2008－EVA－27 (4)
2008／12／1

AnT におけるサーバプログラム間通信機構の評価

岡本幸大† 谷口秀夫†

独自 OS 開発における問題点の一つに，評価の困難さがある．開発した OS 単独で評価する場合，評価の妥当性を述べることが難しい．しかし，他 OS と比較する場合，提供する機能の違いや，動作環境の違いが比較を困難にする．本論文では，我々が独自に開発を進める *AnT* オペレーティングシステムのプログラム間通信機構に関して性能を測定し，評価を行う．

Evaluation of Inter Server Program Communication for *AnT*

KOUTA OKAMOTO† and HIDEO TANIGUCHI†

Evaluation is difficult in the original OS development. Describing validity of the evaluation is difficult on the original OS only. When we compare it with the other OS, the difference of offered functions and hardware requirements cause comparison to be difficult. In this paper, we measure performance about inter program communication of the *AnT* operating system that we develop originally, and evaluate it.

1. はじめに

計算機利用の多様化に伴い，用途に応じた OS が多く開発されている．独自に OS を開発する際の問題点として，OS 機能の評価がある．独自 OS の評価には，開発した OS 単独で評価を行う方法，および既存の OS と比較して評価を行う方法がある．前者の場合，開発した OS の特殊性から，機能や性能の評価結果が妥当であるか否かを示すことが難しい．一方，後者の場合，機能の違いや動作環境の違いのため，比較が難しい．

現在，我々は独自 OS である ***AnT***[1] オペレーティングシステムを開発している．***AnT*** はマイクロカーネル構造[2)3)4)5)] を有する OS であり，多くの OS 機能をプロセス (OS サーバ) として実現している．このため，OS サーバ間での通信性能は OS の性能に大きく影響を与える．そこで，***AnT*** のプログラム間通信機構[6)] に関して性能を測定し，***AnT*** 単独での評価結果を報告する．

2. プログラム間通信制御

2.1 コア間通信データ域

プロセスと内コアの間，あるいはプロセス間で授受するデータについては，コア間通信データ域[7)] (ICA : Inter-core Communication Area) と呼ばれる領域にデータを格納して，通信を行う．ICA は，内コアによりページ単位 (4KB) で管理されており，プロセス間での複写レス通信の機能を支援している．ICA に格納されたデータは，仮想空間のマッピング表を書き換えることにより複写レスでデータを授受する．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．また，剥がしと貼り付けの処理を合わせて貼り替えと呼ぶ．通信する 2 つのプログラム間でのデータ授受は，2 仮想空間の間での ICA の貼り替えにより実現できる．

2.2 通信機構

AnT におけるプログラム間の手続き呼び出しは，内コアが各プロセスに用意した通信のための依頼キューと結果キューを通して行われる．手続きの呼び出しと，処理結果の返却の流れを以下で説明する．なお，授受される ICA の先頭にはキュー管理用の領域が設けられている．

（1） 依頼元プロセスは依頼先プロセスの依頼キューに依頼情報を格納した ICA を登録し，依頼先プロセスへ ICA を貼り替える．

（2） 依頼先プロセスは依頼キューから依頼情報を格納した ICA を取得し処理を実行する．

（3） 依頼先プロセスは依頼元プロセスの結果キューに結果情報を格納した ICA を登録し，依頼元プロセスへ ICA を貼り替える．

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

（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ICAを取得し処理を終了する．

ここで，呼び出す手続きの内容に関する情報(引数や通信制御情報)と扱うデータを別のICAに格納し，手続きの内容に関する情報を格納したICAを制御用ICAと呼び，扱うデータを格納したICAをデータ用ICAと呼ぶ．制御用ICAは，データ用ICAへのポインタ情報を持っており，データ用ICAを利用したデータの持ち回りも複写レスで行うことができる．

2.3 非同期と同期の処理

ファイル管理機能や通信処理機能を提供するOSサーバは，多数の処理依頼を同時に受け付ける．このため，これらのOSサーバからの依頼発行は，非同期型でなくてはならない．一方，非同期型のインタフェース提供は処理が複雑化する．また，そのインタフェースも処理依頼と結果取得に分離されるため，使いやすいとはいえない．これに対し，同期型のインタフェース提供は逆の性質を持つ．そこで，利用に合わせたインタフェース提供として，非同期処理と同期処理の両インタフェースを提供する．

2.4 多段呼び出しと直接返却

OSサーバの連携のため，OSサーバの間を次から次に呼び出してゆく多段呼び出しを効率的に実現する．呼び出す度に新たな制御用ICAを確保する方法は，処理オーバヘッドが大きい．そこで，ひとつの制御用ICAに多段呼び出しに関する情報を積み重ねる．OSサーバの間を次から次に呼び出すに伴い，手続きの内容に関する情報を同じ制御用ICAに積み重ねる．返却の際は，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，結果の返却を可能としている．

多段呼び出しされた処理は，積み重ねられた情報を基に逐次的に結果返却の処理が行われる．しかし，必ず時も逐次的な返却が必要ではない．多段に処理を呼び出した最初のプロセスへ直接的に処理結果を返却できる場合もある．これを可能にするため，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．

2.5 データの保護

プロセス間のデータの転送にはいくつか方法がある．データの複写はデータの保護が強固である一方でオーバヘッドが大きい．また，共有メモリの利用は高速であるがデータの保護を困難にする．そこで，***AnT***ではデータ転送の際にICAの貼り付けと剥がしを利用することで，メモリアクセスを制限し，転送元プロセスのデータ書き換えによる不具合を抑制しながら，データの複写レスを実現する．

表1 評価環境
Table 1 Evaluative environment

OS	***AnT***
CPU	Intel(R) Pentium4 2.8GHz
メインメモリ	256MB(400MHz)
L1 キャッシュ	16KB
L2 キャッシュ	256KB

表2 部分処理の処理時間
Table 2 Processing time of associated operation

処理内容	処理時間
(1) ICA 貼り付け	0.14 μ秒
(2) ICA 剥がし	0.50 μ秒
(3) システムコール発行	0.54 μ秒
(4) ICA への初アクセス	1.36 μ秒
(5) プロセス切り替え	1.21 μ秒

3. 性能評価

3.1 基本性能分析

サーバプログラム間通信制御の基本的な処理として，同期処理依頼とその結果返却，およ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の授受を行なう場合と行なわない場合が存在する．これらについて性能を明らかにする．

依頼元プロセスと依頼先プロセスの2つのプロセス間での通信処理を行い，各基本処理の処理時間を測定する．処理依頼の際に，依頼先プロセスへ渡す引数，および戻り値は無しとする．また，以下に示す(1)〜(5)の部分処理に関して，処理時間と呼び出される回数を調査する．ここで，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，および内コアからプロセスへの処理の戻りを合わせて1回としている．つまり，依頼元プロセスがcallsync()を発行すると，プロセスから内コアへ処理移行し，依頼先プロセスへ依頼を発行した後内コアで待ち状態となる．依頼先プロセスは依頼の取得を内コアで待っており，依頼を受け取ると内コアからプロセスへ処理を戻す．この際，システムコール発行のカウントは1回となる．また，ICAへの初アクセスは，そのプロセスの仮想空間に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オーバヘッドである．これは，TLBミス等のキャッシュミスによるものであると考えられる．これらにより，基本処理の性能の違いを分析する．評価環境を表1に示す．なお，測定にはRDTSC命令を使用し，測定結果は10回試行した場合の平均処理時間である．

(1) ICA貼り付け

表 3　部分処理が呼び出される回数
Table 3　Invoked times of associated operation

		データ用 ICA	(1)	(2)	(3)	(4)	(5)
処理依頼	同期	無	1	0	1	1	1
		有	2	1	1	1	1
	非同期	無	1	1	2	1	1
		有	2	2	2	1	1
結果返却	同期	無	0	1	1	0	1
		有	1	2	1	0	1
	非同期	無	1	1	1	1	1
		有	2	2	1	1	1

(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) プロセス切り替え

各部分処理について処理時間を表 2 に示し，呼び出される回数を表 3 に示す．また，これらによる分析を行った基本処理の処理時間を図 1 に示す．これらより，以下のことが分かる．

(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 1 μ秒，非同期処理で約 0.7 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および ICA への初アクセスによるものであり，非同期処理では ICA への初アクセスおよびシステムコール回数の違いによるものである．

(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 1 μ秒，結果返却で約 1.5 μ秒である．この差分は，処理依頼では制御用 ICA 剥がしおよ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 ICA への初アクセスによるものである．

(3) データ用 ICA 有無に関して，それぞれの差分は約 0.6 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．

(4) 同期の結果返却においてのみ ICA への初アクセスが発生しない．これは，同期の場合，制御用 ICA を依頼元プロセスへ貼り付けたままとしているためである．これにより，同期の結果返却では高速な処理が可能である．

手続き呼び出しの処理は，処理依頼と結果返却を合わせて考える必要がある．処理依頼と結果返却を合わせて考えた場合，最も高速な通信は同期/データ用 ICA 無しの通信であり，約 6.5 μ秒である．また，最も低速な通信は非同期/データ用 ICA 有りの通信であり，約 9.8 μ秒である．また，同様にマイクロカーネル OS である L4 Linux[5] では，Pentium 133MHz CPU 上で，引数無しの RPC で 5 μ秒，ページマッピングを利用して引数渡しを行う RPC で 12 μ秒とある．両者を比較した場合，CPU 性能の違いを考慮すると基本的な性能は提案機構の方が低速であることがわかる．しかし，直接返却を利用することにより，性能の大幅な改善が見込める．直接返却に関する評価は，3.3 節に記述する．

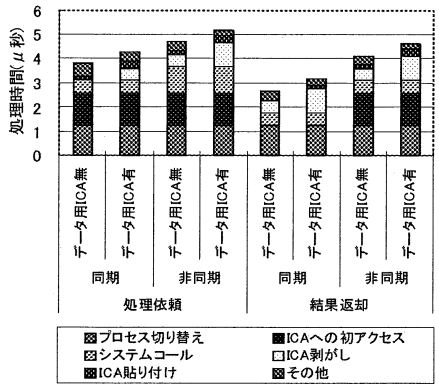

図 1　基本処理の処理時間
Fig. 1　Processing time of basic operation

3.2　保護機能のオーバヘッド

3.2.1　制御用 ICA の保護

提案方式では，制御情報を制御用 ICA に格納し通信を行う．通信の際には，制御用 ICA の貼り替えを行いプロセス間で制御情報の渡しを行う．この方式では，プロセスのメモリアクセス権を変更することでデータ保護が可能となるが，貼り替えによるオーバヘッドは性能低下の要因となる．一方，L4 等で利用される多くの RPC はデータ渡しの際に共有メモリを利用する．この方式では，通信性能の向上が可能であるが，データ保護が困難となる．そこで，本方式において，制御用 ICA を保護した場合と保護しない場合についての基本性能を測定し，メモリ保護のオーバヘッドについて考察する．

測定は，(1) 処理依頼/結果返却，(2) 同期/非同期，(3) データ用 ICA の有/無の 3 つのパターンを組み合わせた 8 通りの処理で，制御用 ICA を保護した場合と保護しない場合について行う．さらに，3.1 節と同様に制御用 ICA を保護しない場合の部分処理の呼び出される回数を調査し，性能の違いを詳細に分析する．評価環境，および測定方法は 3.1 節と同様である．

表 4　制御用 ICA 非保護における部分処理が呼び出される回数
Table 4　Invoked times of associated operation in unprotected control ICA

		データ用 ICA	(1)	(2)	(3)	(4)	(5)
処理依頼	同期	無	0[1]	0[0]	1[1]	0[1]	1[1]
		有	1[2]	1[1]	1[1]	0[1]	1[1]
	非同期	無	0[1]	0[1]	2[2]	0[1]	1[1]
		有	1[2]	1[2]	2[2]	0[1]	1[1]
結果返却	同期	無	0[0]	0[1]	1[1]	0[0]	1[1]
		有	1[1]	1[2]	1[1]	0[0]	1[1]
	非同期	無	0[1]	0[1]	1[1]	0[1]	1[1]
		有	1[2]	1[2]	1[1]	0[1]	1[1]

注：[] 内は制御用 ICA 保護の場合の回数

部分処理の呼び出される回数を表 4 に示し，分析を行った各処理時間を図 2 に示す．これらより，以下のことが分かる．

(1) 同期の処理依頼における制御用 ICA 保護/非保護の差分は約 1.5 μ秒である．これは制御用 ICA 貼り付け 1 回，および ICA への初アクセス 1 回のオーバヘッド削減によるものである．

(2) 非同期の処理依頼における制御用 ICA 保護/非保護の差分は約 1.8 μ秒である．これは，制御用 ICA の貼り付け 1 回，剥がし 1 回，および ICA への初アクセス 1 回のオーバヘッド削減によるものである．

(3) 同期の結果返却における制御用 ICA 保護/非保護の差分は約 0.4 μ秒である．これは，制御用 ICA 剥がし 1 回のオーバヘッド削減によるものである．

(4) 非同期の結果返却における制御用 ICA 保護/非保護の差分は約 1.8 μ秒である．これは，制御用 ICA の貼り付け 1 回，剥がし 1 回，および ICA への初アクセス 1 回のオーバヘッド削減によるものである．

同期処理に関して，制御用 ICA 保護/非保護の差分は処理依頼では約 1.5 μ秒であるのに対して，結果返却では約 0.4 μ秒と小さい．これは，同期処理では制御用 ICA 保護の場合でも制御用 ICA を依頼元プロセスへ貼り付けたままとしていることから，結果返却における制御用 ICA 剥がしのオーバヘッドが削減されたためである．一方，非同期処理に関しては，制御用 ICA 保護/非保護の差分は処理依頼と結果返却で共に約 1.8 μ秒である．これは，非同期処理では制御用 ICA 保護の場合には処理依頼と結果返却で共に制御用 ICA の貼り替えを行っているためである．

3.2.2　データ用 ICA の保護

前項では，制御用 ICA の保護/非保護に関して評価

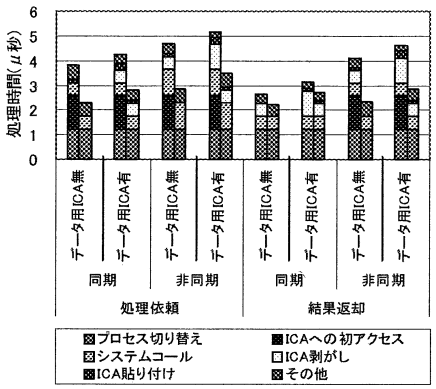

図 2　制御用 ICA の保護/非保護における基本処理の処理時間
Fig. 2　Processing time of basic operation in protected/unprotected control ICA

した．しかし，提案手法では制御用 ICA と同様にデータ用 ICA も貼り替えによる保護が可能である．そこで，本項では制御用 ICA とデータ用 ICA の両方に関して保護した場合と保護しない場合の性能を比較する．なお，OS サーバは，別 OS サーバへの処理依頼中に別の処理依頼を受け付けるために，処理依頼を非同期で発行する．このことから，本評価では非同期の場合のみを評価対象とする．

測定は，(1) 処理依頼/結果返却，(2) 制御用 ICA の保護/非保護，(3) データ用 ICA の保護/非保護の 3 つのパターンを組み合わせた 8 通りの処理で，基本性能について行う．評価環境，および測定方法は 3.1 節と同様である．

測定結果を図 3 に示す．測定結果より，以下のことがわかる．

(1) 制御用 ICA 保護の場合と非保護の場合における処理時間の差分は，約 1.8 μ秒である．これは，3.2.1 項で述べた非同期における制御用 ICA 保護/非保護の差分と同じであることから，制御用 ICA の貼り替え時間と ICA への初アクセス時間の合計であることがわかる．

(2) データ用 ICA 保護の場合と非保護の場合における処理時間の差分は，約 0.6 μ秒である．これは，3.1 節で述べた ICA の貼り替え処理時間と一致することから，データ用 ICA の貼り替え時間である．

上記 (1)(2) より，制御用 ICA の保護で発生するオーバヘッドは，制御用 ICA の貼り替えと ICA への初アクセスによるものである．しかし，データ用 ICA の保護で発生するオーバヘッドは，制御用 ICA の貼り

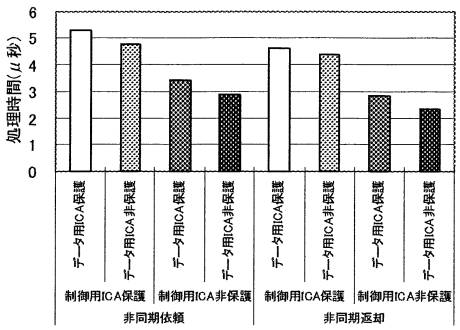

図 3 制御用 ICA とデータ用 ICA の保護/非保護における基本処理の処理時間

Fig. 3 Processing time of basic operation in protected/unprotected control ICA and protected/ unprotected data ICA

替えによるものだけである．この理由として，制御用 ICA は通信処理中にアクセスされるため ICA への初アクセスが発生するが，データ用 ICA は通信処理中にアクセスされないため ICA への初アクセスが発生しないことが原因である．また，データ用 ICA は通信処理中にはアクセスされないが，OS サーバの処理中にアクセスされた場合，やはり ICA への初アクセスのオーバヘッドが発生する．このことから，ICA を利用した 1 ページのデータ保護におけるオーバヘッドは，ICA の貼り替えと ICA への初アクセスを合わせた約 1.8 μ 秒であることがわかる．

3.3 直接返却

提案方式では，多段で処理依頼を行った場合に，依頼元プロセスへの処理結果を直接返却する機能を提供している．直接返却機能を利用することで返却処理の回数を減らし，処理の高速化が可能となる．そこで，直接返却機能について評価を行う．

まず，直接返却機能を利用した場合に効率化される処理時間を定式化する．直接返却機能では，OS サーバへの処理依頼が多段で発生した場合に，その返却処理を省略することで処理を高速化する．このため，OS サーバが 1 段のときは省略される返却処理はない．OS サーバが 2 段のときは 1 回の返却処理が省略される．OS サーバが n 段のときは n-1 回の返却処理が省略される．そこで，返却処理 1 回にかかる処理時間を R とし，直接返却により削減される処理時間を D とすると，以下の式で表すことができる

$$D = R(n-1)$$

ここで，直接返却を利用する場合，返却を省略される OS サーバは非同期で処理依頼を発行しなければならない．さもないと，処理の返却を待ち続けるため他の処理依頼を受け付けることが出来なくなる．このため，R は，非同期における結果返却処理時間となる．

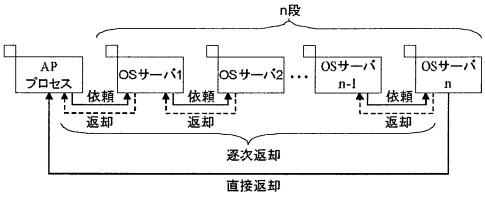

図 4 n 段の直接返却

Fig. 4 N step direct return

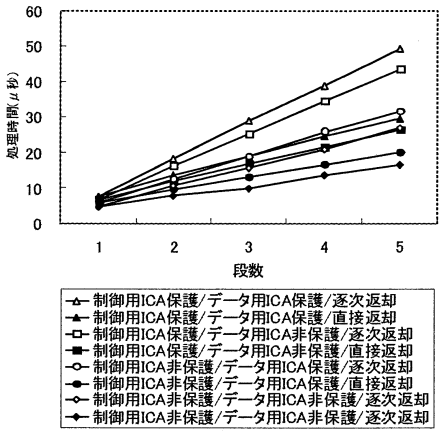

図 5 直接返却と逐次返却の比較

Fig. 5 Comparing direct return with sequential return

次に，直接返却を利用した場合の処理時間を測定する．図 4 に示すように OS サーバを n 段介した処理依頼において，AP プロセスが依頼を発行してから結果が返ってくるまでの処理時間を測定し，直接返却を利用した場合と逐次返却を利用した場合の処理時間を比較する．各 OS サーバは全て非同期で処理依頼を発行し，n は 1〜5 まで変化させる．また，(1) 制御用 ICA の保護/非保護，(2) データ用 ICA の保護/非保護の 2 つのパターンを組み合わせた 4 通りの処理でそれぞれ測定した．評価環境は 3.1 節と同様である．

測定結果を図 5 に示す．図より，逐次返却と直接返却では，共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．逐次返却の場合，処理依頼と結果返却は段数と同じ回数発行される．このため，逐次返却における処理時間の増加は処理依頼と結果返却の回数増加によるものである．一方，逐次返却の場合，処理依頼は段数と同じだけ発行されるのに対し，返却処

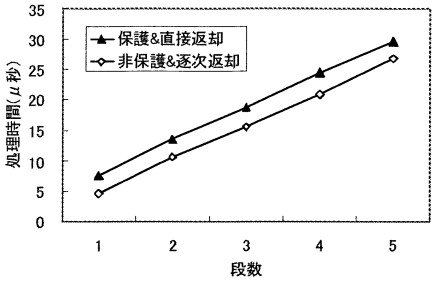

図 6 保護&直接返却と非保護&逐次返却の比較
Fig. 6 Comparing protected & direct return with unprotected & sequential return

理はどの場合も1回のみ発生する．このため，直接返却における処理時間の増加は処理依頼の回数増加によるものである．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て約55%に抑えられていることが分かる．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が高いことが分かる．

さらに，本測定において，制御用ICA保護/データ用ICA保護の直接返却(保護&直接返却)が提案方式を利用した通信方式である．一方，制御用ICA非保護/データ用ICA非保護の逐次返却(非保護&逐次返却)が，多くのOSで利用される共有メモリを利用した通信方式である．そこで，保護&直接返却と非保護&逐次返却の通信方式を比較する．なお，保護&直接返却では，直接返却を利用することで結果返却にかかるオーバヘッドが削減され，非保護&逐次返却では，共有メモリを利用することで処理依頼と結果返却におけるデータ保護のオーバヘッドが削減される．

比較の結果を図6に示す．図6より，保護&直接返却と非保護&逐次返却におけるグラフの傾きが同じとなっていることが分かる．これは，各方式の削減オーバヘッドが同等であるためと推察できる．なお，両方式では処理時間に約5 μ秒の差分がある．これは，基本とする処理時間，つまり1段の場合の処理時間に差があるためであり，この差はデータ保護のオーバヘッドである．

以上より，提案方式である保護&直接返却では直接返却機能を利用することで，段数増加による処理時間の増加の割合を，非保護&逐次返却の方式と同等に抑えることが可能となることを明らかにした．また，両者の間に約5 μ秒の差が存在するが，これは1段の場合のデータ保護によるオーバヘッドである．

4. おわりに

独自に開発を進める ***AnT*** オペレーティングシステムについて，プログラム間通信制御の性能を測定し，評価した．まず，基本となる通信処理の性能について分析し，その違いを明確化した．また，データ保護のオーバヘッドについて評価し，ICAを利用することで，約1.8 μ秒のオーバヘッドと引き換えに，1ページ分のデータ保護が可能となることを明らかにした．さらに，直接返却機能を利用することで，段数増加による処理時間の増加の割合を，逐次返却に比べ約55%に抑えることが可能であることを明らかにした．

残された課題として，本機能を使い易くしたライブラリの実現がある．

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，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.
2) 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(19920315), pp.442-453, 1992.
3) Andrew S. Tanenbaum, Albert S. Woodhull, “Operating Systems Design and Implementation Third Edition,” Prentice Hall, ISBN 0-13-142938-8, 2006.
4) J. Liedtke, “Improving IPC by kernel design, ”Proc. 14th ACM Symp. Operating System Principles, ACM Press, pp.175-188, 1994.
5) H. Hartig et al., “The Performance of Microkernel-Based Systems, ”Proc. 16th ACM Symp. Operating System Principles, ACM Press, pp.66-77, 1997.
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サーバプログラム間通信機構の評価，”電子情報通信学会技術研究報告，Vol.107，No.558，pp.49-54，2008.
7) 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.40，pp.33-40，2007.