2007－OS－105 (4)
2007／4／5

Tender の資源再利用機能を利用した fork&exec 処理の高速化

佐伯 顕治†, 田端 利宏‡, 谷口 秀夫‡

†岡山大学工学部 ‡岡山大学大学院自然科学研究科

OS の処理の中でも，プロセスを生成する fork と exec 処理の負荷は大きく，サービスの処理性能への影響は小さくない．そこで，我々は、分散指向永続オペレーティングシステム ***Tender*** において，OS の構成資源を再利用することによるプロセスの生成と消滅の高速化を実現している．***Tender*** では，他 OS 用のプログラム実行のため，UNIX 系 OS である BSD/OS のインタフェースを実現している．本論文では，資源再利用機能を利用して，fork と exec システムコール処理を高速化する手法を述べ，再利用する資源の種類ごとに行った評価結果について報告する．また，Apache web サーバでの評価を行うことで，実 AP での資源再利用機能による fork と exec システムコール処理の高速化の効果を示す．また，他の OS と比べることで，資源再利用機能の有効性を示す．

Speed-up of fork & exec System-call by Recycling Resource on *Tender*

Kenji SAEKI†, Toshihiro TABATA‡, and Hideo TANIGUCHI‡

† Faculty of Engineering, Okayama University
‡ Graduate School of Natural Science and Technology, Okayama University

The cost of process creation is high in the processing of OS. The costs degrade the performance of program execution. To solve this problem, we have implemented fast process creation and disappearance by recycling process resource in ***Tender***. ***Tender*** has BSD/OS interface for program execution of the BSD/OS. In this paper, we describe the method of speed-up of fork and exec system-call by recycling resource in ***Tender***, and evaluate fork and exec system-call by recycling each resource. Moreover, we report the effect of recycling resource by using Apache web server.

1. はじめに

プロセスの生成処理は，メモリ空間の生成やプログラムの読み込み処理を必要とするため，オペレーティングシステム（以降 OS と略す）の処理の中でも負荷が大きい．このため，プロセスの生成と削除の処理が頻発する応用プログラム（以降 AP と略す）では，サービスの処理性能が低下することがある．例えば，Web サーバが CGI プログラムを実行した場合，プロセスの生成が行われるため，Web サーバの応答時間は長く

なる．このため，プロセスの生成処理を高速化する研究が行われている．プロセスの生成処理を高速化する研究としては，UNIXでの sticky bit や vfork システムコールがある．

我々は分散指向永続オペレーティングシステム ***Tender***(The ENduring operating system for Distributed EnviRonment)において，資源独立化機構を生かして，プロセスを構成する資源(以降，プロセス構成資源と略す)を再利用する機能(以降，資源再利用機能と略す)を用いることでプロセスの生成と削除の処理を高速化できることを示した[1]．また，***Tender*** では，他の OS で利用されている AP を用いた性能評価を目的として，BSD/OS 互換システムコールインタフェース(以降 I/F と略す)[5]を実装している．

そこで，我々は，資源再利用機能によるプロセス生成処理高速化の有効性を示すために BSD/OS システムコールである fork システムコール(以降 fork と略す)と execve システムコール(以降 execve と略す)に資源再利用機能を適用した．また，実現した fork と execve の評価結果を報告する．さらに，実際にプロセスの生成と削除が頻発する事例として，CGI プログラムの実行を伴う Web サーバへのアクセスを取り上げ，***Tender*** 上で Apache Web サーバを利用した評価結果を報告する．

2. *Tender* オペレーティングシステム

2.1. 資源の分離と独立化

Tender は，操作する対象を資源として分離し，独立化している．資源には，資源識別子と資源名を付与し，資源操作のインタフェースを統一している．また，資源の分離と独立化を行うことで，資源の事前生成や保留が可能になり，資源の生成や削除に伴う処理を高速化できる．

2.2. プロセス構成資源

プロセス構成資源を図 1 に示す．ここで，矢印は，資源の依存関係を表している．資源「プロセス」とは，プロセス識別子とプロセス管理表からなり，OS がプログラムの動作を制御する単位になる．資源「プログラム」とは，実行プログラムのテキスト部とデータ部の大きさと先頭アドレス，およびプログラムの開始アドレス情報からなり，プログラムの実行形式を隠蔽している．プログラムの内容は，資源「プレート」上に存在している．ここで，資源「プレート」とは，永続的な記憶をメモリ上に提供するものであり，既存の OS のファイルに相当する．

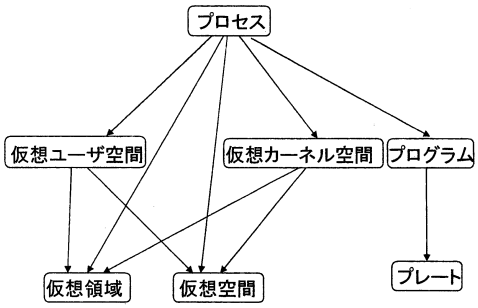

図 1　プロセス構成資源

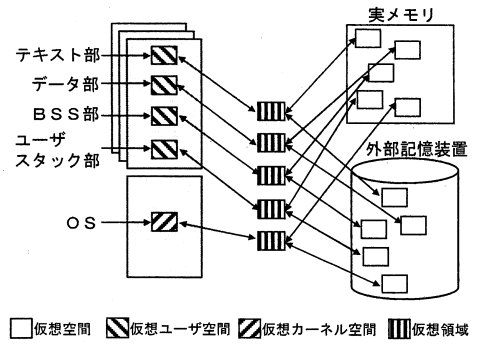

図 2　プロセス構成資源の関係

ここで，プロセス構成資源間の関係を図 2 に示す．資源「仮想空間」とは，仮想アドレスの空間であり，仮想アドレスを実アドレスに変換する変換表に相当する．資源「仮想領域」とは，メモリイメージを仮想化した領域であり，仮想化の実体は，実メモリ，または外部記憶装置に上に存在する．資源「仮想カーネル空間」，ならびに資源「仮想ユーザ空間」とは，プロセッサが仮想ア

ドレスによって，前者はカーネルモードのみ，後者はユーザモードでもアクセス可能な空間である．両者は共に，仮想空間が持つアドレス変換表に，当該の仮想領域のデータ格納情報を設定する(以後，貼り付けると呼ぶ)ことで生成することができる．逆に貼り付けた際に設定したデータ格納情報を解放する(以後，剥すと呼ぶ)ことで，削除することができる．プロセスのテキスト部，初期値を持つ変数や文字列の集合部分であるデータ部，初期値を持たない変数や文字列の集合部分である BSS 部，およびユーザスタック部は，仮想ユーザ空間に読み込まれた状態で仮想空間上に存在する．

2.3. 資源再利用機能

資源再利用機能[1][3]とは，プロセス削除時にプロセス構成資源の削除処理を保留し，プロセス生成時に，プロセス構成資源の生成処理の代わりに資源の再利用を行う機能である．再利用可能なプロセス構成資源は，「プログラム」，「仮想領域」，「仮想空間」，「仮想ユーザ空間」，および「ワーク用仮想カーネル空間」の５つである．

Tender では，ワーク用仮想カーネル空間は再利用で常に効果があり，用いる領域の大きさも小さいので，常に再利用される．

プログラムはプログラムの内容が同じ場合，再利用される．仮想領域は text 部，data 部，bss 部，およびユーザスタック部について再利用可能であり，仮想領域の大きさが同じであれば，再利用される．仮想空間は，仮想ユーザ空間が貼り付いていない場合は，常に再利用できる．仮想ユーザ空間は仮想空間に仮想領域が貼り付けられた状態のまま仮想空間を再利用する場合にのみ，再利用可能である．また，text 部については仮想領域とプログラムの内容が同じであれば，テキスト部の仮想領域を共有することでテキスト部の内容を利用することができる．仮想領域とプログラムが同時に再利用される場合は，text 部の仮想領域を常に共有できるようにすることで，text 部の内容を再利用することができる．

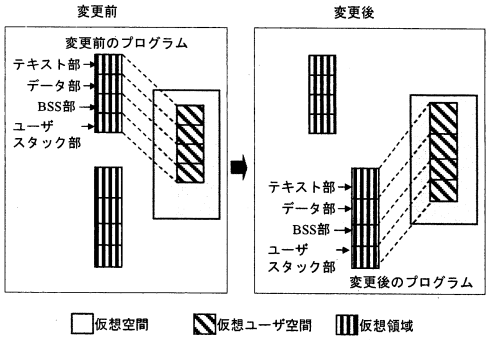

図３　実行プログラム変更機能

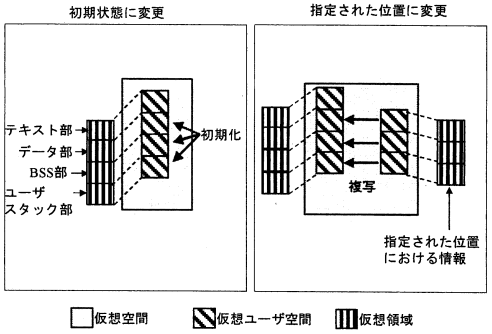

図４　開始位置変更機能

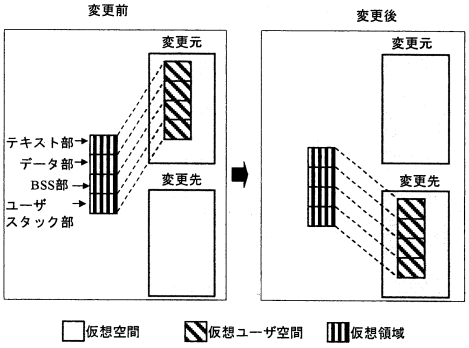

図５　動作空間変更機能

2.4. プロセス変身機能

Tender はプロセスを構成する要素を変更し，プロセスの実行環境を変更する機能を備えている．この機能をプロセス変身機能[4]と呼び，図 3，4，5 に示す以下の 3 つの機能で構成している．

表１　fork システムコールと execve システムコールの再利用可能な資源

システムコール	fork	execve
再利用可能な資源	仮想空間，仮想領域， プログラム，テキスト部の内容	仮想領域，プログラム， テキスト部の内容

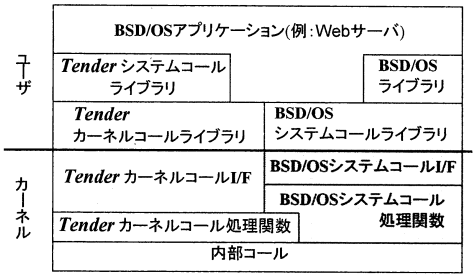

図６　BSD/OS 互換システムコール I/F

(機能１)実行プログラム変更機能
(機能２)開始位置変更機能
(機能３)動作空間変更機能

実行プログラム変更機能とは，プロセスとして実行されるプログラムを変更する機能である．開始位置変更機能とは，変更対象プロセスが次に走行する開始位置を変更する機能である．動作空間変更機能とは，プロセスが利用する仮想空間を別の仮想空間に変更する機能である．また，これらの３つの機能は，単独で利用するだけでなく，組み合わせて利用可能である．

また，**(機能１)(機能３)**は資源再利用機能を用いることで高速化を実現している．具体的には**(機能１)**では図３に示すように，実行プログラムの変更を行うため，仮想領域，プログラムおよび text 部の内容が再利用可能である．**(機能３)**では図５に示すように，動作空間の変更を行うため，仮想空間が再利用可能である．いっぽう，**(機能２)**は図４で示すように，プログラムの開始位置の変更を行うだけで，再利用できるプロセス構成資源が存在しないため，資源再利用機能を用いた高速化は行われていない．

2.5. BSD/OS 互換システムコール I/F

Tender では BSD/OS の実行形式(a.out)をサポートしているため，BSD/OS プログラムからプロセスを生成し，実行することができる．

BSD/OS 互換システムコール I/F を図６に示す．***Tender*** では BSD/OS システムコール処理関数を実装することで，BSD/OS AP の発行するシステムコール番号と引数を取得している．BSD/OS システムコール処理関数は BSD/OS システムコールと同等の処理を行い，***Tender*** の内部コールやカーネルコール処理関数により実現している．

3. fork&exec システムコールの高速化

BSD/OS 互換システムコール I/F の fork システムコールと execve システムコールに資源再利用機能を適用し，実装を行った内容ついて述べる．

3.1. fork

fork をカーネルプロセスの生成，プロセス変身機能**(機能３)**を用いた動作空間の変更，プロセス変身機能**(機能１)**を用いた実行プログラムの変更，プロセス変身機能**(機能２)**を用いたプログラムの開始位置の変更，プロセスが持つ情報の複写，資源「演算」の作成と割り当てによって実現した．fork では．プロセス変身機能**(機能１) (機能３)**を用いているため，仮想空間，仮想領域，プログラムおよび text 部の内容が再利用できる．

3.2. execve

execve を資源「プレート」の有無の確認，プロセス変身機能**(機能１)**を用いた実行プログラムの変更，演算に関する情報の書き換えによって実現した．execve ではプロセス変身機能**(機能１)**を用いているので，仮想領域とプログラムを再利用できる．また，

text 部の内容も再利用できる．

3.3. fork と execve で再利用可能な資源

表 1 に fork と execve で再利用可能な資源をまとめた結果を示す．表 1 から，仮想空間，仮想領域，プログラムおよび text 部の内容が再利用できることがわかる．

4. 評価・考察

4.1. fork と execve の評価

Tender に実現した fork システムコールと execve システムコールにおける資源の再利用の効果を評価するために，実測による評価を行った．測定を行う対象処理は，fork と exec において，各再利用可能な資源を再利用した場合の処理時間である．測定では，まず親プロセスが fork を行う．それから，子プロセスは execve を行って何も処理を行わないプログラムを呼び出して終了し，親プロセスは wait で子プロセスの終了を待つ．この処理をハードウェアカウンタにより計測した．ただし，資源再利用機能を計測するので，初めて fork と execve をする処理は除き，2 回目以降を計測した．

測定するプログラムの基本となる大きさは親，子プロセス共に text 部の大きさを 20KB，data 部の大きさを 4KB，および bss 部の大きさを 4KB として，text 部，data 部，bss 部の大きさを変化させて評価した．

測定には，Celeron 2.8GHz の計算機を用いた．

4.2.1. fork の資源再利用の効果

親プロセスの text 部の大きさを変化させた場合を図 7 に，data 部の場合を図 8 に，bss 部の場合を図 9 に示す．

図 7 より fork するプログラムは text 部の大きさが大きくなっても処理時間は増大しない．これは text 部の共有によって fork するプログラムを作成しているためだと考えられる．

図 8 と図 9 より fork するプログラムは data 部，bss 部の大きさに比例して処理時間は増大する．ただし，bss 部は初期値を持たない分，処理時間は短くなっている．

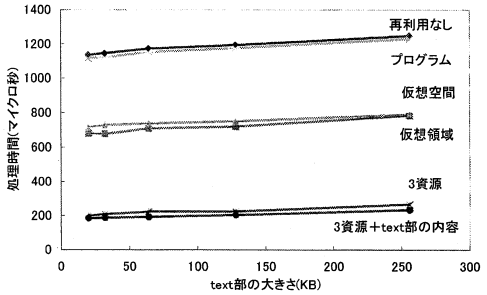

図 7　fork の処理時間(text 部可変)

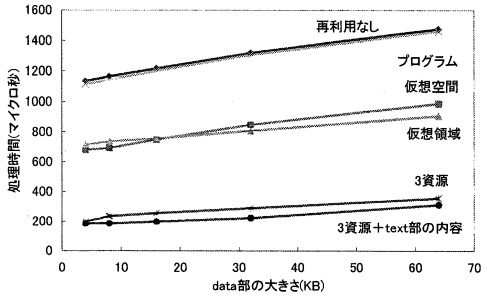

図 8　fork の処理時間(data 部可変)

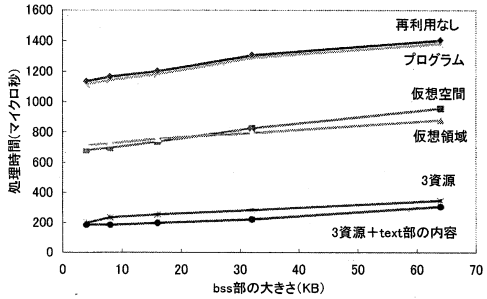

図 9　fork の処理時間(bss 部可変)

図 7，図 8，および図 9 よりプログラムの再利用効果は少ない．これはプログラムの再利用は負荷が大きくないメモリ上に存在するプログラムのデータの大きさと場所情報の再利用を行うためである．

仮想空間の再利用は領域の大きさに関係なく，処理時間を約 500 マイクロ秒短くできる．

仮想領域の再利用は，領域の大きさに比例して再利用効果が高くなる．これは，仮想領域の生成時間は，再利用する場合は一定だが，再利用しない場合は，生成時間が領域の大きさに比例して遅くなるためである．

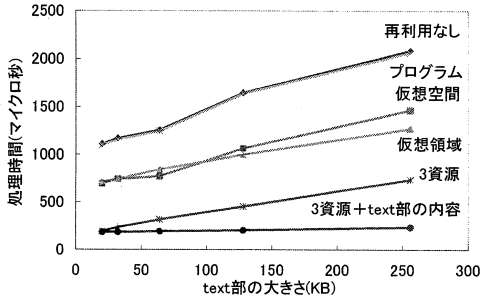

図 10　execve の処理時間(text 部可変)

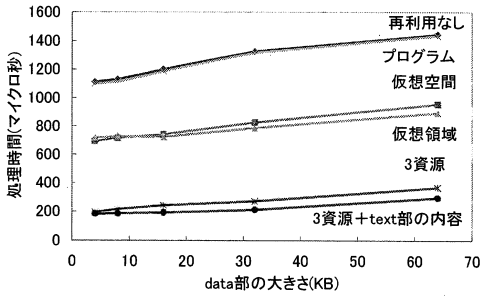

図 11　execve の処理時間(data 部可変)

3 資源(プログラム，仮想空間，仮想領域)の再利用はプログラム，仮想空間，仮想領域をそれぞれ再利用した効果の和にほぼ等しい．

3 資源＋text 部の内容の再利用は最も再利用効果が高く，処理時間を資源再利用しない場合に比べ、20%以下に抑えることができる．

4.2.2. execve の資源再利用の効果

図 7，図 10 より，fork するプログラムより，execve するプログラムの text 部の大きさが処理時間に影響を与えやすいことが分かる．これは，execve のプログラム実行時に text 部の共有が行われていないためである．また，資源の再利用によってその影響を抑えることができることも分かる．

図 10 より，text 部の内容の再利用は execve するプログラムの大きさに比例して効果が高くなる．

図 8，図 9，図 11，および図 12 より data 部と bss 部の execve の資源の再利用効果は fork と比べても大差がない．

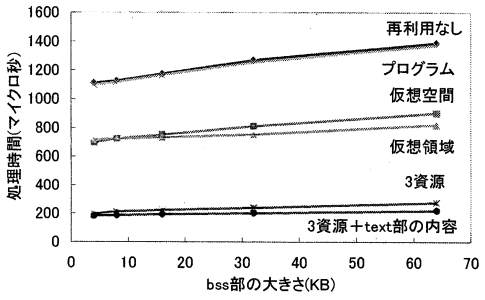

図 12　execve の処理時間(bss 部可変)

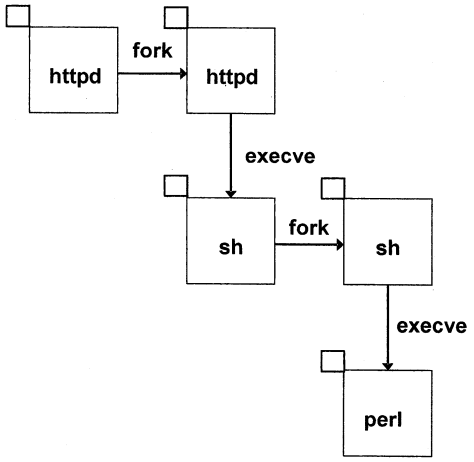

図 13　CGI プログラム実行時の処理の流れ

なお，***Tender*** では Copy-on-Write(以降 CoW と略す)を用いないため，プロセス生成時にすべての内容をメモリに複写している．一方，BSD/OS などの OS は CoW を実現しているため，メモリの複写がなく，プロセス生成は高速である．ただし，実行時にページ例外によるメモリ複写が発生する．このため，両者の比較を行うには，実 AP での評価が必要である．

4.2 Web サーバによる評価

Web サーバがプロセス生成を行う事例として，CGI プログラムの実行を伴う処理を取り上げる．Web サーバは Apache を利用した．CGI プログラムの実行を伴う処理を Web サーバに要求し，その応答時間を評価する．

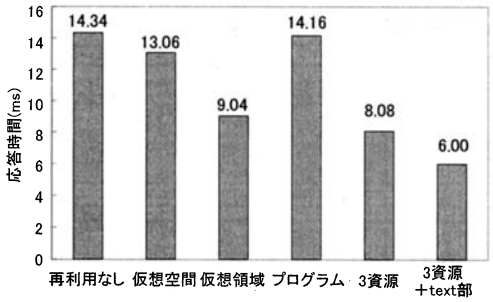

図 14　Web サーバにおける資源再利用の効果

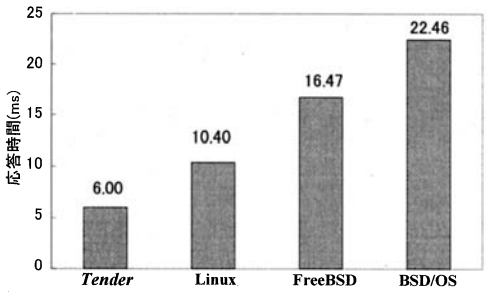

図 15　各 OS の Web サーバの応答時間

4.3 CGI プログラムの実行

評価に用いた Apache の例として，図 13 に CGI プログラムの実行の流れを示す．CGI プログラムは，Web サーバがシェルを呼び出し，シェルが Perl を呼び出すことで実行されている．つまり一回の処理で 2 つのプロセスが生成される．CGI プログラムはアクセスカウンタ等で非常に多く用いられているため，これらのプロセス生成処理が Web サーバの処理性能に大きな影響を与えている．

本評価では，CGI プログラムとして，Perl によって記述されたアクセスカウンタを利用した．CGI プログラムはカウンタの値を 1 つ増やして，その値を Web サーバに送信し，ファイルに格納する．

4.2.2 評価環境

サーバ計算機（CPU：Celeron 2.8GHz，OS：***Tender***）1 台とクライアント計算機（CPU：Pentium4 3.0GHz，OS：FreeBSD4.3-RELEASE）1 台を 100Base-TX の Ethernet で接続して評価を行った．Web サーバは Apache 1.3.33，ベンチマークツールは ApacheBench を利用した．

4.4. 評価結果

Web サーバにおける資源再利用機能の効果を図 14 に示す．図 14 から以下のことがわかる．

(1) 仮想空間の再利用を用いることで 1.28ms(=14.34ms-13.06ms) 応答時間を短縮できる．
(2) 仮想領域の再利用を用いることで 5.40ms(=14.34ms-9.04ms)応答時間を短縮できる．
(3) プログラムの再利用を用いることで 0.18ms(=14.34ms-14.16ms) 応答時間を短縮できる．
(4) 3 資源の再利用を用いることで 6.26ms(=14.34ms-9.04ms)応答時間を短縮できる．
(5) 3 資源+text 部の内容の再利用を用いることで 8.34ms(=14.34ms-6.00ms)応答時間を短縮できる．

以上のことから実 AP においても資源再利用機能は効果があることを示した．

また，***Tender***，**Linux**，**FreeBSD**，および **BSD/OS** の Web サーバの応答時間を図 15 に示す．ただし，***Tender*** は 3 資源＋text 部を再利用した場合の応答時間である．これより，***Tender*** は 3 資源+テキスト部の内容を再利用することで他の OS よりも速く，CGI プログラムの処理を伴う Web サーバ処理が行えることがわかる．

5. まとめ

Tender における資源再利用機能を利用した fork システムコールと exec システムコールの高速化手法について述べた．これらのシステムコールの実装では，プロセス変身機能を利用しており，プログラム，仮想空間，および仮想領域が再利用可能であることを示した．

資源再利用機能別に fork と execve の性能評価を行い，プログラム，仮想空間，仮想領域，3 つの資源，および 3 つの資源+text 部の内容の再利用の効果を明らかにし，資

源再利用により，プロセスの生成処理を高速化できることを示した．また，Web サーバでのプロセス生成が行われる事例として，CGI プログラムの実行を伴う処理を取り上げ，Web サーバの応答時間の評価を行った．この結果，CGI プログラムの実行を伴う処理時の資源再利用機能の効果（最大で約58%高速化）を明らかにした．さらに，他の OS との比較を行い，***Tender*** は 3 つの資源とテキスト部の内容を再利用することで，他の OS よりも速く，CGI プログラムの処理を伴う Web サーバ処理が行えることを示した．

謝辞 本研究の一部は，科学研究費補助金若手研究(B)（課題番号 18700030）による．

参考文献

[1]田端利宏，谷口秀夫：プロセス構成資源の効率的な再利用を目指した資源管理方法の提案，情報処理学会論文誌：コンピュータシステム，Vol44，No.SIG10(ACS2)，pp.48-61(2003).

[2]野村和孝，田端利宏，谷口秀夫：プロセス変身機能における資源再利用効果，情報処理学会研究報告，2004-OS-96，Vol.2004，No.21，pp.149-156(2004).

[3]谷口秀夫，青木義則，後藤真孝，村上大介，田端利宏：資源の独立化機構による ***Tender*** オペレーティングシステム，情報処理学会論文誌，Vol.41，No.12，pp.3363-3374(2000).

[4]石井陽介，谷口秀夫：位置透過に利用可能な構成要素を用いたプロセスの変身機能，情報処理学会論文誌，Vol.44，No.7，pp.1666-1679
(2003).

[5] 田端利宏，野口直樹，中島耕太，谷口秀夫：Web サーバを利用した ***Tender*** の資源「演算」の評価 －複数プロセスの実行性能調整機能－，情報処理学会研究報告，2002-OS-090，Vol.2002，No.60，pp.33-40(2002).