DKM:適応的カーネルオブジェクト管理機構の性能評価

5Q－1

服部真由美[1] 追川修一[2] 徳田英幸[3]

慶應義塾大学総合政策学部[1] カーネギーメロン大学計算機科学学部[2]
慶應義塾大学環境情報学部[3]

1 はじめに

DKM(Dynamic Kernel Module)は，RT-Mach[1]マイクロカーネルにおける内部機能の拡張単位である．DKMを動的にロード，アンロードすることにより，カーネル内の動的再構成を実現することができる[2]．これにより，変化する環境やアプリケーションごとの要求に対してカーネルレベルで動的かつ柔軟に適応することが可能になる．

本稿では，DKM管理機構がDKMを組み込み，取り外しする過程における時間的コストを測定／解析し，DKMアーキテクチャの性能評価，考察を行なう．

2 DKMアーキテクチャ

2.1 ソフトウェアアーキテクチャ

DKMアーキテクチャでは，DKMへの操作は，ユーザレベルで動作するDKMサーバを通して行なわれる[2]．サーバは，マイクロカーネルによって提供されるインターフェースを用いてモジュールのロード，アンロードを行なう．モジュールに対する呼び出しは，カーネル内に登録されたモジュール関数を呼び出すことによって行なわれる．

図1に，DKMアーキテクチャの概観図を示す．

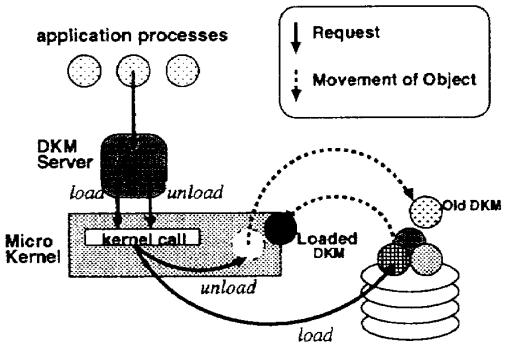

図1: DKMアーキテクチャ

2.2 DKMの組み込み

DKMサーバは，不要になったモジュールをアンロードし，組み込むDKMインターフェースの正当性の検査を行なった後に，DKMのロードを実行する．また，サーバはモジュール間の依存関係を監視し，関連する必要なモジュールを自動的にロードする機能を持つ．

実際の組み込み過程は，モジュールロード時に確保すべきメモリ領域の大きさを知るための予備リンク，メモリ領域確保，シンボル解決のためのリンク，カーネルへのロード，カーネルへ関数登録という段階を経る．

3 評価

3.1 方針

ここでは，DKMの機構を利用する際に生じるオーバーヘッドの原因を追求することを目的として評価を行なう．具体的には，以下に示す2つのアプローチから，DKM組み込み及び取り外しに要する時間を測定／評価する．

1. **DKM組み込み／取り外しの各処理過程に要する時間を計測，評価**
   DKMの組み込み及び取り外しの過程において，各処理がどのような割合を占めているのか評価する．
2. **データ量の異なるDKM間での組み込みに要する時間の比較**
   ここでは，DKMのテキスト／データセグメントの大きさを「DKM内のデータ量」と定義する．
   DKM組み込みの処理時間には，DKM内のデータ量の違いが最も影響すると考えられる．実際には，DKMは関数自身によってのみ操作されるデータしか持たないため[3]，データセグメントの大きさが大幅に変わることは考えられない．しかし，実際の処理においてテキスト／データセグメントは同一に扱われることを考え，ここではDKM内の初期化されたデータ量を変化させることによってDKM内のデータ量とロードの各処理部分の所要時間との関連を考察する．

なお計測の際にロードするDKMとして，測定1では現時点でDKMとして実装されているスケジューリングポリシモジュールを，測定2では測定用に用意されたダミーの変数と関数を持つモジュールを用いる．

また，この測定及び評価では，純粋にモジュール組み込みの処理部分のみを対象とする．

3.2 結果

計測のためのプログラムは，IBM Thinkpad 560(Pentium 133MHz,RAM 40M)上で実行した．また，以下の結果はPentiumプロセッサ内蔵のカウンタを用いて算出した値である．

3.2.1 測定1

ここでは，複数のスケジューリングポリシモジュールのロード，アンロードに要する時間の平均値を割り出した．この時，モジュールのデータ量の平均値は1.536KBであった．表1に，要した時間の平均値と全体に占める割合を示す．

DKMの組み込み処理全体では，770ミリ秒程度ですんでいる．しかし，予備リンクを省略し，楽観的な予測によって領域確保を行なうこともできる[3]ので，そうした高速性重視の最適化を図れば，全体で約400ミリ秒強にコストを減らすことが可能である．また，各過程におけるコストを見ると，一連の処理の流れの中で予備と実際の2度のリンクが全体の90%を占めていることがわかる．

DKM: The Evaluation of Adaptive Kernel Object Management Mechanism
Mayumi Hattori[1] Shuichi Oikawa[2] and Hideyuki Tokuda[3]
[1]Faculty of Policy Management, Keio University [2]School of Computer Science, Carnegie Mellon University [3]Faculty of Environmental Information, Keio University

表 1: ロード，アンロードの各過程におけるコスト

	平均値 (msec)	割合
予備リンク	332.31	43%
メモリ割り当て	1.25	0.2%
リンク	374.08	49%
ロード	13.78	1.8%
登録	1.92	0.2%
その他	47.02	6.1%
ロード全体	770.35	-
アンロード	2.66	-

一方で，アンロード処理に要する時間は 2.66 ミリ秒とロード処理全体に対して 1/300 以下の時間になっている．アンロードでは，カーネルからのエントリの削除とメモリ領域解放を要求するカーネルコールを呼び出しているだけであり，比較的軽い処理ですんでいるものと思われる．

3.2.2 測定 2

ここでは，2 度のリンク及びカーネルへのロード処理において，データ量に比例する形の増加率が見られた．図 2，3にその結果を示す．

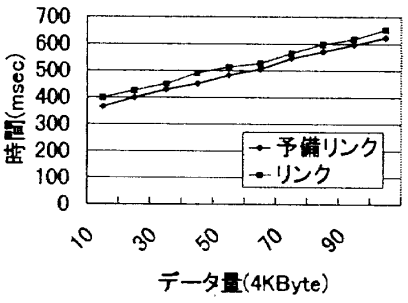

図 2: リンクに要する時間の変化

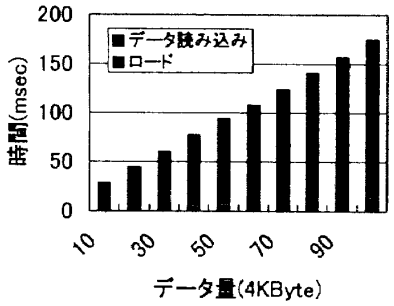

図 3: カーネルへのロードに要する時間

メモリ割り当ての場合にも，同様な変化は見られたが，もともとのロード時間全体に占める割合が非常に少ないこと，またデータ量の増加に伴う処理時間の増加率も極めて小さいこと（データ量 10：1msec，データ量 100：1.5ms）から，ここでは，ほとんどロード時間の変化に影響を与えないものとして扱った．

2 度のリンクでは，実際のリンクの方が多少値が大きくなるが，ほぼ同様の傾きを持つ一次曲線に近似できる結果を得られた（図 2）．

カーネルへのロードでは，おおまかに分けて 2 つの処理を行なっている．まずユーザ空間にメモリ領域を確保し，そこに DKM のテキスト／データセグメントを読み込む．その後に，カーネルコールによってその領域をカーネル空間にロードする処理を行なっている．この時図 3 より，ロード処理時間の 9 割を，ユーザ空間へのデータの読み込みが占めていることがわかる．しかもどちらの処理もデータ量に比例して処理時間が増加してゆくため，両方を加えたロード処理全体では，リンクの場合と同様に一次曲線に非常に近い結果を得ることができた．

4 考察

以上の測定により，第一に，ロード時間において大部分を占めるのが 2 度のリンク処理であること，第二に，ロードするモジュールのデータ量によって，予備リンク，リンク，カーネルへのロードに要する時間は，それぞれ回帰分析により求められる式 (1)(2)(3) で表される一次関数に近似できる増加傾向を示すことがわかった．

$$y = 28.573x + 339.39 \quad \cdots \text{式 (1)}$$
$$y = 27.801x + 371.03 \quad \cdots \text{式 (2)}$$
$$y = 16.135x + 12.068 \quad \cdots \text{式 (3)}$$

(x：データ量 (4KB) y：処理時間 (msec))

さらに上記の式から，モジュールのデータ量をあらかじめ知っておくことにより，DKM 組み込みに要する時間が予測可能である．特に，RT-Mach のような実時間性をある程度保証するリアルタイム OS では，処理時間の予測は重要であると考える．

一方で，今回の測定では，既に述べたサーバの機能を実現する処理部分のオーバーヘッドを全く考慮していない．従って，実際にアプリケーションの要求によって動的にモジュールの入れ換えなどを行なう場合には，DKM ロードの前にサーバが，(1) 既にロードされている不要な DKM をアンロードし，(2) 依存関係を調べて他に必要な DKM があればロードし，(3) 新たにロードする DKM のインターフェースが正しいことを検査する，といった処理にかかる時間を含めて考えなければならない．

5 まとめ

本稿では，動的ロード，アンロードが可能なオブジェクトモジュールを用いた DKM アーキテクチャのロードに際する時間的コストの評価を行なった．DKM 内のデータ量からロードに要する時間的予測が可能であり，特に RT-Mach のようなリアルタイム OS には重要であることを述べた．今後の課題として，DKM アーキテクチャにおいてサーバが実現する機能の処理部分についての性能評価が必要である．

参考文献

[1] Hideyuki Tokuda,Tatsuo Nakajima,Prithvi Rao,"RT-Mach: Towards a Predictable Real-TimeSystem" InProceedings of USENIX Mach Workshop, October 1990.

[2] 追川修一，杉浦一徳，西尾信彦，徳田英幸，マイクロカーネルにおけるカーネルモジュールのサポート．第 53 回情処全大論文集，1996.

[3] 追川修一，喜多山卓郎，徳田英幸，スクリプト言語によるカーネルレベルでの動的適応．マルチメディア，分散，協調とモーバイルワークショップ，1997.