2007－OS－104 (5)
2007／1／30

AnT オペレーティングシステムにおけるメモリ領域管理の設計と実現

梅本昌典† 田端利宏†
乃村能成† 谷口秀夫†

AnT オペレーティングシステムでは，マイクロカーネル構造を基本構造としている．これにより，多種多様なサービスを提供可能としている．しかし，マイクロカーネル構造を有することにより，頻発するプロセス間通信のデータ授受によるオーバヘッド削減が課題となる．そこで，本稿では，メモリ領域管理の設計，実装および評価について述べる．実現したメモリ領域管理により，プログラム間の保護を実現でき，プロセス間通信を高速化できる．さらに，高速な OS の再立上げ処理を実現できる．また，これらの効果を明らかにするため，データ授受時間と OS の再立上げ時間を評価した．

Design and Implementation of Memory Management for *AnT* Operating System

Masanori Umemoto,† Toshihiro Tabata,† Yoshinari Nomura†
and Hideo Taniguchi†

AnT is an operating system based on micro-kernel architecture. Therefore, ***AnT*** can provide various kinds of service. However, there is a problem about a data transferring overhead caused by a lot of inter-process communications. Therefore, we need to examine memory management so that ***AnT*** can transfer data with low overhead . In this paper, we present design, implementation and evaluation of memory management. Resulting from the design and the implementation, ***AnT*** can realize memory protection from each program, high speed inter-process communication and high speed rebooting. We evaluate data transferring and rebooting.

1. はじめに

計算機の進歩に伴い，多種多様なサービスの提供や高速なデータ通信が OS に要求されている．

そこで，我々は，***AnT*** オペレーティングシステム (以降，***AnT*** と呼ぶ) を開発している[1)2)]．***AnT*** は，新しいハードウェアやサービスへの適応制御機能を持つ OS である．***AnT*** では，デバイスドライバをプロセスとして動作させることを可能としているため，マイクロカーネル構造を基本構造としている．これにより，多種多様なサービスを提供可能としている．しかし，高速なデータ通信については，マイクロカーネル構造を有することにより，頻発するプロセス間通信のデータ授受によるオーバヘッド削減が課題となっている．このため，メモリ領域管理[3)] の機能を工夫し，プロセス間通信によるオーバヘッドを最小限に抑える必要がある．

本稿では，***AnT*** におけるメモリ領域管理の設計と実現について述べる．まず，***AnT*** の基本構造について述べる．次に，メモリ領域管理の設計について述べ，最後に，実装と評価について述べる．具体的には，メモリ領域管理への要求を示し，メモリ空間の構成，メモリ領域管理の機能，および，要求への対処について述べる．また，プログラム間でのデータ授受時間，およびリブート時間の評価について述べる．

† 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

2. *AnT* の基本構造

AnT の基本構造を図 1 に示し，以降に説明する．

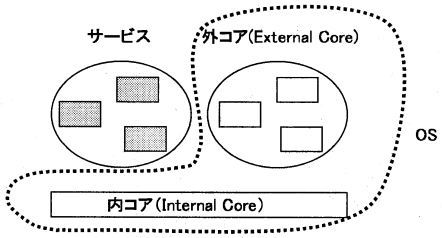
プログラムは，OS とサービスからなる．OS は，内コアとプロセスとして動作する外コアからなる．サービスは，利用者の要求するサービスを提供するプロセスからなる．各々のプログラムについて，以下に説明する．

(1) 内コア

システムの最小限な機能の動作を保証するプログラム部分である．主な機能として，メモリ領域管理，プロセスの実行権制御部がある．

(2) 外コア

適応したシステムに必須なプログラムであり，動的



(1)OS:内コア＋外コア
・内コア:システムの最小限な機能の動作を保証する部分
・外コア:適応したシステムに必須な部分(動的再構成構造)
(2)サービス:利用者によって要求される機能を提供する部分

図 1　***AnT*** の基本構造

に再構成可能な構造を有する．主な機能として，デバイスドライバやファイル管理がある．

(3)　サービス

利用者の要求する機能を提供する部分である．

3. メモリ領域管理の設計

3.1 要　　求

メモリ領域管理への要求として，以下のものがある．

(要求 1)　プログラム間の保護

内コアとその他のプログラム間の保護は，プログラム走行モードをスーパバイザモードとユーザモードに分離することで行うが，内コア以外のプログラム間の保護は，メモリ領域管理が行う必要がある．具体的には，ひとつのプログラムの不具合が他のプログラムへ悪影響を及ぼさないようにする対処が必要である．

(要求 2)　プログラム間通信の高速化

データのメモリ間複写は，OS の性能に大きな悪影響を与える．特に，***AnT*** のようにマイクロカーネル構造を有する OS においては，プロセス間通信が多発するため，プロセス間通信の際のデータ授受を高速化することが重要である．もちろん，プロセスと内コア間のデータ授受を高速化することも必要である．

(要求 3)　リブート機能の支援

動作中に不具合が発生した場合，その再起動は速やかに行うことが求められる．このため，メモリ上に存在する OS プログラムを利用したリブート機能 (以降では，リスタート機能と呼ぶ) は有効であり，この機能実現に向け，メモリ領域管理は何らかの支援を行う必要がある．

3.2 メモリ空間構成と機能

3.1 節で述べた要求を満足するため検討を行い，メモリ領域管理を設計した．以降に，メモリ空間の構成および機能について説明し，その後，3.3 節において，要求と対処の関係を説明する．

3.2.1 メモリ空間の構成

メモリ空間の構成を図 2 に示し，以下に説明する．

＜実メモリ空間＞

I/O hole の後に，カーネル（Text 部），領域管理用域，各モジュール用領域，コア間通信データ域（ICA : Inter-process Communication Area），4KB ページ域（PGA : PaGe Area）を管理する．各領域のアクセス権の違いから，領域管理用域の先頭アドレスは 4MB の整数倍の位置とした．なお，カーネルと領域管理用域の間の領域および各モジュール用領域とコア間通信データ域の間の領域は，論実アドレス連続域（SMA : Straight Mapping Area）として利用する．

各領域の特徴や用途については，後述する．

＜仮想メモリ空間＞

0 から 3GB をプロセス空間，3GB から 4GB をカーネル空間と呼び，プロセス空間を多重仮想空間とする．

プロセス空間の 0 から 2GB の領域は，PGA を利用した 4KB 単位マッピングであり，プロセスのプログラムを格納する領域である．

プロセス空間の 2GB から 3GB の領域は，ICA を利用した 4KB 単位マッピングであり，プロセス間のデータ授受を行う領域である．

カーネル空間の領域は，アクセス権の違いから，カーネルと領域管理用域以降に分割し，4MB の整数倍を境界とした．カーネルは 4MB 単位マッピングである．また，領域の確保と解放を自由に行える領域，つまり初期化時ではなくサービス提供中にカーネルが必要となる領域として，SMA と PGA を用意した．SMA の領域は，4MB 単位マッピングと 4KB 単位マッピングの境界を含んでいる．なお，SMA と PGA の違いは，物理アドレスと論理アドレスの連続性を 4KB を超えて保証しているか否かである．I/O hole は，参照更新が必要であるため，PGA の直上の位置に配置した．

3.2.2 機　　能

メモリ領域管理の機能として，管理する領域を説明し提供インタフェースを述べる．

＜領域の管理＞

用途に合わせて，領域を 5 つに分割して管理する．各領域について，以下に説明する．

(1)　領域管理用域

領域管理用域は，OS の初期化処理時に設定し利用する情報を格納する領域である．例えば，初期化処理時に利用するページディレクトリやページテーブル，および各モジュール用領域に存在する各モジュールの管理情報域の先頭アドレス情報である．このため，格

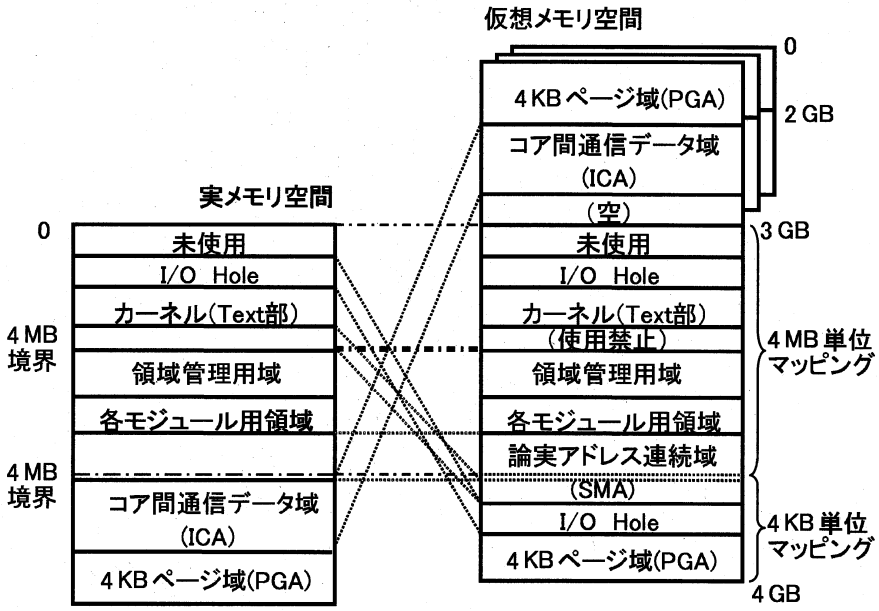


図 2 メモリ空間の構成

納された情報は初期化後にも利用される．しかし，初期化処理後に領域の大きさが変化することはない．

(2) 各モジュール用領域

各モジュール用領域は，内コアの各モジュールの管理情報を格納する領域である．この領域は，各モジュールの初期化処理で確保され，初期化処理後に大きさが変化することはない．

(3) 論実アドレス連続域 (SMA : Straight Mapping Area)

SMA の領域は，前部分が 4MB 単位マッピング，後ろ部分が 4KB 単位マッピングされており，各部分において，4KB を超える領域の物理アドレスと論理アドレスの連続性を保証している領域である．この領域は，内コアのモジュールが利用する領域を増やしたい場合に利用する領域である．

(4) コア間通信データ域 (ICA : Inter-process Communication Area)

ICA は，2GB から 3GB の空間において，4KB を超える領域の物理アドレスと論理アドレスの連続性を保証している領域である．なお，マッピングは 4KB 単位で行われる．したがって，プロセスと内コアのモジュールがデータ授受用の領域として利用する．また，ICA は，プロセス間で共用可能な空間に位置づけられるため，プロセス間の高速なデータ授受を可能としている．

(5) 4KB ページ域 (PGA : PaGe Area)

PGA は，0 から 2GB の空間と，読み書き可能な I/O Hole 以降の空間において，4KB 単位でマッピングされる領域である．したがって，プロセスや内コアのモジュールが利用する領域を増やしたい場合に，PGA を利用する．なお，SMA や ICA と異なり，4KB を超えて物理アドレスと論理アドレスが連続である保証はない．

＜提供インタフェース＞

各領域の提供インタフェースを表 1 から表 5 に示し，以下に説明する．

(1) 領域管理用域

表 1 に示すように，内コアの各モジュールが持つ管理情報域の先頭アドレス情報を格納している領域の先頭アドレスを取得する機能を提供する．内コアの各モジュールは，管理情報を取得するため，この機能を用いる．

(2) 各モジュール用領域

表 2 に示すように，各モジュール用領域を確保する機能を提供している．内コアの各モジュールは，初期化処理で，この機能を用いて，各モジュール用領域を確保し，確保した領域に管理情報を格納する．なお，この機能は，初期化終了後は利用できない．

(3) 論実アドレス連続域

表 3 に示すように，SMA の確保，解放，貼り付け，

表 1　領域管理用領域の提供インタフェース

形式	概要	提供先モジュール		
		外コアとサービス	内コアのモジュール	メモリ領域管理の内部モジュール
osareaheadaddr( )	各モジュール用領域の管理情報域の先頭アドレス情報を格納している領域の先頭アドレスを取得する	-	○	○

表 2　各モジュール用領域の提供インタフェース

形式	概要	提供先モジュール		
		外コアとサービス	内コアのモジュール	メモリ領域管理の内部モジュール
getosarea(size)	size バイト分の領域を確保する	-	○	○

剥がし，作成，および削除の 6 つの機能を提供している．ただし，作成は，確保と貼り付けを組み合わせた機能であり，削除は，解放と剥がしを組み合わせた機能である．ここで，内コアのモジュールが利用する領域を増やしたい場合に，SMA を利用するため，内コアに作成と削除の機能を提供している．作成と削除以外の機能は，内コアのモジュールやプロセスから利用されないため，メモリ領域管理にのみ提供している．なお，確保と解放の機能は，管理領域を適切に管理するため，メモリ領域管理のみ利用可能としている．したがって，SMA，ICA，そして PGA の確保，解放では，共通のインタフェースを提供している．これにより，SMA を確保する場合は，オペレータ op を用いて，明示的に SMA を確保するように指定する．なお，SMA を確保する場合，flag は，ICA の確保時に利用するため，利用しない．

(4)　コア間通信データ域

表 4 に示すように，ICA の確保，解放，貼り付け，剥がし，作成，および削除の 6 つの機能を提供している．ただし，作成は確保と貼り付けを組み合わせた機能であり，削除は解放と剥がしを組み合わせた機能である．ここで，プロセスと内コアのモジュールは，データ授受用の領域として，ICA を利用するため，作成と削除の機能を内コアのモジュールとプロセスに提供している．また，内コアのモジュールによって，プログラム間通信が行われるため，内コアのモジュールへ貼り付けと剥がしの機能を提供している．なお，ICA の領域を適切に管理するため，確保と解放の機能をメモリ領域管理にのみ提供している．また，ICA を確保する場合は，op を用いて，明示的に ICA を確保するように指定する．

ICA では，確保した領域をひとつの単位として管理する．このため，確保した単位で，貼り付け，剥がし，および解放を行う．ただし，利用可能な ICA の総量は，OS 立上げ時に決定するため，効率的に管理する必要がある．そこで，利用期間の長短を指す flag を導入し，利用時間の長い領域と短い領域が混在する外部断片化を抑制した．現在，この flag は，ICA の作成時に，プロセスが設定している．

(5)　4KB ページ域

表 5 に示すように，PGA の確保，解放，貼り付け，剥がし，作成，および削除の 6 つの機能を提供している．ただし，作成は，確保と貼り付けを組み合わせた機能であり，削除は，解放と剥がしを組み合わせた機能である．ここで，内コアのモジュールが利用する領域を増やしたい場合やプロセスのプログラムを格納したい場合に，PGA を利用するため，内コアのモジュールに PGA の作成と削除の機能を提供している．また，内コアのモジュールによって，プロセス間のプログラム共有を実現するため，内コアのモジュールに PGA の貼り付けと剥がしの機能を提供している．さらに，PGA の領域を適切に管理するため，PGA の確保と解放は，メモリ領域管理にのみ提供している．ここで，PGA を確保する場合は，op を用いて，明示的に PGA を確保するように指定する．なお，4KB のページ単位で PGA を管理しているため，確保の引数である size と flag，および解放の引数である size は，利用されない．

3.3　期待される効果

要求への対処状況および設計により期待される効果について以下に説明する．

(1)　プログラム間の保護

ひとつのプログラムの不具合による他のプログラムへの影響を抑制するため，仮想メモリ空間の 0 から 3GB までの空間を多重仮想空間とした．これにより，プロセス毎に異なる空間で動作可能とし，(要求 1) を

表 3　論実アドレス連続域の提供インタフェース

形式	概要	提供先モジュール		
		外コアとサービス	内コアのモジュール	メモリ領域管理の内部モジュール
getmem(op, size, flag)	size バイト分の領域を確保する	-	-	○
freemem(raddr, size)	raddr で指定された領域を解放する	-	-	○
attachsma(vmid,vaddr, prot)	prot で指定した保護情報で仮想空間識別子 vmid と論理アドレス vaddr で指定した領域を貼り付ける	-	-	○
detachsma(vmid, vaddr)	仮想空間識別子 vmid，論理アドレス vaddr で指定された領域を剥がす	-	-	○
createsma(size)	size バイト分の領域を SMA に作成する	-	○	○
deletesma(vaddr, size)	論理アドレス vaddr から始まる size バイト分の領域を削除する	-	○	○

表 4　コア間通信データ域の提供インタフェース

形式	概要	提供先モジュール		
		外コアとサービス	内コアのモジュール	メモリ領域管理の内部モジュール
getmem(op, size, flag)	size バイト分の領域を flag で指定された方法で確保する	-	-	○
freemem(raddr, size)	raddr で指定された領域を解放する	-	-	○
attachica(vmid, vaddr, prot)	仮想空間識別子 vmid と論理アドレス vaddr で指定した領域を prot で指定した保護情報で貼り付ける	-	○	○
detachica(vmid, vaddr)	仮想空間識別子 vmid，論理アドレス vaddr で指定された領域を剥がす	-	○	○
createica(size, flag)	flag で指定された方法で size バイト分の領域を作成する	○	○	○
deleteica(vaddr)	論理アドレス vaddr から始まる領域を削除する	○	○	○

表 5　4KB ページ域の提供インタフェース

形式	概要	提供先モジュール		
		外コアとサービス	内コアのモジュール	メモリ領域管理の内部モジュール
getmem(op, size, flag)	size バイト分の領域を確保する	-	-	○
freemem(raddr, size)	raddr で指定された領域を解放する	-	-	○
attachpage(vmid, vaddr, raddr, prot)	仮想空間識別子 vmid と論理アドレス vaddr で指定した領域を raddr と prot で指定した保護情報で貼り付ける	-	○	○
detachpage(vmid, vaddr)	仮想空間識別子 vmid と論理アドレス vaddr で指定された領域を剥がす	-	○	○
createpage(vaddr, size, prot)	論理アドレス vaddr から size バイト分の領域を作成する	-	○	○
deletepage(vaddr, size)	論理アドレス vaddr から始まる size バイト分の領域を削除する	-	○	○

満足させた．

(2)　プログラム間通信の高速化

(要求 2) を満足するため，プロセス空間の領域を対象に，貼り替えによるデータ授受を支援している．ここで，貼り替え処理は，貼り付け，剥がしの処理からなる．貼り替えでは，データのメモリ間複写を削減するため，マッピングを変更することにより，データ授受を行なう．さらに，ICA の領域において，貼り替え時と入出力処理時の論実アドレス変換処理を高速化した．具体的には，全ての仮想空間において，ICA を予約した領域としている．さらに，ICA では，物理アドレスと論理アドレスの対応を固定しているため，論実

アドレス変換処理を高速化できる．つまり，ICA では，論理アドレスから物理アドレスへの論実アドレス変換を，該当論理アドレスから ICA の先頭論理アドレス (2GB) を引き，この値に，ICA の先頭物理アドレスを足すことによって行う．一方，PGA では，ページディレクトリ，ページテーブルの 2 段のアドレス変換表を参照して論実アドレス変換を行う．したがって，ICA では，ページ貼り付け時におけるアドレス変換表の参照オーバヘッドを削減できる．

また，ICA では，論理アドレスと物理アドレスの対応を固定しているため，ICA の領域を貼り替える場合，送信元プロセスと同じ論理アドレスへ貼り付ける．これにより，送信先プロセスの貼り付け先の領域を探索する手間を省略できる．したがって，メモリ領域管理への要求である高速なプログラム間通信を実現できる．

(3)　リブート時間の短縮

(要求 3) のため，メモリ領域管理では，内コアの各モジュールのデータ用領域として，各モジュール用領域を提供し，OS の実行モジュールを Text 部のみで実行することを可能にした．これにより，内コアの各モジュールは，OS 立上げ時に，各モジュール用領域を確保する機能を利用し，自身のデータ用領域を確保し，この領域へ管理情報を格納する．したがって，メモリ上に存在する OS プログラムを利用したリスタート機能を実現可能とした．

(4)　TLB ヒット率の向上

頻繁にアクセスされる内コアでの TLB のヒット率を向上させるため，内コア部分を 4MB 単位マッピングとした．

(5)　アドレス変換テーブルの作成と登録処理の高速化

アドレス変換テーブルの作成と登録処理において，SMA を利用することとし，処理を高速化した．SMA では，物理アドレスと論理アドレスの対応を固定している．このため，仮想メモリ空間作成時において，アドレス変換テーブルの作成，および登録処理で行う論実アドレス変換処理を高速化できた．

4. 実装と評価

4.1 実　　装

アセンブリ言語と C 言語を用いて，メモリ領域管理を実現した．ここで，メモリ領域管理を実現するために要したソースコードの行数は，

(1)　アセンブリ言語：40 行

(2)　C 言語：2566 行

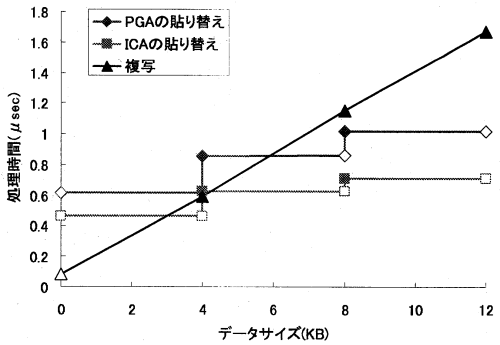


図 3　データ授受に要する処理時間

の計 2606 行である．

4.2 評　　価

4.2.1 評価項目

(要求 2) と (要求 3) への対処の効果を明らかにするため，プログラム間通信の基本処理部分であるデータ授受時間，および OS の再立上げ時間を評価する．

4.2.2 データ授受時間

ICA の領域を貼り替えることによるデータ授受の効果を明らかにするため，ICA の領域の貼り替えに要する処理時間とデータのメモリ間複写によるデータ授受に要する処理時間を測定した．

また，論実アドレス変換の高速化による効果を明らかにするため，加減算による論実アドレス変換を行う場合とアドレス変換表の利用による論実アドレス変換を比較する．具体的には，ICA の貼り替え時間と PGA の貼り替え時間を比較する．なお，比較の精度向上のため，PGA の貼り替え時間は，ICA においてアドレス変換表の利用による論実アドレス変換処理の時間とした．

測定は，Pentium4 2.4GHz の PC/AT 互換機上で行った．データ授受に要する処理時間を図 3 に示し，以下に説明する．

＜貼り替えとデータのメモリ間複写＞

(1)　4 バイトの処理時間

4 バイトの処理時間は，ICA の領域を貼り替えた場合 0.47(μ sec)，メモリ間のデータ複写をした場合 0.08(μ sec) である．

(2)　処理時間の増加傾向

ICA の領域を貼り替えた場合の処理時間は，貼り替え単位が 4KB の整数倍となっているため，4KB の整数倍で処理が増加する．一方，データのメモリ間複写の処理時間は，4B 単位で複写するため，線形で増加する．

(3)　貼り替えとデータのメモリ間複写の比較

データサイズが 3500 バイトまでと 4KB から 4KB+768 バイトまでのデータ授受において，データのメモリ間複写は ICA の貼り替えより高速である．

＜論実アドレス変換＞

(4)　4 バイトの処理時間

4 バイトの処理時間は，PGA の貼り替えの場合 0.61(μ sec) であり，ICA の貼り替えの場合 0.47(μ sec) である．

(5)　貼り替え処理の内訳

貼り替え処理は，貼り付け処理 (論実アドレス変換処理を含む) と剥がし処理からなる．クロックカウンタを用いた測定により，4KB の場合，論実アドレス変換処理の除く貼り付け処理の時間は 0.18(μ sec) であり，剥がし処理の時間は 0.21(μ sec) であった．一方，図 3 より，4KB の場合，ICA の貼り替え処理時間は 0.47(μ sec)，PGA の貼り替え処理時間は，0.61(μ sec) である．したがって，論実アドレス変換処理は，加減算による場合 0.08(μ sec/4KB)，アドレス変換表による場合 0.22(μ sec/4KB) といえる．

(6)　処理時間の増加傾向

論実アドレス変換を加減算によって行う効果は，データサイズが大きくなるほど大きい．

以上のことから，データ量が少ない場合は，貼り替えよりもデータのメモリ間複写が速いものの，4KB を超える付近から，貼り替えが効果的であるといえる．また，論実アドレス変換を加減算によって行う効果は，アドレス変換表による場合に比べ 0.14(μ sec/4KB) 高速であるため，データ量が多いほど有効であるといえる．

4.2.3　再立上げ時間

リブート時間とリスタート時間を比較する．リブートおよびリスタートの処理の流れを図 4 に示し，以下に説明する．

リブート処理では，リブート準備，BIOS 処理，***AnT*** の読み込み，***AnT*** の初期化処理の順に処理する．***AnT*** の初期化処理では，内コアの各モジュールの初期化を行う．一方，リスタート処理では，リスタート準備，***AnT*** の初期化処理の順に処理する．ここで，リスタート準備では，OS の実行モジュールを実メモリ空間上のアドレスと同じアドレスとなるように仮想メモリ空間へマッピングする処理を行う．

リブート処理は図 4 の A から C まで，リスタート処理は図 4 の B から C までを測定した．なお，測定は，Pentium4 2.4GHz の PC/AT 互換機を利用した．測定結果を図 5 に示す．

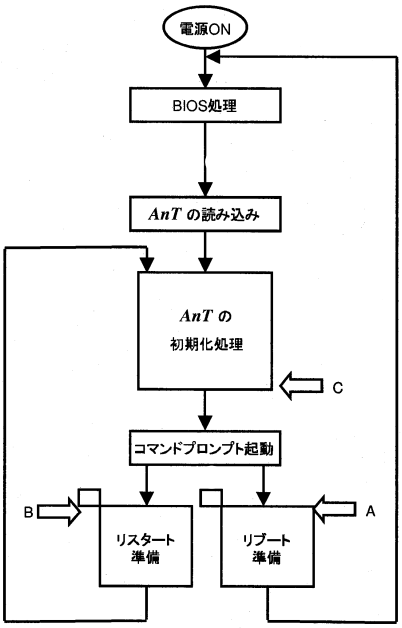


図 4　リブートおよびリスタートの処理の流れ

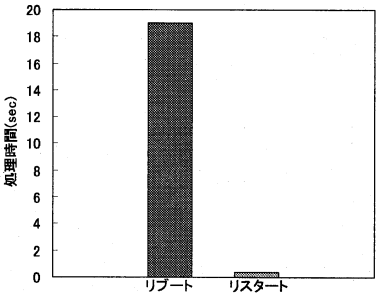


図 5　リブートおよびリスタートに要する処理時間

図 5 から，リブート処理は 19(sec) の処理時間を要し，リスタート処理は 0.4(sec) の処理時間を要す．したがって，リスタートの機能は，OS 再立上げ時間の短縮に対して大きな効果をもたらすことが出来るといえる．

5. ま と め

AnT におけるメモリ領域管理の設計と実現について述べた．メモリ領域管理への要求として，プログラム間の保護，プログラム間通信の高速化，およびリブート機能の支援があることを示した．また，メモリ空間の構成を説明し，領域の用途に合わせて領域を 5 つ (領域管理用域，各モジュール用領域，論実アドレス連続域，コア間通信データ域，4KB ページ域) に分

割して管理することを説明した．さらに，コア間通信データ域を用いたデータの空間間貼り替えによるプログラム間のデータ授受により，プログラム間通信が高速化できることを示した．最後に，各モジュール用領域を内コアの各モジュールに提供し，管理情報を格納する領域を動的に確保できるようにすることで，高速なリスタート処理が実現できることを示した．

評価により，データ量が少ない場合は，貼り替えよりもデータのメモリ間複写が速いものの，4KB を超える付近から，貼り替えが効果的であることを示した．また，論実アドレス変換を加減算によって行う効果は，アドレス変換表による場合に比べ 0.14(μ sec/4KB) 高速であるため，データ量が多いほど有効である．

残された課題として，詳細な評価がある．

謝辞 本研究の一部は，科学研究費補助金 基盤研究（B）「適応性と頑健性を有する基盤ソフトウェアのカーネル開発」（課題番号:18300010）による．

参 考 文 献

1) 谷口 秀夫，乃村 能成，田端 利宏，“***AnT*** オペレーティングシステムの設計”，情報処理学会第 68 回全国大会講演論文集，分冊 1，pp.41-42(2006).
2) 谷口秀夫，乃村能成，田端利宏，安達俊光，野村裕佑，梅本昌典，仁科匡人，“適応性と堅牢性をあわせ持つ ***AnT*** オペレーティングシステム”，情報処理学会研究報告，2006-OS-103，Vol.2006，No.86，pp.71-78(2006).
3) 田端利宏，梅本昌典，安達俊光，谷口秀夫，“***AnT*** オペレーティングシステムのメモリ領域管理”，情報処理学会第 68 回全国大会講演論文集，分冊 1，pp.45-46(2006).