Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

マルチコアプロセッサ上での負荷分散を可能にする *AnT* オペレーティングシステムの開発

井 上 喜 弘[†1]　佐古田 健志[†2]　谷 口 秀 夫[†1]

AnT オペレーティングシステムは，マイクロカーネル構造を有し，シングルコアプロセッサ上で動作する．マルチコアプロセッサにおいて，OS サーバを複数のコア上で動作させることで，OS 機能の負荷分散を実現できる．ここでは，マルチコアプロセッサ向けマイクロカーネル構造 OS の方針と設計について述べる．また，AnT における実現内容として，プログラム構造とサーバプログラム間通信機構について述べる．さらに，評価として，マルチコアプロセッサに対応した AnT のメモリ占有量とサーバプログラム間通信機構の性能について報告する．

AnT operating system Enabling Load Balancing on Multi-core Processor

Yoshihiro INOUE,[†1] Takeshi SAKODA[†2]
and Hideo TANIGUCHI[†1]

AnT is an operating system based on microkernel architecture. ***AnT*** works on a single core processor. In a multi-core processor, load balancing of OS function is realized by execution of os servers on multiple cores. This paper discusses the policy and the design of the microkernel structure OS for multi-core processors. In addition, this paper describes the implementation of the program structure and the inter server program communication mechanism for ***AnT***. Furthermore, the result of evaluations for memory utilization and performance of the inter server program communication mechanism for ***AnT*** that works on a multi-core processor is showed in this paper.

†1 岡山大学大学院自然科学研究科 Graduate School of Natural Science and Technology, Okayama University
†2 岡山大学工学部 Faculty of Engineering, Okayama University

1. はじめに

計算機サービスの多様化，一方ではプログラムの相互干渉による不具合の発生から，オペレーティングシステム (以降，OS) に対し高い適応性と堅牢性が求められている．また，プロセッサの高集積化によりマルチコアプロセッサが普及し，処理の並列実行による OS 機能の負荷分散が可能になった．

高い適応性と堅牢性を実現する OS のカーネル構造として，マイクロカーネル構造[1)–4)]がある．マイクロカーネル構造は，OS 機能の大半をプロセス (以降，OS サーバ) として実現する．このため，OS 機能の追加や削除が容易であり，他の OS 機能の不具合の影響を抑制できる特徴をもつ．また，マルチコアプロセッサにおいて，複数のコア上で OS サーバを動作させることで，OS 機能の負荷分散が可能になる．しかし，これらを実現するには，サーバプログラム間通信の高速化が必要である．

ここでは，マルチコアプロセッサ向けマイクロカーネル構造 OS の開発において，方針と設計を述べ，マイクロカーネル構造を有する ***AnT*** オペレーティングシステム[4)]（以降，***AnT***）における実現内容について述べる．また，実現したマルチコアプロセッサ対応版 ***AnT*** について，OS のメモリ占有量とサーバプログラム間通信性能を報告する．

2. 方針と設計

2.1 方　　針

マルチコアプロセッサ向けマイクロカーネル構造 OS の設計においては，以下の方針に基づいて設計する必要がある．

（方針 1） 各コアの独立動作
（方針 2） 各カーネルの軽量化
（方針 3） 処理の高速化

１つのコアの処理がネックにならないようにして処理全体を高速化するには，コアに閉じた処理を可能にすること，つまり，各コアの独立動作（方針１）を可能にすることが必要である．また，マイクロカーネル構造 OS の特徴である適応性や堅牢性を生かすには，カーネルの軽量化（方針２）を保持することは非常に重要であり，可能な部分については更なる軽量化が必要である．さらに，処理の高速化（方針３）は当然要求される．特に，マイクロカーネル構造 OS の急所であるサーバプログラム間通信速度の低下は可能な限り抑制する必要がある．

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

2.2 設　　計

ここでは，前節の方針に基づき，コアの制御法，プロセスの実行制御法，カーネルの機能分担とプログラム構造，およびサーバプログラム間通信機構について述べる．

コアの制御法には，単一カーネルが全コアを制御する方式 (単一カーネル方式)，および各コアにカーネルを配置し各コアを制御する方式 (マルチカーネル方式) がある．方針１（各コアの独立動作）を満足するには，マルチカーネル方式が良い．さらに，次に述べるプロセスの実行制御法も考慮するとマルチカーネル方式が良い．なお，マルチカーネル方式においても，各カーネルの機能分担により，様々な形態がある．これについては，カーネルの機能分担とプログラム構造として説明する．

プロセスの実行制御法は，単一スケジューラが全プロセスを制御する方式 (単一スケジューラ方式)，および各コアにスケジューラを配置し当該コア上のプロセスを制御する方式 (マルチスケジューラ方式) がある．スケジューラは，カーネルの中で非常に頻繁に動作する処理であり，その高速化が望まれる．したがって，方針３（処理の高速化）を満足するには，マルチスケジューラ方式が良い．

カーネルの機能分担とプログラム構造について述べる．マルチカーネル方式において，全カーネルを同機能とすると，OS のメモリ占有量が大きくなり，方針２（各カーネルの軽量化）を実現できない．一方，各カーネルに固有の機能を持たせると，機能利用時に機能保有カーネルを選択して通信することになり，カーネル間の通信機構つまりサーバプログラム間通信機構が複雑化し，方針３（処理の高速化）を実現できない．そこで，カーネルをマスタカーネル（最初に立ち上がるカーネル）とスレーブカーネルの二つに分類し，機能を分担する．このとき，マスタカーネルは，カーネルに必要な全機能を有する．一方，スレーブカーネルは，方針２（各カーネルの軽量化）を実現するために，スケジューラ機能などに機能を絞り，軽量化を図る．

なお，マスタ／スレーブカーネルのプログラム構造として，手続き部（テキスト部）の共有と個別化を行い，OS のメモリ占有量を抑制して，方針２（各カーネルの軽量化）を実現する．ここで，制御に必要な管理情報（データ部）の構造も工夫する必要がある．カーネル毎に保有するとメモリ占有量が多くなり，方針２（各カーネルの軽量化）を実現できない．また，全カーネルで共有すると操作時の排他制御オーバヘッドが大きくなり，方針３（処理の高速化）を実現できない．そこで，管理情報を次の３つに分類して，方針２（各カーネルの軽量化）と方針３（処理の高速化）を満足させる．一つは，全カーネル共通に１つ存在し，参照は全カーネルが行うものの更新は１つのカーネル（マスタカーネル）に限られるものである．もう一つは，管理のために操作するカーネルは動的に変化するものの，同時に複数のカーネルが操作しないものである．最後の一つは，特定のカーネルのみが操作するものである．このようにすることにより，管理情報のメモリ占有量を抑制でき，方針２（各カーネルの軽量化）を実現できる．また，排他制御を削除できるため，方針３（処理の高速化）を図れる．

サーバプログラム間通信機構は，方針３（処理の高速化）を図るために，プロセス間での複写レスデータ授受は必須である．さらに，排他制御オーバヘッドの削減が必要である．サーバプログラム間通信における主な排他制御箇所は，OS サーバ間での処理依頼と結果返却を管理するキュー操作である．この部分は，各コアが操作するため，全コア間での排他制御（以降，全排他制御と呼ぶ）となり，排他制御オーバヘッドが大きい．この箇所の排他制御オーバヘッドを削減する方法として，OS サーバ間の通信相手を制限する方法がある．例えば，以下の方法である．(方法１）は，１つの OS サーバが受け取る処理依頼は，１つのコアに存在する複数プロセスに限定する方法である．これにより，処理依頼キュー操作での排他制御は，OS サーバが走行するコアと処理依頼を行う複数プロセスが走行するコアの間の排他制御（以降，2 排他制御と呼ぶ）になる．2 排他制御にすることで，全排他制御に比べ，そのオーバヘッドを削減できる．この方法は,「OS サーバへの依頼元プロセスを１つのコアへ配置する」という，プロセスのコア配置に制限を加えるものである．さらに,（方法１）と同様に,（方法２）は，結果返却キュー操作での排他制御を 2 排他制御にする方法である．この方法は,「OS サーバの依頼先プロセスを１つのコアへ配置する」という，プロセスのコア配置に制限を加えるものである．以上，２つの方法により，サーバプログラム間通信機構におけるキュー操作を全排他制御から 2 排他制御に変更でき，排他制御オーバヘッドを削減できる．

3. *AnT* での実現方式

3.1 *AnT* オペレーティングシステム

AnT は，マイクロカーネル構造を有するオペレーティングシステムである．以下で，既存の ***AnT***（以降，シングルコア ***AnT***）のプログラム構造，仮想空間の構成，およびサーバプログラム間通信機構について述べる．

プログラム構造を図 1 に示す．OS は，内コア（カーネル）とプロセス (OS サーバ) として動作する外コアからなる．内コアは，最小のシステム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，スケジュール管理やメモリ管理の機能がある．外コアは，適応し

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

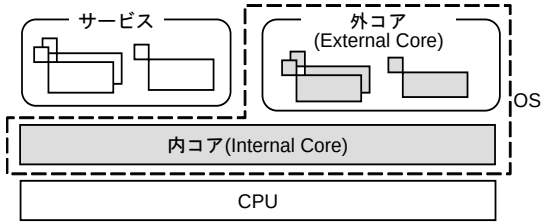


図 1 基本構造

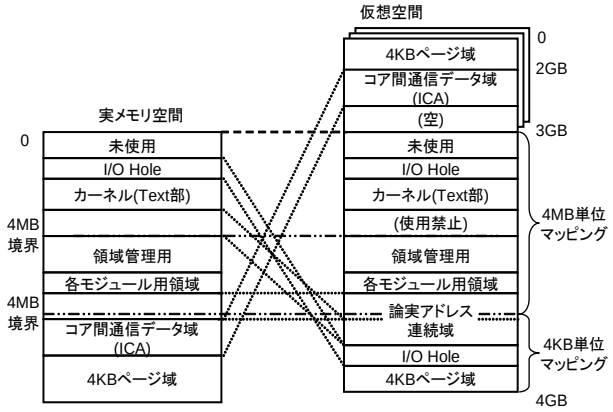


図 2 仮想空間の構成

たシステムに必須なプログラム部分である．例えば，入出力制御機能やファイル管理機能を OS サーバとして提供する．サービスは，サービスを提供するプログラム部分である．

仮想空間の構成を図 2 に示す．仮想空間は，0 から 3GB をプロセス空間，3GB 以降をカーネル空間とし，多重仮想記憶である．0 から 2GB のプロセス空間は，外コアやサービスのプログラムが利用する空間であり，4KB 単位でマッピングする．これに対し，2GB から 3GB のプロセス空間は，内コア，外コア，およびサービスのプログラムが相互のデータ通信

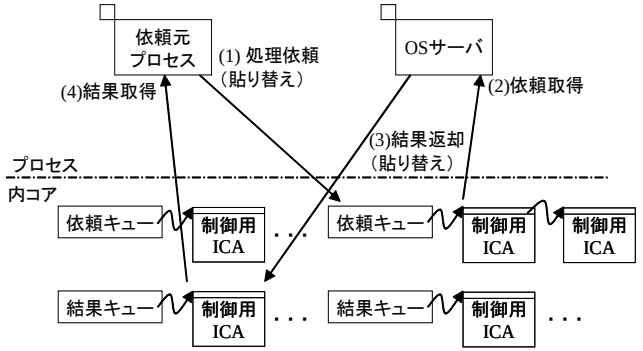


図 3 サーバプログラム間通信機構の処理流れ

に利用する空間である．この領域をコア間通信データ域（ICA：Inter-core Communication Area）と呼ぶ．ICA の特徴として，以下の 3 つがある．

(1) ページ（4 KB）を単位とし，nページ分の領域の確保と解放
(2) 確保した領域（nページ）の実メモリ連続の保証
(3) 2 仮想空間の間での領域の貼り替え

ICA を利用したプロセス間でのデータ授受は，授受するデータを格納した ICA をデータ授受元プロセスの仮想空間から剥がし，データ授受先プロセスの仮想空間へ貼り付けることで行える．つまり，2 つの仮想空間の間で ICA の貼り替えを行うことで，通信する 2 つのプロセス間でのデータ授受を複写レスで実現する．

サーバプログラム間通信機構は，複写レスデータ授受機能を利用した高速な通信機構である．[5] この機構は，制御用情報とデータ情報を各々ICA に格納し，サーバプログラム間でのデータ授受を複写レスで実現している．また，1 つの制御用 ICA を複数のプロセスで持ち回り，依頼情報を積み重ねること (多段依頼) でオーバヘッドを抑制する．さらに，多段依頼に対する結果返却は，逐次的な返却ではなく，結果の返却を必要とする依頼元のプロセスへ直接的に返却し，処理を高速化している．サーバプログラム間通信機構の処理流れを図 3 に示し，以下に説明する．

(1) 依頼元プロセスが処理依頼を行うと，内コアは OS サーバの依頼キューに依頼情報を格納した制御用 ICA を登録し，OS サーバへ制御用 ICA を貼り替える．

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

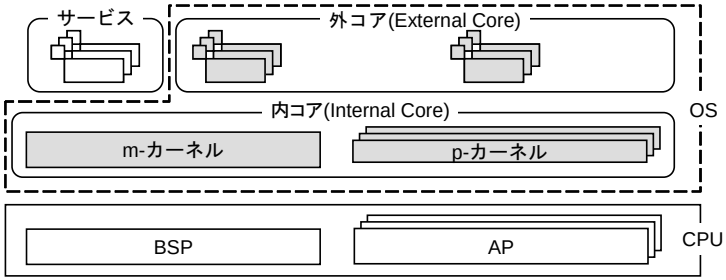


図 4 マルチコア ***AnT*** の基本構造

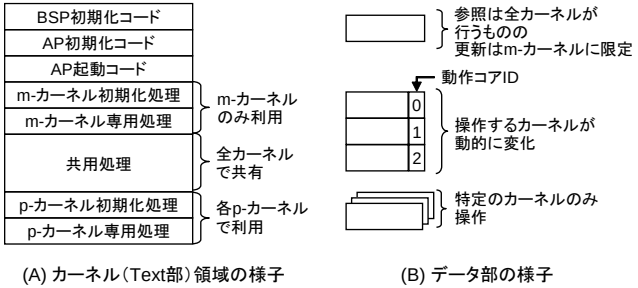


図 5 プログラム構造

（2） OS サーバは，依頼キューから依頼情報を格納した制御用 ICA を取得し処理を実行する．

（3） OS サーバが結果返却を行うと，内コアは依頼元プロセスの結果キューに結果情報を格納した制御用 ICA を登録し，依頼元プロセスへ制御用 ICA を貼り替える．

（4） 依頼元プロセスは，結果キューから結果情報を格納した制御用 ICA を取得し処理を終了する．

3.2 実現方式

3.2.1 基本構造

2 章の方針と設計に基づいて，マルチコアプロセッサ対応版 ***AnT***（以降，マルチコア ***AnT***）を実現する．

基本構造を図 4 に示す．コアの制御法はマルチカーネル方式とし，プロセス実行制御法はマルチスケジューラ方式を採用する．また，カーネルの機能分担として，マスタカーネルを m-カーネル，スレーブカーネルを p-カーネルと名付け，機能を分担する．m-カーネルは，電源投入時に最初に起動するコア（以降，BSP）を制御し，マイクロカーネルに必要な全機能を有する．p-カーネルは，BSP 以外のコア（以降，AP）を制御し，例外・割り込み管理機能，サーバプログラム間通信機能，およびスケジュール機能を有する．

以降では，プログラム構造とサーバプログラム間通信機構について述べる．

3.2.2 プログラム構造

カーネルの手続き部（テキスト部）のメモリ占有量を抑制するため，図 2 のカーネル（Text 部）領域を分割し，カーネル間での共有と個別化を行う．カーネル（Text 部）領域の構成を図 5（A）に示す．カーネル（Text 部）領域は，BSP 初期化コード，AP 初期化コード，AP 起動コード，m-カーネル初期化処理，m-カーネル専用処理，共用処理，p-カーネル初期化処理，および p-カーネル専用処理からなる．BSP 初期化コードは，BSP の初期化に関する処理である．AP 初期化コードと AP 起動コードは，それぞれ AP の初期化と起動に関する処理である．m-カーネル初期化処理は，m-カーネルが利用するデータ部の初期化に関する処理である．m-カーネル専用処理は，m-カーネルのみが有する機能に関する処理であり，例えば，メモリの確保・開放やプロセスの生成・削除の処理である．共用処理は，m-カーネルと p-カーネル双方が有する機能に関する処理であり，例えば，サーバプログラム間通信の処理である．p-カーネル初期化処理は，p-カーネルが利用するデータ部の初期化に関する処理である．p-カーネル専用処理は，p-カーネルのみが有する機能に関する処理であり，例えば，メモリの確保・開放やプロセスの生成・削除の処理を m-カーネルへ依頼する処理である．

これらの領域のうち，m-カーネル初期化処理と m-カーネル専用処理は，m-カーネルのみが利用する．また，p-カーネル初期化処理と p-カーネル専用処理は，各 p-カーネルで共有して利用する．また，共用処理は，全カーネルで共有して利用する．

カーネルのデータ部のメモリ占有量を抑制し，かつデータ部の操作における排他制御オーバヘッドを抑制するため，データ部を 3 つに分割する．この様子を図 5（B）に示す．1 つ目は，全カーネル共通に 1 つ存在し，参照は全カーネルが行うものの更新は m-カーネルに限られるデータ部であり，例えば，メモリ管理表がある．メモリ管理表の初期化は m-カーネル初期化処理で行われ，更新は m-カーネルが行う．また，参照は各カーネルが行う．2

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

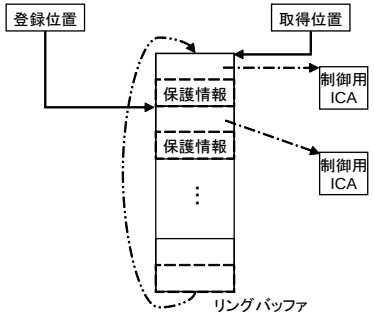


図 6 リングバッファを利用した制御機構

つ目は，管理のために操作するカーネルは動的に変化するものの，同時に複数のカーネルが操作しないデータ部であり，プロセス管理表がある．この部分は，操作するカーネルが動的に変化するため，現在操作する権限を有するカーネルの情報 (これを動作コア ID と名付ける) を持つ．プロセス管理表の初期化は m-カーネル初期化処理で行われ，更新はプロセス管理表に設定された動作コア ID を持つカーネルのみが行う．ここで，動作コア ID の更新の流れを以下に説明する．例として，カーネル間でプロセスの移譲[6] を行う際の動作コア ID の更新の流れを示す．

(1)　移譲元カーネルは，プロセス管理表の動作コア ID に，移譲中を示す値を設定する．
(2)　移譲元カーネルは，プロセスの移譲を移譲先カーネルへ通知する．
(3)　移譲先カーネルは，プロセス管理表の動作コア ID を確認し，動作コア ID に移譲中を示す値が設定されていた場合，自身の動作コア ID を設定する．

３つ目は，特定のカーネルのみが操作するデータ部であり，例えば，例外・割り込み管理表やスケジュールキューがある．これらの初期化は m-カーネル初期化処理または p-カーネル初期化処理で行われ，更新は各カーネルが行う．

3.2.3 サーバプログラム間通信機構

OS サーバ間の通信相手の制限により，サーバプログラム間通信機構におけるキュー操作を２排他制御にする．さらに，リングバッファを用いた制御機構により，排他制御を行わない形で実現する．この様子を図 6 に示す．リングバッファの各エントリには，制御用 ICA のアドレスと保護情報 (読み書き可または読み込み専用) を登録する．また，登録位置と取得位置を設け，依頼元プロセスが動作するコアと OS サーバが動作するコアがリングバッファ内の異なるエントリを操作するようにする．登録位置は，次に依頼または結果を登録する位置であり，処理依頼または結果返却を行うプロセスが走行するコアで更新する．取得位置は，次に依頼または結果を取得する位置であり，処理依頼または結果返却を受け取るプロセスが走行するコアで更新する．

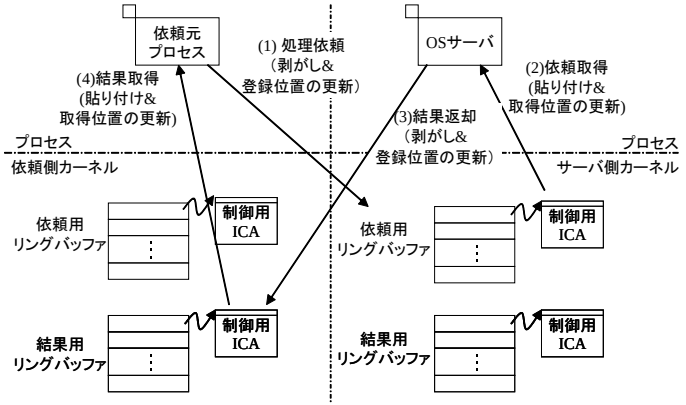


図 7 マルチコア ***AnT*** におけるサーバプログラム間通信機構の処理流れ

また，ICA の授受方式を改善した．シングルコア ***AnT*** では，ICA 操作処理を局所化するため，OS サーバへの ICA の貼り替え（剥がし&貼り付け）を依頼元プロセスが動作するコアで行っている．しかし，マルチコア ***AnT*** では，OS サーバと依頼元プロセスの走行コアが異なることがあるため，この処理を行えない．そこで，サーバプログラム間通信時における ICA の貼り替えは，操作対象の仮想空間を有するプロセスが行うこととした．具体的には，依頼元プロセスからの ICA の剥がしは，依頼元プロセスの動作するコアで行い，OS サーバへの ICA の貼り付けは，OS サーバの動作するコアで行う．

以上に基づくサーバプログラム間通信機構の処理流れを図 7 に示し，以下に説明する．

(1)　依頼元プロセスが処理依頼を行うと，依頼元プロセスが動作するコア上のカーネル（以降，依頼側カーネル）は，OS サーバの依頼用リングバッファの登録位置が示すエントリに制御用 ICA のアドレスと保護情報を登録し，登録位置を更新する．その後，依頼元プロセスから制御用 ICA を剥がす．

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

表 1 手続き部のコード量

OS		行数	
シングルコア ***AnT***		19,437	
マルチコア ***AnT***	BSP 初期化コード	173	21,566
	AP 初期化コード	91	
	AP 起動コード	22	
	m-カーネル初期化処理	1,819	
	m-カーネル専用処理	5,991	
	共用処理	7,882	
	p-カーネル初期化処理	802	
	p-カーネル専用処理	263	
	その他	4,523	

(2) OS サーバが動作するコア上のカーネル（以降，サーバ側カーネル）は，依頼用リングバッファの取得位置が示すエントリに登録された制御用 ICA のアドレスと保護情報をもとに，OS サーバに制御用 ICA を貼り付け，取得位置を更新する．

(3) OS サーバが結果返却を行うと，サーバ側カーネルは，依頼元プロセスの結果用リングバッファの登録位置が示すエントリに制御用 ICA のアドレスと保護情報を登録し，登録位置を更新する．その後，OS サーバから制御用 ICA を剥がす．

(4) 依頼側カーネルは，結果用リングバッファの取得位置が示すエントリに登録された制御用 ICA のアドレスと保護情報をもとに，依頼元プロセスに制御用 ICA を貼り付け，取得位置を更新する．

4. 評　　価

4.1 メモリ占有量

マルチコア ***AnT*** のメモリ占有量を評価するため，カーネルの手続き部のコード量とデータ部のサイズを評価する．

手続き部のコード量を表 1 に示す．表 1 より，以下のことがわかる．

(1) 手続き部のコード量の差は，2,129 行 (21,566 行 −19,437 行) である．これは，***AnT*** のマルチコアプロセッサ対応に伴う以下の 5 つの処理の追加によるものである．

（A） リングバッファの制御処理
（B） p-カーネルで利用する管理表の初期化処理
（C） p-カーネルから m-カーネルへの処理依頼のための処理
（D） AP の起動・初期化処理

表 2 データ部のサイズ

OS		サイズ（Byte）		備考
シングルコア ***AnT***		1,234,836		
マルチコア ***AnT***	参照は全カーネルが行うものの更新は m-カーネルに限定	127,012	1,366,724	
	操作するカーネルが動的に変化	40,448		プロセス数 128 の場合
	特定のカーネルのみ操作	1,199,264		条件により増減あり

（E） 共用処理内での動作コア ID の確認処理

(2) 共用処理は，7,882 行である．m-カーネルと p-カーネルで処理を共有しない場合，この処理を m-カーネルと p-カーネルでそれぞれ用意する必要がある．このとき，手続き部の全コード量は 29,448 行 (173 行 +91 行 +22 行 +1,819 行 +5,991 行 +(7,882 行 ∗2) + 802 行 +263 行 +4,523 行) となる．つまり，m-カーネルと p-カーネルで処理を共有することにより，約 27%コード量を削減できる．

(3) p-カーネル初期化処理と p-カーネル専用処理は，それぞれ 802 行と 263 行である．もし，これらの処理を共有しない場合，p-カーネル数分用意する必要がある．例えば，動作コア数が 4 の場合，この部分のコード量は 3,195 行 ((802 行 +263 行) ∗ (4 − 1)) であり，手続き部の全コード量は 23,696 行 (173 行 +91 行 +22 行 +1,819 行 +5,991 行 +7,882 行 +(802 行 +263 行) ∗ (4 − 1) + 4,523 行) となる．つまり，動作コア数が 4 の場合，この部分の共有により，約 9%コード量を削減できる．

動作コア数が 4 の場合のデータ部のサイズを表 2 に示す．表 2 より，以下のことがわかる．

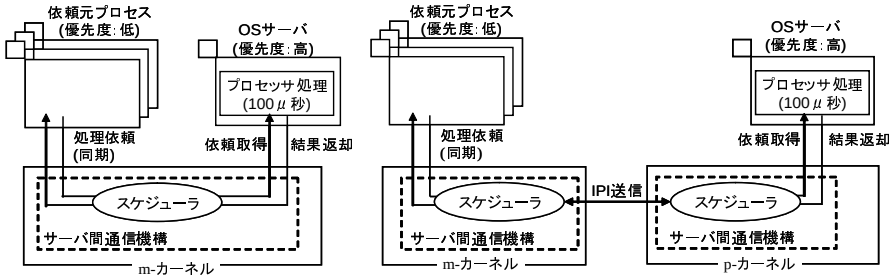
(1) データ部のサイズの差は，131,888Byte(1,366,724Byte−1,234,836Byte) である．これは，***AnT*** のマルチコアプロセッサ対応に伴う以下の 3 つのデータ部の追加によるものである．

（A） 各カーネルの動作コア ID の管理や現在動作中のコア数を管理する管理表
（B） サーバプログラム間通信時に利用するリングバッファ
（C） p-カーネルが利用する各種管理表

(2) 参照は全カーネルが行うものの更新は m-カーネルに限られるデータ部のサイズは 127,012Byte である．この部分は m-カーネルのみが利用するため，動作コア数に関わらずサイズは一定である．

(3) 操作するカーネルが動的に変化するデータ部のサイズは 40,448Byte である．この部分はプロセス管理表であり，プロセス数に比例して増加する．例えば，プロセス数が 128 の

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1



(A) 1コア動作の場合　　(B) 2コア動作の場合

図 8 測定環境

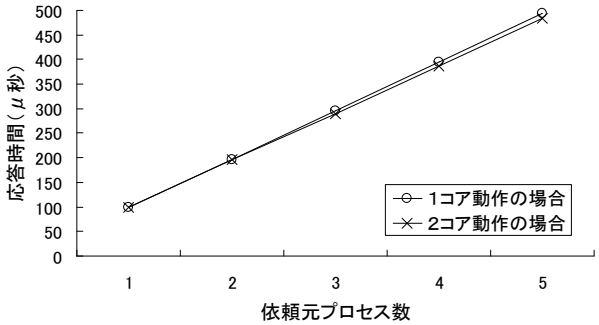


図 9 OS サーバの応答時間

場合，プロセス管理表のサイズは 316Byte であるため，40,448Byte(316Byte∗128) である．

(4)　特定のカーネルのみが操作するデータ部のサイズは 1,199,264Byte である．このうち，1,048,576Byte は，各プロセスに貼り付けられた ICA の管理のための管理表であり，ICA を利用するプロセス数と実メモリ空間上の ICA の総数に比例して増加する．例えば，ICA を利用するプロセス数が 32，ICA の総数が 4,096 ページの場合，管理表のサイズは 8Byte であるため，1,048,576Byte(8Byte∗(4,096∗32)) である．また，73,728Byte は，各プロセスに割り当てられるリングバッファであり，プロセス数に比例して増加する．例えば，プロセス数が 128 の場合，リングバッファのサイズは 288 バイトであるため，73,728Byte(288Byte∗(128∗2)) である．残りの 76,960Byte は，各カーネルで利用する各種管理表であり，動作コア数に比例して増加する．例えば，動作コア数が 4 の場合，管理表の合計サイズは 19,240Byte であるため，76,960Byte(19,240Byte∗4) である．

4.2 サーバプログラム間通信機構

4.2.1 応 答 時 間

応答時間の測定環境を図 8 に示す．OS サーバは，処理依頼を受け取ると，プロセッサ処理（特定の領域のインクリメントを繰り返す処理）を 100 μ秒行った後，結果返却を行う．応答時間は，依頼元プロセスが OS サーバへ同期の処理依頼を行った後，結果返却を受け取るまでの時間である．この処理を 10 回行い，平均処理時間を算出した．なお，OS サーバの優先度は依頼元プロセスより高く，依頼元プロセスの優先度は全て同じである．

測定結果を図 9 に示す．図 9 より，以下のことがわかる．

(1)　応答時間は，1 コア動作の場合と 2 コア動作の場合で同様である．また，依頼元プロセス数の影響を受けない．これにより，サーバプログラム間通信のオーバヘッドは小さいといえる．

(2)　依頼元プロセス数の増加に伴い，2 コア動作の場合の応答時間は，1 コア動作の場合に比べ短くなる．これは，次の理由による．1 コア動作の場合，依頼元プロセス数に関係なく，依頼元プロセスと OS サーバが交互に走行し，プロセス切り替えが多発する．一方，2 コア動作の場合，依頼元プロセス数が増加すると，OS サーバは次々と依頼を処理するために WAIT 状態に移行せず RUN 状態で処理を継続して実行する．つまり，プロセス切り替えの数が減少するためである．

(3)　依頼元プロセス数が 1 のとき，2 コア動作の場合の応答時間は，1 コア動作の場合に比べ約 1 μ秒長い．これは，2 コア動作の場合，OS サーバの起床は，プロセス切り替えだけでなく，p-カーネルへの IPI（Inter-Processor Interrupt）送信を伴うためである．

4.2.2 スループット

4.2.1 項と同じ環境で，OS サーバのスループットを測定した．測定結果を図 10 に示す．図 10 より，以下のことがわかる．

(1)　1 コア動作の場合，依頼元プロセス数に関係なくスループットは一定になる．これは，1 コア動作の場合，依頼元プロセス数に関係なく，依頼元プロセスと OS サーバが交互に走行し，OS サーバへの依頼数が常に 1 になるためである．

Vol.2012-DPS-150 No.37
Vol.2012-CSEC-56 No.37
2012/3/1

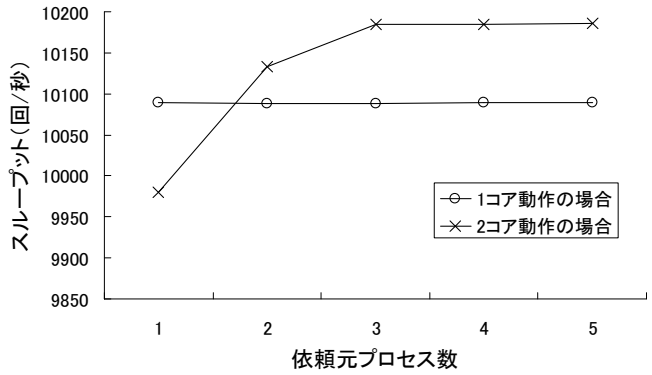


図 10 スループット

(2) プロセス数が 1 のとき，2 コア動作の場合のスループットは，1 コア動作の場合に比べ低い．これは，2 コア動作の場合，OS サーバの起床は，プロセス切り替えだけでなく，p-カーネルへの IPI 送信を伴うためである．

(3) 依頼元プロセス数の増加に伴い，2 コア動作の場合のスループットは高くなる．これは，2 コア動作の場合，依頼元プロセス数が増加すると，依頼元プロセスと OS サーバが並列動作するためである．つまり，負荷分散が実現できている．

(4) プロセス数が 3 以上のとき，2 コア動作の場合のスループットは一定になる．これは，依頼元プロセスによる処理依頼の発行間隔が OS サーバの処理時間よりも短くなり，OS サーバの処理時間がボトルネックとなるためである．

5. おわりに

マルチコアプロセッサ向けマイクロカーネル構造 OS の方針について述べ，設計としてコアの制御法，プロセスの実行制御法，カーネルの機能分担とプログラム構造，およびサーバプログラム間通信機構について述べた．また，***AnT*** における実現方式として，マルチコア ***AnT*** の基本構造，プログラム構造，およびサーバプログラム間通信機構について述べた．基本構造として，BSP を制御し全機能を有する m-カーネルと各 AP を制御し例外・割り込み管理機能，スケジュール機能，およびサーバプログラム間通信機能を有する p-カーネルで構成することを述べた．また，OS のメモリ占有量を抑制し，コア間の排他制御オーバヘッドを抑制するカーネルの手続き部とデータ部のプログラム構造について述べた．さらに，排他制御を行わない形で実現したサーバプログラム間通信機構について述べた．

評価として，マルチコア ***AnT*** のメモリ占有量とサーバプログラム間通信性能について報告した．メモリ占有量の評価では，m-カーネルと p-カーネルでカーネルの手続き部を共有することにより，コード量を約 27%削減できることを示した．また，各 p-カーネルでカーネルの手続き部を共有することにより，動作コア数が 4 の場合，コード量を約 9%削減できることを示した．さらに，カーネルのデータ部を 3 つに分割することにより，データ部の増加を抑制できることを示した．さらに，サーバプログラム間通信性能の評価では，依頼元プロセスと OS サーバを 1 コア上で動作させた場合と 2 コア上で動作させた場合の応答時間が同様であり，依頼元プロセス数を増加させた場合，2 コア上で動作させた場合の応答時間が 1 コア上で動作させた場合よりも短くなることを示した．さらに，2 コア上で動作させた場合，依頼元プロセス数の増加に伴い，OS サーバのスループットが高くなることを示し，負荷分散の有効性を明らかにした．

残された課題として，マルチコア ***AnT*** におけるサービス評価がある．

参 考 文 献

1) J. Liedtke, “Toward Real Microkernels,” Commnuications of The ACM, Vol.39, Issue 9, pp.70-77, 1996.
2) A.S. Tanenbaum, J.N. Herder, and H. Bos, “Can we make operating systems reliable and secure?,” IEEE Computer Magazine, Vol.39, No.5, pp.44-51, 2006.
3) D.L.Black, D.B.Golub, D.P.Julin, R.F.Rashid, R.P.Draves, R.W.Dean, A.Forin, J.Barrera, H.Tokuda, G.Malan, and D.Bohman, “ Microkernel Operating System Architecture and Mach, ”Journal of Information Processing, Vol.14, No.4, pp.442-453, 1992.
4) 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もつ ***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.
5) 岡本幸大, 谷口秀夫, “ ***AnT*** オペレーティングシステムにおける高速なサーバプログラム間通信機構の実現と評価, ” 電子情報通信学会論文誌, Vol.J93-D, No.10, pp.1977-1989, 2010.
6) 佐古田 健志，井上 喜弘，谷口 秀夫，“ マルチコア ***AnT*** におけるプロセス移譲機能, ” 電子情報通信学会 2012 年総合大会，2012. （掲載予定）