5D-5 wrapper型資源予約機構の実装と検証

梶原 史雄　盛合 敏
NTT 情報流通プラットフォーム研究所

1　はじめに

アプリケーションのバグや DoS 攻撃 (Denial of Service Attack) による資源の浪費に対して, アプリケーションの実行に必要な資源を予約できる資源予約機構は有効に働く. しかし, 従来の予約ドメイン [1] を利用する統合的資源予約機構は新規にアプリケーションを作成することを前提としているため, 既に普及しているアプリケーションに対して資源予約機構を適用することができない. 本論文では, 予約ドメインを既存のアプリケーションに対して適用できるようにアプリケーション単位で資源の予約が可能な wrapper 型資源予約システムを提案する. この wrapper 型資源予約システムを RT-Mach 上に実装し, 代表的なアプリケーションである Apache を用いて評価を行い, 本システムの有効性を示す.

2　細粒度資源予約の問題点

従来の予約ドメインを用いた統合的資源予約機構ではスレッド, ファイル, ソケットなどを最小単位とする細粒度の資源予約を行うことができる [2]. 例えば, 1つの WWW サーバアプリケーションで複数の仮想サイトを作成したり, 複数のユーザで利用したりする際に仮想サイトごと, あるいはユーザごとに予約ドメインを設定して, 個々の予約ドメインに対して資源予約を行うことができる.

但し, このような統合的資源予約機構では上記のような資源予約を行うアプリケーションを新規に作成することを前提としており, 既存のアプリケーションに適用することを考慮されてはいない. このような既存のアプリケーションの変更を行なう際には二つの問題点がある. まず, バイナリ配布アプリケーションの問題である. プログラムソースが公開されているアプリケーションであれば資源予約を行うように変更できるが, バイナリ配布されるアプリケーションは変更できない. 第二にドメイン分割の問題である. プログラムソースが公開されているアプリケーションでもアプリケーションの予約ドメインをどのように分割するかという問題がある. 細分化しすぎると, アプリケーション全体で予約する資源の全体像が把握しづらくなる.

アプリケーションの変更なしに細粒度の資源の予約を行うにはアプリケーションから OS に対して行われるプロセスの作成, ファイルに対するファイル・ディスクリプタの作成, ネットワークソケットの作成などに対して意味付けをして, 各々をどの予約ドメインに属させるかを自動的に行う必要がある. しかし, 通常のアプリケーションではファイル・ディスクリプタやソケットの生成は頻繁に行われる. これらについて意味付けを行うのは困難である.

3　wrapper 型資源予約機構

本論文ではアプリケーション全体を一つの予約ドメインとして扱うことで, アプリケーションの変更なしに資源の予約を行うことを目指す.

アプリケーション全体は通常単一の主プロセスあるいは単一の主プロセスから作成される複数の副プロセスで構成される. そこで, 予約を行うアプリケーションの主プロセスに対して予約ドメインを定義し, 主プロセスから作成される副プロセスはすべて同じドメインに属するようにする. また, 特定の予約ドメインに属さない (予約を行うアプリケーションを構成するプロセスではない) プロセスは予約ドメインに対して予約された資源の残りを使って動作するようにする.

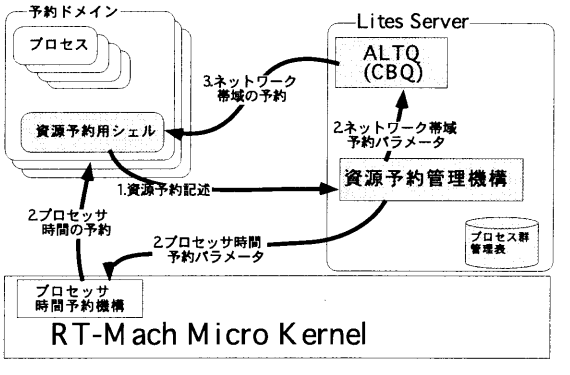

図 1: wrapper 型資源予約機構の RT-Mach+Lites への実装

本論文では前述のようにプロセス群を管理し, 資源予約機構を利用してアプリケーションに対して資源を予約する wrapper 型資源予約機構を RT-Mach NR2[3] 上の FreeBSD 互換サーバの Lites(以下, RT-Mach+Lites) に実装した.

実装するプラットホームとして RT-Mach を選択した理由は RT-Mach がマイクロカーネルであるためである. マイクロカーネルでは OS の大部分を占める

ネットワークプロトコル処理部やファイルシステムをユーザモードで実行されるOSサーバで処理する．このため，カーネルモードで処理される時間がモノリシックカーネルである他のUNIX系OSに比べて短かくてすみ，予約されたプロセッサ時間の保証や優先度制御がより行いやすいという長所がある．

このRT-Mach+Lites上への実装の構成を図1に示す．構成要素のうち，資源予約用シェルはwrapper型資源予約機構の予約ドメインであるプロセス群を構成する基になり，同時に予約ドメインに対する資源の予約を資源予約管理機構により行う．資源予約管理機構はプロセス群管理表の管理を行う機能と資源の予約を資源予約機構を用いて行う機能とを持つ．

4 wrapper型資源予約機構の検証

RT-Mach+Lites上に実装したwrapper型資源予約機構がバグやDoS攻撃による資源浪費に対して有効であることを代表的サーバアプリケーションであるApacheを用いて検証する．実験では表1のようなサーバホストとクライエントホストを接続し，WebBench[4]というベンチマークアプリケーションを用いてクライエント側からみたスループット [req/sec] を測定する．

表1. 各ホストの性能

	サーバホスト	クライエントホスト
CPU	Celeron300A	Celeron300A
メモリ	256Mbytes	128Mbytes
NIC	FastEtherNIC(fxp)	FastEtherNIC(fxp)
OS	FreeBSD-3.4R FreeBSD-2.2.8R Linux-2.2.13 RT-Mach+Lites (+wrapper型資源予約システム)	WindowsNT-4.0

実験ではプロセッサ時間とネットワーク帯域という資源を浪費する妨害プロセスを複数個作り，これらの妨害プロセスにnice値-20を与えたときのApacheサーバのスループットを測定する．また，比較のため，wrapper型資源予約システムを実装していないRT-Mach+Lites，代表的UNIX系OSとしてFreeBSD-2.2.8R, FreeBSD-3.4R, Linux-2.2.13についても同様の測定を行う．なお，wrapper型資源予約機構ではApacheサーバに対してプロセッサ時間を20%，ネットワーク帯域を30%予約した．この結果を図2に示す．

図2よりwrapper型資源予約機構を実装したRT-Mach+Litesでは妨害プロセス数を32としてもApacheサーバのスループットはほぼ低下しておらず，wrapper型資源予約機構によるApacheサーバに対する資源予約は有効に働いていることがわかる．

これに対して他のOSでは少からず妨害プロセスの影響を受ける．モノリシックカーネルであるLinuxとFreeBSDでは妨害プロセスの増大による影響が大

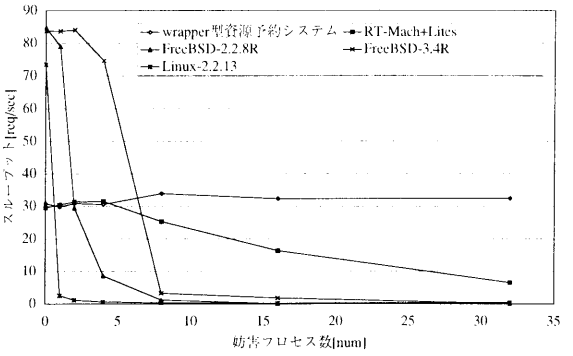

図2: OSによる妨害プロセスに対する耐性

きく，特にLinuxでは妨害プロセス数が1つでもあるとほぼスループットが0となって今回比較したOSの中では資源浪費に対する耐性が最も弱いことがわかる．他方，マイクロカーネルであるRT-Mach+Litesでは妨害プロセスの影響を受けるものの，モノリシックカーネルであるOSに比較して影響が軽いこともわかる．

5 おわりに

本論文では資源予約機構に対応していない既存のアプリケーションに対して資源予約を行なうためのwrapper型資源予約機構を提案し，RT-Mach+Lites上に実装した．

また，実装したwrapper型資源予約機構のDoS攻撃などの資源浪費に対する有効性をApacheサーバを用いて検証した．検証の結果，他のUNIX系OSでApacheサーバのスループットがほぼ0となる資源浪費を行うプロセス数が32個とした場合でも，wrapper型資源予約機構ではスループットの低下がみられず，資源浪費に対して効果があることを示せた．

参考文献

[1] G.Banga, P.Druschel, and J.Mogul, "Resource containers: A new facility for resource management in server systems.", In Proceedings of the USENIX 3rd Symposium on Operating System Design and Implementation New Orleans, LA, October 1999, Feburary 1999.

[2] J. Blanquer, J. Bruno, E. Gabber, M. Mcshea, B. ESC,AVzden, and A. Silberschatz, "Resource Management for QoS in Eclipse/BSD.", In Proceedings of the FreeBSD 1999 Conference, Berkeley, California, October 1999.

[3] "Real-Time Mach NTT Release", http://info.isl.ntt.co.jp/rtmach/

[4] "WebBench 3.0" http://www.zdnet.com/zdbop/webbench/