Vol.2011-OS-118 No.12
2011/7/27

SH-4上で動作する *AnT* オペレーティングシステムのサーバプログラム間通信機構の評価

鶴谷昌弘[†1]　山内利宏[†1]　谷口秀夫[†1]

マイクロカーネル構造の ***AnT*** オペレーティングシステムは，高速なサーバプログラム間通信機構を持ち，Pentium4 上で性能評価が行われている．また，SH-4 に移植されている．ここでは，SH-4 上でサーバプログラム間通信機構の基本処理の性能を測定し，Pentium4 上での測定結果と比較評価した結果を述べる．また，評価結果を基に，SH-4 の特徴を生かしたサーバプログラム間通信の高速化手法を述べる．

Evaluation of Inter Server Program Communication for *AnT* operating system on SH-4

Masahiro Tsuruya,[†1] Toshihiro Yamauchi[†1]
and Hideo Taniguchi [†1]

AnT is an operating system based on microkernel architecture. ***AnT*** has a fast inter-server program communication mechanism, and has been evaluated performance of the mechanism on Pentium4. On the other hand, ***AnT*** has been ported to SH-4. In this paper, we evaluate basic performance of the inter-server program communication for ***AnT*** executed on SH-4 by comparing with ***AnT*** executed on Pentium4. In addition, this paper describes speedup techniques utilizing features of SH-4 of the communication between server programs.

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

1. はじめに

高い適応性と堅牢性を実現する OS のプログラム構造として，マイクロカーネル構造がある．マイクロカーネル構造は，割込処理や例外処理といった最小限の OS 機能をカーネルとして実現し，ファイル管理やデバイスドライバなどの OS 機能をプロセスとして実現するプログラム構造である．つまり，多くの OS 機能は，カーネル外にプロセス（以降，OS サーバと呼ぶ）として実現する．これにより，プログラムの追加や削除を容易にできる．また，機能を OS サーバごとに分散させることにより，プログラムの暴走によるシステム全体の破壊を防止できる．このため，マイクロカーネル構造の研究が行われている[1)2)3)]．しかし，マイクロカーネル構造では，OS サーバ間の通信が頻発するため，Linux のようなモノリシックカーネル構造の OS と比べ，性能が低下する．そこで，この性能低下を抑制する機構が必要となる．

我々は，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)5)]（An operating system with adaptability and toughness）（以降，***AnT*** と略す）を開発している．***AnT*** は高速なサーバプログラム間通信機構[6)] を持ち，Pentium4 上で性能評価が行われている．また，SH-4 に移植されている[7)]．

ここでは，***AnT*** のサーバプログラム間通信機構について SH-4 上で性能を測定し，Pentium4 上での測定結果と比較評価した結果を述べる．また，評価結果を基に，SH-4 の特徴を生かしたサーバプログラム間通信の高速化手法を述べる．

2. *AnT* オペレーティングシステム[6)]

2.1 プログラム構造

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．***AnT*** のプログラムは，OS とサービスからなる．この様子を**図 1** に示す．OS は，カーネル（内コア）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．外コアは，システムの利用形態に適応するために必須なプログラム部分であり，動的に再構成可能な構造を有する．サービスは，サービスを提供するプログラム部分である．

2.2 複写レスデータ授受

プロセス間の通信を高速化するため，コア間通信データ域 (ICA: Inter-core Communication Area) を利用した複写レスでのデータ授受機能がある．ICA の特徴として以下の 3

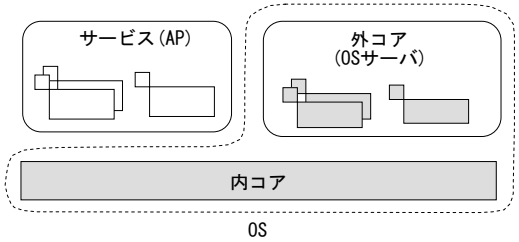

図 1 ***AnT*** の基本構造

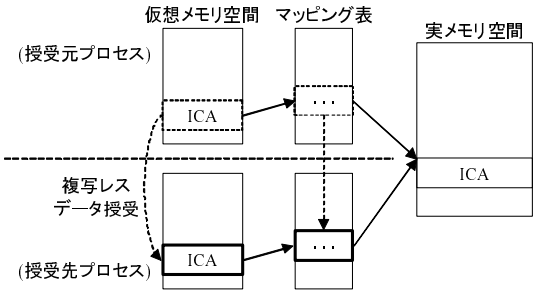

図 2 複写レスデータ授受の様子

つがある．

(1) ページ (4KB) 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 仮想空間での領域の貼り替え

ICA は，内コアによりページを最小単位として管理される領域であり，ICA へのアクセスは，プロセスごとの仮想空間のマッピング表を通して行われる．ここで，マッピング表への書き込みを貼り付けと呼び，マッピング表からの削除を剥がしと呼ぶ．プロセス間の複写レスでのデータ授受の様子を**図 2** に示す．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われる．これらの操作をまとめて ICA の貼り替えと呼ぶ．

ICA に関するインタフェースを**表 1** に示す．createica() と deleteica() は AP と外コア及び内コアのモジュールに提供されており，attachica() と detachica() は内コアのモジュールのみに提供されている．また，ICA の確保時に 4KB 以上の size を指定することにより，複数ページを確保することが可能である．複数ページ領域で確保された ICA の貼り替えや削除は，複数ページまとめて行われる．なお，プロセスで確保されたまま解放されない ICA は，プロセス終了時に内コアでまとめて解放される．

表 1 ICA に関するインタフェースの機能と形式

機能	形式
ICA の確保	createica(size, flag); size: 確保する領域の大きさ flag: 確保する際に使用するフラグ
ICA の解放	deleteica(vaddr); vaddr: 解放する領域の先頭論理アドレス
ICA の貼り付け	attachica(vmid, vaddr, prot); vmid: 貼り付け先の仮想空間識別子 vaddr: 貼り付ける領域の先頭論理アドレス prot: 貼り付ける領域の保護情報
ICA の剥がし	detachica(vmid, vaddr); vmid: 剥がし先の仮想空間識別子 vaddr: 剥がす領域の先頭論理アドレス

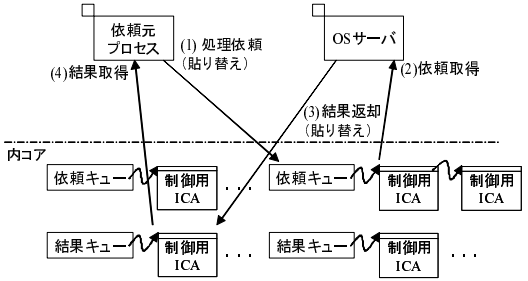

図 3 サーバプログラム間通信の基本機構

2.3 サーバプログラム間通信機構

2.3.1 基 本 機 構

サーバプログラム間通信の基本機構を**図 3** に示す．ICA を利用することにより，プロセス間の複写レスデータ授受を実現している．具体的には，OS サーバへ渡す引数や通信制御の情報（以降，依頼情報）を制御用の ICA（以降，制御用 ICA）に格納し，扱うデータをデータ用の ICA（以降，データ用 ICA）に格納する．内コアは，各プロセスごとに通信の

表 2 サーバプログラム間通信インタフェースの機能と形式

機能	形式
サーバ登録	registserver(coreid); coreid: 登録する OS サーバのコア ID
同期処理依頼	callsync(p); p: 依頼情報を格納した制御用 ICA のアドレス
非同期処理依頼	callasync(p); p: 依頼情報を格納した制御用 ICA のアドレス
結果返却	ret(p); p: 返却情報を格納した制御用 ICA のアドレス
依頼取得 & 結果返却	get(p); p: 返却情報を格納した制御用 ICA のアドレス または NULL

ための依頼キューと結果キューを持つ．処理依頼は，依頼キューへ依頼情報を格納した制御用 ICA を登録し，これを OS サーバへ貼り替えることで行う．また，結果返却は，結果キューへ処理結果（以降，結果情報）を格納した制御用 ICA を登録し，これを依頼元プロセスへ貼り替えることで行う．基本的な通信の流れを以下に述べる．

(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．

(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．

2.3.2　通信インタフェース

同期型と非同期型の通信インタフェースを同様な形式で提供し，両インタフェースを選択して利用可能である．例えば，AP プロセスから OS サーバへの処理依頼はシステムコールに相当するため，同期型の通信である．また，OS サーバは，多数の処理依頼を同時に受け付けるため，OS サーバから OS サーバへの処理依頼は非同期型の通信である．

サーバプログラム間通信に関し，内コアがプロセスへ提供するインタフェースの機能と形式を**表 2** に示し，以降で説明する．

registserver() は，プログラム間通信の際に必要な情報を OS サーバ自身が内コアに登録する操作である．OS サーバは registserver() の引数として自 OS サーバのコア ID を指定する．これにより，コア ID と OS サーバのプロセス ID が対応づけられて内コアへ登録される．他プロセスはコア ID を利用することで OS サーバへの処理依頼が可能となる．

callsync() は，OS サーバに対し同期的に処理依頼を発行する操作である．依頼元のプロセスは依頼情報を制御用 ICA に格納し，格納した制御用 ICA を引数として callsync() を発行する．これにより，OS サーバの依頼キューへ依頼が登録される．その後，依頼元のプロセスは OS サーバでの処理が終了するまで待ち状態へ移行する．

callasync() は，OS サーバに対し非同期的に処理依頼を発行する操作である．処理は callsync() と同様であるが，依頼元のプロセスへは OS サーバでの処理終了を待たずに制御が戻る．

ret() は結果返却を行う操作である．OS サーバは依頼された処理を終了すると，結果情報を格納した制御用 ICA を引数に指定し，ret() を発行する．これにより，依頼元のプロセスへ結果が返却される．なお，引数に指定する制御用 ICA は必ず処理依頼により渡されたものと同じでなければならない．これは，処理依頼により渡された制御用 ICA には依頼元のプロセス情報が格納されており，これを用いて結果を返却するためである．

get() は，結果返却，及び依頼または結果を取得する操作である．OS サーバは，引数に NULL を指定して get() を発行することで，依頼または結果を取得する．依頼または結果が登録されていない場合，登録されるまで待ち状態へ移行する．引数に返却情報を格納した制御用 ICA を指定した場合，結果返却を行った後，依頼または結果の取得を行う．つまり，get(p); は ret(p);get(NULL); と等価である．なお，依頼と結果の両方が登録されている場合，結果の取得を優先する．

2.3.3　多段依頼と直接返却

マイクロカーネル OS では，OS サーバを多段に経由して処理依頼が発行される．このとき，制御用 ICA を処理依頼のたびに確保すると処理オーバヘッドが大きい．そこで，1 つの制御用 ICA を持ち回り，依頼情報を積み重ねることでオーバヘッドを抑制する．また，積み重ねられた情報を基に依頼元プロセスへ結果を渡すことで，結果返却を実現する．

基本的に，多段依頼された処理は，積み重ねられた依頼情報を基に中継した OS サーバを経由した逐次的な返却が行われる．しかし，必ずしも逐次的な返却を行う必要はないため，依頼元のプロセスへの直接的な返却を実現することで，処理を高速化する．具体的には，制御用 ICA に flag を設け，返却の要否を設定できることとした．結果返却時に内コアが flag を確認し，返却が必要な依頼元のプロセスに直接返却を行う．

表 3 測定環境

	SH-4_***AnT***	Pentium4_***AnT***
OS	***AnT***	***AnT***
CPU	SH-4 240MHz (SH7751R)	Intel(R) Pentium4 2.8GHz
パイプライン段数	5 段	20 段
メインメモリ	32MB	256MB
命令キャッシュ	16KB	-
オペランドキャッシュ	32KB	-
L1 キャッシュ	-	16KB
L2 キャッシュ	-	256KB

3. 評　　価

3.1 評価項目と測定環境

サーバプログラム間通信機構について，基本性能と直接返却を利用した場合を評価する．基本性能については，基本処理の部分ごとの処理（部分処理）の処理時間を測定し，評価する．また，基本処理の処理時間を測定し，部分処理の処理時間を用いて分析を行う．***AnT*** のサーバプログラム間通信機構の特徴の 1 つである直接返却については，直接返却による通信オーバヘッドの削減効果について評価する．なお，以降では，SH-4 上で動作する ***AnT*** を SH-4_***AnT***，Pentium4 上で動作する ***AnT*** を Pentium4_***AnT*** と略す．

測定環境を**表 3** に示す．なお，測定には SH-4 に搭載されているタイマカウンタを用いた．これは，SH-4 は Pentium4 の RDTSC 命令に相当する命令を備えていないためである．また，推定値の算出は各プロセッサの動作周波数の比を用いて行う．表 3 より，Pentium4 の動作周波数は 2.8GHz であり，SH-4 の動作周波数 240MHz の約 11.7 倍である．したがって，Pentium4_***AnT*** の処理時間を 11.7 倍したものを SH-4_***AnT*** の処理時間の推定値とする．

3.2 基 本 性 能

サーバプログラム間通信制御の基本処理として，同期処理依頼とその結果返却，及び非同期処理依頼とその結果返却がある．また，それぞれの処理においてデータ用 ICA の授受を行う場合と行わない場合が存在する．これらについて，SH-4 上での性能を明らかにする．この際，Pentium4 上での性能[6] を参考に比較評価を行う．

図 4 に示すように，依頼元プロセスと OS サーバ間で通信を行い，各基本処理の処理時間を測定する．なお，各基本処理は，以下の部分処理からなる．

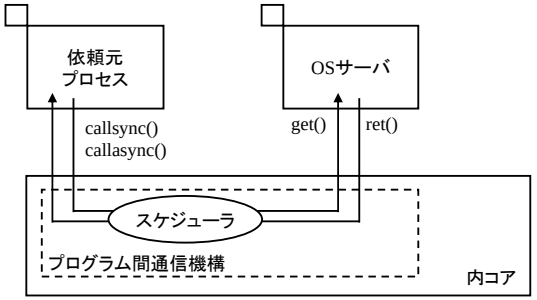

図 4 基本処理の測定の様子

表 4 基本処理において部分処理が呼び出される回数

			ICA 貼り付け	ICA 剥がし	システムコール	ICA への初アクセス	プロセス切換
処理依頼	同期	データ用 ICA 無	1 回	0 回	1 回	1 回	1 回
		データ用 ICA 有	2 回	1 回	1 回	1 回	1 回
	非同期	データ用 ICA 無	1 回	1 回	2 回	1 回	1 回
		データ用 ICA 有	2 回	2 回	2 回	1 回	1 回
結果返却	同期	データ用 ICA 無	0 回	1 回	1 回	0 回	1 回
		データ用 ICA 有	1 回	2 回	1 回	0 回	1 回
	非同期	データ用 ICA 無	1 回	1 回	1 回	1 回	1 回
		データ用 ICA 有	2 回	2 回	1 回	1 回	1 回

(1) ICA 貼り付け
(2) ICA 剥がし
(3) システムコール発行
(4) ICA への初アクセス
(5) プロセス切換

また，各基本処理において，部分処理が呼び出される回数を**表 4** に示す[6]．ここで，(3) システムコール発行は，プロセスから内コアへの処理移行と内コアからプロセスへの処理の戻りを合わせた処理である．また，(4)ICA への初アクセスは，そのプロセスに ICA が貼り付けられた後，初めてアクセスした際に発生する処理である．これは，TLB ミス等のキャッシュミスによるものである．

各部分処理の処理時間を**表 5** に示す．表 5 より，以下のことがわかる．

(1) ICA 貼り付けと ICA 剥がしの処理時間は，他の部分処理の処理時間と比較し，推定

表 5 部分処理の処理時間

処理内容	SH-4_***AnT*** の処理時間	Pentium4_***AnT*** の処理時間
ICA 貼り付け	1.5（1.6）μ秒	0.14 μ秒
ICA 剥がし	4.8（5.8）μ秒	0.50 μ秒
システムコール発行	1.2（6.3）μ秒	0.54 μ秒
ICA への初アクセス	10.3（15.9）μ秒	1.36 μ秒
プロセス切換	8.3（14.1）μ秒	1.21 μ秒

注：() 内は Pentium4_***AnT*** の測定結果からの推定値

値に近い値である．これは，ICA 貼り付けと ICA 剥がしは PU 処理の割合が大きく，プロセッサの動作周波数の影響を大きく受けるためである．

(2)　システムコール発行の処理時間は，推定値の 20 %以下である．これは，SH-4 は Pentium4 と比較し，パイプライン段数が少ないためであると推察する．

(3)　TLB ミスの際，Pentium4_***AnT*** はハードウェアがマッピング表を探索し，アドレス変換情報を TLB へ登録する．一方，SH-4_***AnT*** はソフトウェアによりマッピング表を探索し，アドレス変換情報を TLB へ登録する．このため，ICA への初アクセスの処理時間は，推定値と異なる．

(4)　プロセス切換処理は，ICA 貼り付けと ICA 剥がしと同様に，PU 処理の割合が大きい．しかし，処理時間は推定値の 60 %以下である．これは，SH-4_***AnT*** と Pentium4_***AnT*** では，プロセス切換に含まれる仮想空間切換処理が異なるためである．Pentium4_***AnT*** では，仮想空間を切換える際，TLB をフラッシュする．一方，SH-4_***AnT*** では，TLB は仮想空間識別子を有するため，TLB をフラッシュしない．このため，TLB フラッシュの有無による TLB ミスのオーバヘッドの差であると推察する．

次に各基本処理の処理時間について述べる．測定において，処理依頼は，OS サーバへ渡す引数及び戻り値を無しとし，データ用 ICA のサイズを 4KB にした．また，通信機構のオーバヘッドを明確化するため，OS サーバでは通信に関連する処理以外の固有な処理は行わない．

表 4 と表 5 を用いて分析を行った結果も含めた内容を**図 5** に示す．図 5 より，以下のことがわかる．

(1)　処理依頼と結果返却に関して，それぞれの差分は，同期処理で約 7 μ秒，非同期処理で約 1 μ秒である．この差分は，同期処理では，制御用 ICA の貼り付けと剥がしの差，及び制御用 ICA への初アクセスによるものであり，非同期処理ではシステムコール発行回数の違いによるものである．

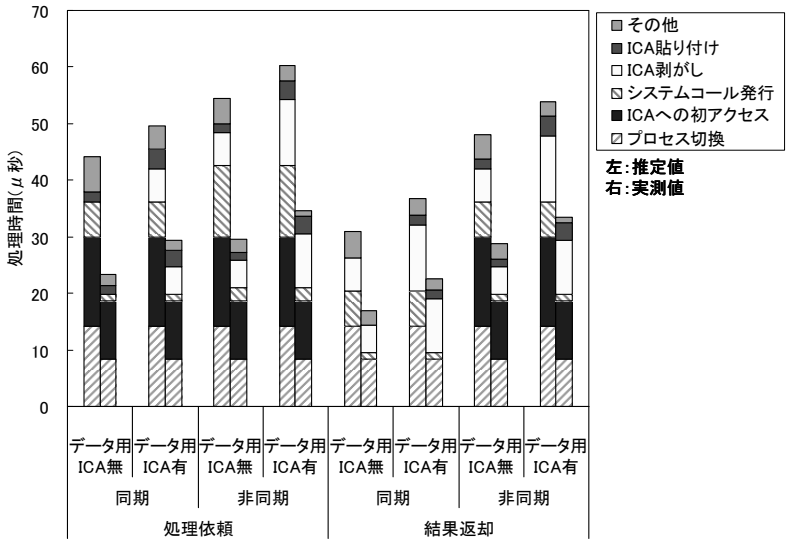

図 5 基本処理の処理時間

(2)　同期処理と非同期処理に関して，それぞれの差分は，処理依頼で約 6 μ秒，結果返却では約 10 μ秒である．この差分は，処理依頼では制御用 ICA 剥がし，及びシステムコール発行回数の違いによるものであり，結果返却では制御用 ICA への初アクセスによるものである．

(3)　データ用 ICA の有無に関して，それぞれの差分は約 6 μ秒である．この差分は，データ用 ICA の貼り替え（ICA の貼り付けと剥がし）によるものである．処理依頼と結果返却において，データ用 ICA はアクセスされないため，データ用 ICA への初アクセスによるオーバヘッドは発生しない．

(4)　ICA への初アクセスとプロセス切換の処理時間は，他の部分処理の処理時間と比較し大きい．また，ICA への初アクセスは同期の結果返却を除く全ての基本処理に含まれ，プロセス切換は全ての基本処理に含まれる．このため，ICA への初アクセスとプロセス切換処理のオーバヘッドを削減することで，サーバプログラム間通信を高速化することができる．

3.3 直接返却

直接返却を利用した場合の処理時間を評価する．OS サーバを n 段介した処理依頼において，AP プロセスが処理依頼を発行してから結果が返却されるまでの処理時間を測定し，逐

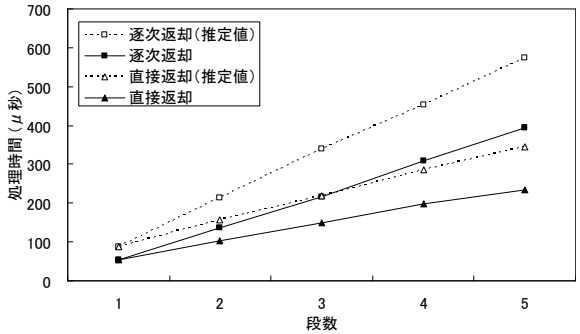

図 6 直接返却と逐次返却の比較

次返却を利用した場合と直接返却を利用した場合の処理時間を比較する．

測定結果を**図 6** に示す．図 6 より，逐次返却と直接返却では，ともに段数増加に比例して処理時間が増加していることがわかる．これは，段数増加に比例し，逐次返却の場合は，処理依頼と結果返却の回数が増加し，直接返却の場合は処理依頼の回数が増加するためである．ここで，各処理時間が推定値より小さいのは，3.2 節で述べた理由による．また，段数増加による処理時間の増加の割合が直接返却では逐次返却に比べ約 50 %に抑えられていることがわかる．このことから，直接返却を利用することによる処理効率化の有効性は高い．

4. SH-4 での高速化手法

4.1 高速化の考え方

SH-4_***AnT*** におけるサーバプログラム間通信機構の性能は，Pentium4_***AnT*** の性能から推定される性能よりは良い．しかし，十分高性能であるとはいえない．SH-4_***AnT*** は，Pentium4_***AnT*** を SH-4 に移植したものであるため，サーバプログラム間通信機構は Pentium4 などの IA-32 アーキテクチャのプロセッサに最適されており，SH-4 の特徴は生かされていない．そこで，SH-4_***AnT*** のサーバプログラム間通信機構を SH-4 用に最適化する必要がある．

3.2 節より，SH-4_***AnT*** における通信オーバヘッドは，主に ICA 操作（ICA の貼り付けと剥がし），TLB ミス，及びプロセス切換からなることがわかる．このうち，プロセス切換は，切換前プロセスのコンテキストの保存，仮想空間切換，及び切換先プロセスのコンテキストの復元の処理からなる．また，各処理はプロセス切換処理において必要不可欠であり，仮想空間切換の処理はすでに SH-4 に適した処理となっている．このため，プロセス切換のオーバヘッドの削減は難しい．

一方，ICA 操作と TLB ミスのオーバヘッドの削減は可能であると考えられる．これは，ICA の管理方法が Pentium4 の MMU を考慮したものとなっており，SH-4 に適していないためである．また，ICA の管理方法を改良することにより，ICA 操作のオーバヘッドを削減できると考えられる．さらに，TLB ミスのオーバヘッドも削減可能であると考えられる．

4.2 SH-4 の MMU の特徴

SH-4 の MMU の特徴を生かし，SH-4_***AnT*** におけるサーバプログラム間通信を高速化する．SH-4 の MMU の特徴を以下に示す．

(1) ひとつの仮想空間内において，ページサイズを 1KB，4KB，64KB，及び 1MB から選択でき，各ページサイズを混同して利用できる．

(2) TLB ミス発生時は，例外が発生し，ソフトウェアへ処理が移行する．このため，TLB へのアドレス変換情報の登録をソフトウェアで自由に設定できる．

(3) ソフトウェアで事前に TLB へのアドレス変換情報の登録を行うことにより，TLB ミスの発生を防ぐことができる．

(4) SH-4 の TLB は 64 エントリで構成され，TLB へアドレス変換情報を登録する際，どのエントリを利用するか指定できる．指定しない場合，SH-4 に搭載されているランダムカウンタの値により決定される．

(5) 多重仮想記憶を利用する場合，TLB のエントリ情報として各仮想空間の識別子を保持する．このため，仮想空間切換の際に TLB をフラッシュする必要がない．

一方，Pentium4_***AnT*** を SH-4 に移植した SH-4_***AnT*** は，ページサイズは 4KB のみ利用している．また，TLB へアドレス変換情報の登録を行う際，エントリの指定を行っていない．

4.3 高速化手法

4.3.1 方　　針

以下の方針で高速化する．

(方針 1) 現在のサーバプログラム間通信機構の基本的な機構には，変更を加えない．これは，サーバプログラム間通信のインタフェースの変更を防ぐためである．

(方針 2) サーバプログラム間通信の機能に制限を設ける．具体的には，OS サーバの数と ICA を利用する AP プロセスの数を制限する．これにより，制限内では，ICA の管理を簡素化し高速化する．これは，SH-4 は組み込みプロセッサであり，組み込みシステムでの制

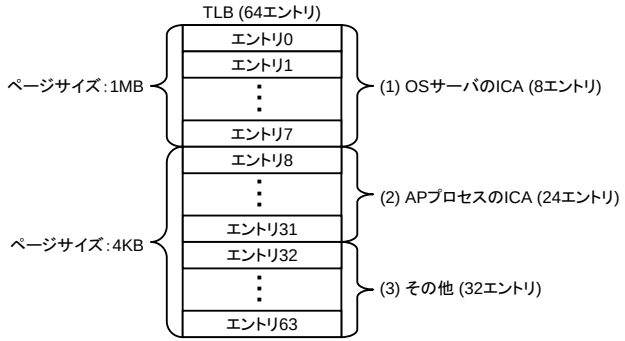

図 7 TLB エントリの分割管理

限された環境でのプログラム走行を想定したものである．

4.3.2 TLB のエントリの分割管理

SH-4 の TLB は 64 エントリで構成されている．また，ページサイズは 1KB，4KB，64KB，及び 1MB から選択できる．TLB のエントリを利用目的別に分割し，エントリ指定での TLB へのアドレス変換情報の登録を行うことにより，TLB ミスの回数を減らす．TLB のエントリの内訳を**図 7** に示し，以下で説明する．

(1) OS サーバの ICA

ページサイズを 1MB とし，8 エントリを割り当て，各 OS サーバは 1 エントリを使用するものとする．このため，OS サーバ数は最大 8 個，各 OS サーバで利用可能な ICA のサイズは最大 1MB である．

(2) AP プロセスの ICA

ページサイズを 4KB とし，24 エントリを割り当て，各 AP プロセスは 4 エントリを使用するものとする．このため，ICA を使用する AP プロセス数は最大 6 個，各 AP プロセスで利用可能な ICA のサイズは最大 16KB である．

(3) その他

ページサイズを 4KB とし，上記 (1)，(2) 以外へ 32 エントリを割り当てる．

4.3.3 ICA の管理方法

ICA のアドレス変換情報を常に TLB へ登録しておく．また，ICA の保護は TLB のエントリ情報の変更により実現し，AP プロセスからのアクセスに対してのみ行う．つまり，OS サーバは ICA を常に Read/Write 可能とする．これにより，ICA へのアクセスに対しては

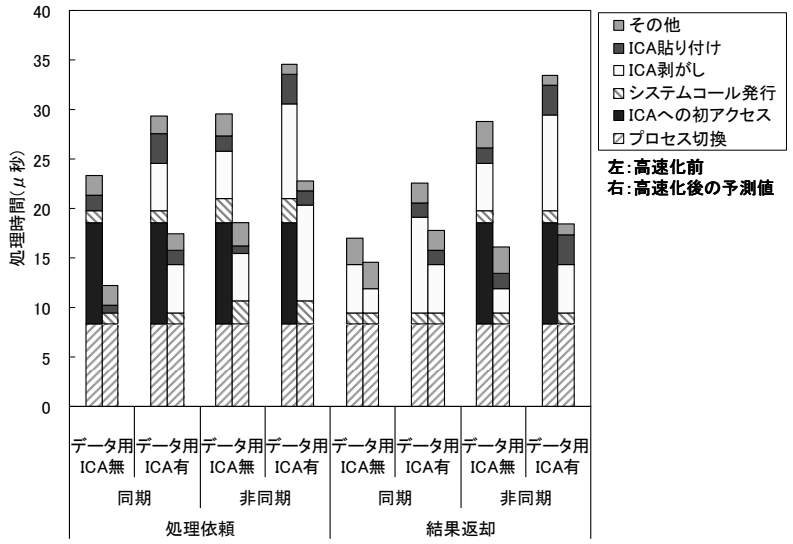

図 8 予測される基本処理の処理時間

TLB ミスが発生しないため，TLB ミスの回数を減らすことができる．また，ICA 貼り付けと ICA 剥がし処理のオーバヘッドを削減できる．

ICA の保護にはビットマップ管理表を用いる．ビットマップ管理表は，各 AP プロセスの ICA への Write 権限 (0 : Read Only，1 : Read/Write) を保持するものとする．ICA を保護する場合，対応するビットマップ管理表のビットを 0 に変更し，TLB の該当エントリ情報を Read Only に変更する．これにより，ICA への不正アクセス (Write) に対し，TLB 保護例外が発生するため，ICA を保護できる．また，マッピング表の簡略化により，マッピング表の探索に要するオーバヘッドを削減できる．

4.4 期待される効果

高速化手法を実現した場合の期待される効果について述べる．予測される基本処理の処理時間を**図 8** に示し，以下で説明する．

(1) 高速化手法では，ICA のアドレス変換情報は常に TLB に登録されている．このため，ICA への初アクセスの際のオーバヘッドは発生しない．

(2) 高速化手法では，OS サーバは ICA を常に Read/Write 可能である．このため，処理依頼の際の ICA 貼り付けのオーバヘッドを削減できる．

(3)　上記 (2) と同様の理由により，結果返却の際の ICA 剥がしのオーバヘッドを削減できる．

以上より，高速化手法は，サーバプログラム間通信の基本処理の処理時間について，処理依頼では約 30〜50 %，結果返却では約 10〜50 %オーバヘッドを削減可能である．また，サーバプログラム間通信の際にデータ用 ICA を用いた場合，データ用 ICA へのアクセスが発生する．この時，現方式では，データ用 ICA への初アクセスが発生するが，高速化手法では，発生しない．さらに，TLB エントリを有効的に利用可能であるため，処理全体の TLB ミスの発生率が減少すると期待できる．

5. おわりに

SH-4_***AnT*** におけるサーバプログラム間通信機構について，Pentium4_***AnT*** と比較し，性能評価を述べた．また，SH-4 の特徴を生かした高速化手法を示し，基本性能を予測した．

SH-4 と Pentium4 の動作周波数の比から，SH-4_***AnT*** におけるサーバプログラム間通信の性能を推定し，推定値と実測値を比較評価した．また，各プロセッサの特徴から測定結果の分析を行い，SH-4 の特徴を生かすことで，通信オーバヘッドを削減できる可能性を示した．また，高速化手法により，処理依頼では約 30〜50 %，結果返却では，約 10〜50 %のオーバヘッドを削減可能であることを示した．

残された課題として，SH-4 上で動作する ***AnT*** におけるサーバプログラム間通信の高速化と評価がある．

参考文献

1) J. Liedtke,“Toward Real Microkernels,”Commnuications of The ACM，Vol.39，Issue 9，pp.70-77，1996.
2) Andrew S. Tanenbaum, Jorrit N. Herder, Herbert Bos,“Can we make operating systems reliable and secure?,”IEEE Computer Magazine，Vol.39，No.5，pp.44-51，2006.
3) Black, D.L., Golub, D.B., Julin, D.P., Rashid, R.F., Draves, R.P., Dean, R.W., Forin, A., Barrera, J., Tokuda, H., Malan, G., and Bohman, D. ,“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.
5) 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におけるメモリ領域管理の設計と実現,”情報処理学会研究報告，2007-OS-104，Vol.2007，No.10，pp.33-40，2007.
6) 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 (D)，Vol.J93-D，No.10，pp.1977-1989，2010.
7) 鶴谷昌弘，山内利宏，谷口秀夫,“***AnT*** オペレーティングシステムの SH-4 への移植,”情報処理学会研究報告，Vol.2011-OS-117，No.3，pp.1-8，2011.