MKngプロジェクトにおける単一アドレス空間アーキテクチャ: 単一仮想記憶型OSにおけるIPCの高速化手法†

3Z－4

寺本 圭一　　岡本 利夫

(株) 東芝　研究開発センター　情報・通信システム研究所

1 はじめに

近年，64-bit版プロセッサの出現により，広大な仮想アドレス空間資源をより有効に活用し，大規模DBへの適用や高性能サーバシステムの効率改善に役立てるための研究が盛んに行われている．本システムでは，従来システムでいうところのプロセスやデータを全て同一のアドレス空間上に配置するという単一アドレス空間アーキテクチャを採用することによって，今後急速に増加するであろう64-bit化されたプロセッサにおいて，広大な仮想アドレス空間上で複数のプログラム間のデータ送受や実行を高速に行う環境を提供することを目標としている．

本稿では，こうした単一アドレス空間上に配置される複数のプログラム間でスレッドによる通信を行う際の一最適化手法について述べる．これは，保護ドメインやスレッドモデルの拡張，ダイナミック・リンカの協調などにより実現される．

本研究は，次世代マイクロカーネル(MKng)研究プロジェクト[1]のサブテーマの一つである．

なお，本システムはマイクロカーネルアーキテクチャを採用しているが，従来の多重アドレス空間を使用したシステムとは多くの相違点を持ち，また，単一アドレス空間アーキテクチャの特性を活かしてシステムの最適化をはかることを第一の目標としているため，MKngにてベースとしているマイクロカーネル(Real－Time Mach)の特徴は一部利用しているが，実装はこれとは別に独自に行っている．

2 単一仮想記憶空間における保護空間

本システムでは，単一仮想記憶空間上で走行するスレッド毎に参照可能なメモリ領域集合に対する保護属性が個別に設定できるという保護ドメインモデルを採用している．これにより，個々のスレッドによるデータアクセスやプログラム実行に関する保護を安全かつ柔軟な形で実現する枠組みを提供している．

このように，単一の仮想アドレス空間を提供し，必要に応じて複数の保護空間を設定可能にしているが，一般に保護空間を切替える際には，既存のMMUやキャッシュアーキテクチャの種類によっては，TLBフラッシュやページテーブルの切替え，キャッシュラインのフラッシュなどが必要となる．

すなわち，異なる保護ドメインを有するスレッド間で通信を行う場合には，こうした保護空間の切替えコストを伴うことになる．

3 保護ドメインの拡張によるIPC

本システムでは，保護ドメインは，1つ以上のスレッドによって共有されうる．各スレッドは別々のスタックを保持するが，保護ドメインは同一のものを共有することが許される．これは，多重仮想記憶方式を採用している既存システムにおける一つのプロセス空間内を走行するユーザレベルスレッドと同様の扱いである．既存システムでは，基本的にスレッドが直接アクセス可能な領域はそのユーザプロセス空間内に限定されており，他プロセス空間中のプログラムを直接参照することはできない．本システムでは，離散配置される複数の領域集合をアクセス可能な保護ドメインと規定している．同一保護ドメイン内に存在する複数スレッド間で通信を行う場合には，その保護空間を切替える必要は無い．この性質を利用し，本来保護ドメインには含まれていない外部プログラム領域内に実装されたメソッドを，呼び出し元のスレッドが保持する保護ドメインに付加させて，ダイレクトにバインド，実行する機構を提供する．これにより，従来異なるプロセス空間中にあるプログラム間で通信するときのような空間切替え/スレッド切替えが不要となり，呼び出し元スレッドによる一連のRPC的な処理による通信(メソッド呼び出し)が可能となる．

4 ダイナミック・リンカの適用

通常，共有ライブラリを使用する場合，コンパイラ(リンカ)は，プログラムから使用されるシンボルをあらかじめチェックし，必要な共有ライブラリの情報などを実行ファイル中のリンク編集情報部に記録する．プログラムは実行中に必要に応じてダイナミック・リンカの助けを借りて共有ライブラリを動的にマップ，バインディング

A Single Address Space Architecture in the MKng Project: An Optimization of IPC in a Single Virtual Address Space Operating System

Keiichi TERAMOTO and Toshio OKAMOTO
Communication and Information Systems Research Labs., R&D Center, TOSHIBA CORPORATION
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan
E-Mail: <teramoto@isl.rdc.toshiba.co.jp>

†この研究は，情報処理振興事業協会（IPA）が実施している創造的ソフトウェア育成事業「次世代マイクロカーネル研究プロジェクト」のもとに行われた．

するなどして実行を続ける．ダイナミック・ローディングの場合は，あらかじめシンボルのチェックをすることなく，プログラム中の任意の時点で共有ライブラリを指定して呼び出すことが可能となっている．これらはいずれも共有ライブラリのリンクに関するものであるが，localな計算機環境下で動的に複数のプログラムを単一の仮想記憶空間中にロード，マップ，リンクするといった操作を行う場合にも同様の機構が適用できる．

5　ネームサーバの導入

あるプログラム領域 (text 領域) 中にあるメソッドを，呼び出し元のスレッドが保護ドメインを拡張することによって，ダイレクトにアクセス/実行できるようにするために必要な環境をネームサーバの導入によって実現する．

一般に，コンパイラ (リンカ) は，実行プログラムを直接リンク対象とすることはないため，実行プログラム内のシンボル情報をプログラム外部へ公開することはない．そこで，通信相手となるプログラム間で共有のネームサーバを設置し，そこへ公開したい (可能な) メソッドを登録させておいて，これに対する問い合わせを呼び出し元スレッド側から行って獲得したいシンボルの情報を取得し，ダイナミック・ロード/リンクを介した後，実際のメソッドを呼び出すという機構を用意する．具体的にネームサーバに登録される情報は，“公開メソッド名”，“メソッドの開始位置 (仮想アドレス)”，“アクセス保護属性” などの組から構成される．単一仮想アドレスアーキテクチャを採用しているため，仮想記憶中に配置されたプログラムの位置は一意に決定することができる．ネームサーバへの問い合わせにより必要な参照の獲得に成功すると，アクセス対象となるメソッドを含むプログラム領域が呼び出し元スレッドの保護ドメインに追加され，それ以降はダイナミック・リンカと同様なリンク作業を行うことになる．

図 1では，公開関数として登録されている func1() をネームサーバに問い合わせて，呼び出し元の保護ドメインとして追加しようとしている例を示している．

なお，図 1に示しているように，実際にネームサーバに登録される対象は関数に限定されていない．任意のデータやシンボルを持たないオブジェクト位置 (例：バッファ中の 100 バイト目など) をも登録することが可能である．

この特性を利用し，共有メモリ内に配置されたデータへのアクセス手段も提供できる．

6　おわりに

本稿では，あるプログラム上で走行しているスレッドが保持する保護ドメインに対して，単一仮想記憶空間上に配置された他のプログラム領域のアクセス許可属性を追加することによって，呼び出し元スレッドによるダイレクトな RPC を可能にし，スレッド切替えや保護空間の

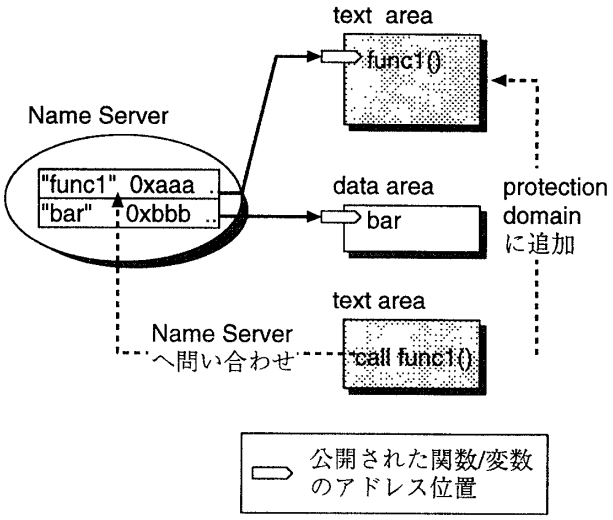

図 1: ネームサーバへの公開関数 func1() の問い合わせと呼び出し元保護ドメインへの追加

切替えコストの無い，高速な実行を提供する手法について述べた．この際，一意にアドレッシング可能という単一アドレス空間方式の性質を利用し，ネームサーバを介した公開関数の登録/参照機構を提供することによって，ダイナミックなバインド処理を実現している．なお，ネームサーバにより登録/参照可能な対象は，関数に限定されず任意のオブジェクトに適用可能なため，データ共有などの際にも適用できる．

参考文献

[1] 徳田 他: “MKng: 次世代マイクロカーネル研究プロジェクトの概要,” 第 55 回情処全大論文集, 1Z-2 (1997).

[2] Okamoto et al., “A Micro Kernel Architecture for Next Generation Processor”, pp.83-94, Microkernels and Other Kernel Architectures Symposium, USENIX, 1992/4.

[3] T. Wilkinson et al., “Compiling for a 64-Bit Single Address Space Architecture”, Technical Report TCU/SARC/1993/1, Systems Architecture Research Centre, City University, London, 1993/3.

[4] Jeffrey S. Chase et al., “Sharing and protection in a single address space operating system”, ACM Transactions on Computer Systems, 1994/5.

[5] “SunOS 5.3 Linker and Libraries Manual”, SunSoft, 1993.

[6] Jeffrey S. Chase et al., “Lightweight shared objects in a 64-bit operating system”, Technical Report 92-03-09, University of Washington, 1992/3.