる。

ただし、メモリ割り当ての RPC は他の RPC と違いメモリマップを変更するので、カーネルの機能を利用してメモリマップの変更を行う必要がある。ここで、メモリマップを変更するシステムコールを提供し、RPC スタブ内で呼び出す実装が考えられる。しかし、システムコールの発行は遅く、大幅な性能の向上があまり見込めない。メモリ管理サーバをモノリシック化して性能の向上が見込めるのは、カーネル空間で動作させる場合のみである。そこで、メモリマップを変更するアドレス管理機構内の関数を、RPC スタブ内で直接呼び出す。これにより、メモリ割り当て RPC の意味をかえずに、高速化をはかる。

7　まとめ

本稿では、Lambda のメモリ管理についてのべた。組込みシステム向け OS はさまざまな要求にこたえるために柔軟な設定を行うことができる必要があり、メモリ管理についても例外ではない。そこで、アドレス変換フレームワークライブラリを提供し、多くのプロセッサにおいて共通して必要となる機能を再実装せずに、必要な機能のみの実装をおこなうだけで、さまざまな要求に対応できるようにする。また、開発効率と性能の両立のために、メモリ管理にもモノリシック化を適用する手法の提案をおこなった。

謝辞

本研究の一部は科学技術振興事業団戦略的基礎研究推進事業 (CREST) の支援のもとに行われたものである.

参考文献

[1] 福田晃, 最所圭三, 片山徹郎, 中西恒夫: 組込システム向け実行環境の自動生成 – δ プロジェクトの構想 –, 電子情報通信学会技術研究報告, No. 726, pp. 17–22 (2000).

[2] 久住憲嗣, 中西恒夫, 福田晃: 組込み向けマイクロカーネル OS Lambda の保護機構, コンピュータシステムシンポジウム論文集, pp. 81–88 (2001).

[3] S.Furber: ARM プロセッサ, CQ 出版社 (1999).