マイクロカーネルにおける動的カーネルモジュールのサポート

5B−10

追川 修一[1]　杉浦 一徳[2]　西尾 信彦[1]　徳田 英幸[1,3]

[1]慶應義塾大学環境情報学部
[2]慶應義塾大学大学院政策・メディア研究科
[3]カーネギーメロン大学計算機科学学部

1 はじめに

移動計算機環境 [2] を実現するためには，計算機自身の構成を含めて変化する環境に動的に適応可能しなければならない．そのような，動的適応可能なソフトウェアアーキテクチャにおけるカーネルは，アプリケーション自身が，必要とする機能をカーネルに提供することができるものでなければならない．そのための機能として，マイクロカーネルにおける DKM (Dynamic Kernel Module) のサポートについて述べる．

DKM はマイクロカーネルの拡張の単位であり，関数とそれによってのみ操作されるデータを持つ．DKM は，マイクロカーネルに単に機能を追加することができるだけでなく，限られた資源しか持たない移動計算機においては重要となる，必要でなくなった機能をカーネルからアンロードしたり，または一時的にカーネル外にデタッチすることができる機能を持つ．DKM は，それ自身の機能を規定しないため，DKM 間の関係を管理し，そのフレームワークの維持を行なう DKM サーバが必要となる．DKM サーバは，DKM のカーネルへのロード，カーネルからのデタッチまたはアンロードを行なうだけでなく，プログラム可能にすることにより，計算機環境の変化に動的適応するためのイベント処理を自律的に行なうことができる．

以下本論文では，2 章において DKM，DKM サーバについてより詳細に述べ，3 章においてノート型 PC における PCMCIA カード管理への応用について述べる．4 章では関連研究と比較を行ない，5 章で本論文をまとめる．

2 ソフトウェアアーキテクチャ

2.1 DKM: Dynamic Kernel Module

DKM は拡張の単位であり，ファイルやメモリオブジェクトといった形態でカーネルに提供することができる．それぞれの DKM は，関数とそれによってのみ操作されるデータを持つ．DKM の関数は，その DKM が提供する関数のリストを通して呼び出すことができる．

アプリケーションは，一般的なカーネルが提供することのない機能を必要とする場合がある．そのようなアプリケーションは，その実行をより効率よく行うために，必要とする機能を持つ DKM をカーネルにロードすることができる．ロードされた DKM を用いることにより，アプリケーションに特化したサービスがカーネルにより提供されることになる．

DKM は，それ自身がポリシーを提供するものであるか，それともメカニズムを提供するものであるかは規定していないため，そのどちらの用途にも使用することができる．DKM が実際にどちらにあてはまるかは，DKM によってではなく，DKM の関係を定義するフレームワークによって決定される．例えば，ある DKM をカーネルにロードしようとした場合，もしその DKM がメカニズムまたはポリシーのみを提供するものであった場合，それだけでは不十分である．その DKM がポリシーモジュールの場合はメカニズムを提供する DKM，メカニズム DKM の場合は，別にポリシーモジュールが指定されていなければ，デフォルトポリシーを提供する DKM も一緒にロードしなければならない．そこで，DKM のフレームワークを維持，管理するサーバとして DKM サーバが必要になる．

2.2 DKM サーバ

DKM サーバは，DKM のカーネルへのロード，カーネルからのデタッチを矛盾なく行うことに責任を持つ．サーバは，(1) カーネルへロードする DKM を受け取り，(2) それが正しいインタフェースを提供しているかどうか検査し，(3) 外部シンボルを解決するためカーネルとリンクを行い，最後に (4) カーネルへロードする．DKM をカーネルへロードするために必要なインタフェースは，マイクロカーネルにより提供される．

もし DKM サーバが正しいインタフェースを提供していない DKM やアクセスが許されていないカーネルのデータを使用しようとする DKM を発見した場合，その DKM のロードは拒否される．しかしながら，DKM サーバはおかしな動作によりカーネル内のデータを破壊してしまう DKM を発見することはできない．そのような動作を防止するためには，型安全な言語や SFI[1] [4] を使用することができる．しかしながら，長時間ロックを取得したまま実行を続けたり，無限ループから抜け出さないといった，セマンティックな間違いは，型安全な言語や SFI によっても防ぐことはできないため，これらのセマンティックな間違いによるカーネルのハングアップやパニック，データの破壊を防止するためのメカニズムの開発が必要である．

DKM サーバは，DKM のフレームワークを管理する．DKM 間の関係を把握し，またカーネル内における DKM の状態を絶えず注意している．ある DKM のカーネルへのロードが要求された時，その DKM だけでなく，他に必要な DKM があればそれらも同時にカーネルへロードされるようにする．また，ある DKM が不必要になった時，それをカーネルからアンロードする．メモリが不足する状態になった場合，重要でないアプリケーションによって使用されている DKM をカーネルからデタッチするように，カーネルは DKM サーバに通知する．もし，DKM を退避するストレージが利用可能でないなどの理由で，デタッチが不可能な場合は，そのアプリケーションの実行は中止されることになる．

DKM サーバは，ユーザレベルサーバとして実装される．DKM サーバは，カーネルの動的再構成に重要な役割を持つが，動的再構成は常に必要なわけではない．例えば組み込みシステム用のカーネルを構成することを考えた場合，DKM を用いることにより若干異なった機能を持つカーネルの構成が容易になるが，カーネルの動的再構成は必ずしも必要でない．マイクロカーネ

"Dynamic Kernel Module Support in a Microkernel"
Shuichi Oikawa[1], Kazunori Sugiura[2], Nobuhiko Nishio[1], and Hideyuki Tokuda[1,3]
[1]Faculty of Environmental Information, Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan, [2]Graduate School of Media and Governance, Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan, [3]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[1]Software-Based Fault Isolation

ルアーキテクチャにおける他のサーバと同じように，DKMサーバもユーザレベルサーバとして実装することにより，動的再構成が必要なシステムにおいてはDKMサーバを用いることができるようになる．

DKMサーバには，インタプリタが内蔵される．モーバイルシステムにおいて計算機環境の変化に動的に対応することを考えると，DKMサーバはプログラム可能でなければならない．プログラム可能にすることにより，アプリケーションはある特定のイベントに対応した動作をDKMサーバは自律的に処理することができるようになる．動的適応のためのイベント処理をDKMサーバに集中することによって，DKMとイベント処理やDKMの管理を分離することができ，DKMの実装は一般化され再利用性が高まることになる．

3 PCMCIAカード管理への応用

DKMサーバが，実際にノート型PCのPCMCIAカード管理に有効であることを示す．現在市販されているほとんどのノート型PCにはPCMCIAスロットが備わっており，そこにPCMCIAカードを挿すことにより，拡張性が乏しいノート型PCに新たなデバイスを追加することが可能である．PCMCIAカードは，ホットスワッピングと呼ばれる，電源が入って動作している途中でPCMCIAカードを取り外したり挿したりすることができる機能を持っている．即ち，ユーザはノート型PCの動作中に，不必要になったカードを取り外し，新たに必要となったカードを挿すことができる．多くのノート型PCで利用可能になっているスリープ機能とあわせて，PCMCIAカードの持つホットスワッピング機能は，従来のOSにおけるデバイス管理方式に変革を必要としている．

あるユーザが実際にどのPCMCIAカードを使用するかどうかは予め知ることができないが，使うであろうすべてのPCMCIAカードのためのデバイスドライバをカーネルに含むようにしてしまうのは良い考えではない．それらのデバイスドライバはコードやデータ領域にかなりのメモリ資源を消費してしまうが，実際にはある決まったPCMCIAカードのためのデバイスドライバしか使われないからである．また，別の問題点として，PCMCIAカードは，それを使用可能にするための初期化プロセスにおいて，PCMCIA独特の処理を必要とするということがある．従って，ほとんど同じデバイスドライバがあるにも関わらず，PCMCIAカード用に別のデバイスドライバが必要になってしまう場合が多い．

DKMおよびDKMサーバをPCMCIAカード管理に応用することにより，以上2つの問題点を解決することができる．デバイスドライバをDKMとして実装することにより，実際にそのデバイスドライバが必要になったときにはじめてカーネルに組み込むことができる．これにより，実際には使われていないメモリ資源をデバイスドライバが専有してしまうという問題点は解消される．DKMはまた，使われなくなった場合に，カーネル外に退避または排出してしまうことが可能であるため，PCMCIAカードを取り出した場合にも，対応することができる．

DKMサーバは，PCMCIAカード管理に必要な2つの機能を提供することができる．1つは，どのDKMをロードすれば良いかという選択であり，もう1つは，PCMCIAカードの初期化である．DKMサーバをプログラム可能にしたことにより，これらの機能の提供が可能になっている．DKMサーバは，PCMCIAカードの挿入というイベントをカーネルから受け取り，PCMCIAスロットを管理するデバイスドライバを通して，挿入されたカードの情報を入手する．PCMCIAカードは，そのカードが何であるかの情報をCIS[2]として持っているため，その情報からそのカードに必要なデバイスドライバを持つDKMを決定することができる．DKMサーバは，そのDKMをロードした後に，PCMCIAスロットを管理するデバイスドライバを通して，PCMCIAカードの初期化を行う．そのための情報は予めDKMサーバに与えられている必要があるが，デバイスドライバとは分離された，DKMサーバのためのプログラムとして提供される．その後は通常のデバイスとして，ロードされたデバイスドライバを通して初期化を行い，PCMCIAカードは利用可能になる．

4 関連研究

OMOS (Object Meta-Object Server) [1] は，実行オブジェクトのダイナミックリンクを管理する．オブジェクトのダイナミックリンクのためのルールがサーバに提供され，実行時の状況からルールを解釈し，その時の実行に最適なライブラリが選択，ダイナミックリンクされる．

ルールとその時々の状況から最も適したオブジェクトを提供するという点で，OMOSとDKMサーバは似ているが，DKMサーバは，OMOSとは以下の点で異なっている．DKMサーバは，複数のDKMの間の関係を把握し，DKMの構成が矛盾したものにならないように，DKMを管理する．また，動的な計算機環境の変化に対応できるように，イベント処理を行い，それによってDKMの構成を動的に変化させることも可能である．そして，DKMサーバにおけるイベント処理をプログラムし自律的に動作させることにより，DKMサーバが動的適応の主体となることが可能になっている．

5 まとめと今後の予定

本論文では，移動計算機環境を実現するためカーネルに必要な機能として，計算機自身の構成を含めて変化する環境に動的に適応可能するソフトウェアアーキテクチャとして，DKMおよびDKMサーバから構成されるマイクロカーネルについて述べた．

現在，DKM，DKMサーバはRT-Mach実時間マイクロカーネル[3]に実装中であり，今後，DKM，DKMサーバを用いて，PCMCIAカード管理を行なう予定である．

謝辞

研究を進めるにあたって，有益なコメントを頂きました北陸先端大学院大学中島達夫助教授およびM^3Cグループメンバの方々に深く感謝します．

この研究は，情報処理振興事業協会(IPA)が実施している独創的情報技術育成事業「モーバイルコンピューティングのための動的適応可能なソフトウェアアーキテクチャ」プロジェクトのもとに行なわれています．

参考文献

[1] D. B. Orr, R. Mecklenburg. OMOS - An Object Server for Program Execution. In *Proceedings of International Workshop on Object Oriented Operating Systems*, IEEE Computer Society,September 1992.

[2] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proceedings of the Fifteenth ACM Symposium on Principles of Distributed Computing*, May 1996.

[3] H. Tokuda, T. Nakajima, and P. Rao. Real-Time Mach: Towards a Predictable Real-Time System. In *Proceedings of USENIX Mach Workshop*, October 1990.

[4] R. Wahbe, S. Lucco, T. Anderson and S. Graham. Efficient Software-Based Fault Isolation. In *Proceedings of Fourteenth ACM Symposium on Operating System Principles*, December 1993.

[2] Card Information Structure