実行中プログラム部分入替え法における入替え処理時間の短縮

中島雷太[†] 谷口秀夫[†]

提供中のサービスを停止することなく，そのプログラムの一部を変更することができれば，より柔軟な計算機環境を構築することができる．そこで，我々は，実行中プログラムの一部分を入れ替える制御法を提案した．また，主に入出力処理を行うプログラムでは，入替え対象であるプログラム部分の入替え処理にかかる時間は，入替え対象であるプログラム部分の処理時間に依存し，比較的短いことを示した．本論文では，主にプロセッサ処理を行うプログラムの場合や，他にプロセスが同時走行している場合の入替え時間について評価を行い，提案した入替え法では，入替え処理時間が増加する問題点を明らかにする．次に，この問題点への対処法を示す．さらに，対処法を実現して評価し，その有効性を示す．

Reduction of the Exchange Processing Time about Mechanism for Run-time Replacement of Program Parts

Raita Nakajima[†] and Hideo Taniguchi[†]

We have proposed the mechanism for run-time replacement of program parts. We have reported the evaluation of exchanging time about I/O processing program. And we have described that the exchanging time depended on the processing time of program parts. In this paper, we describe the problem of proposed mechanism about the exchange processing time. We suggest two methods about improvement of the exchange processing time. One method is the early suspend of processes which condition are able to exchange and the other is that the execution of processes which condition aren't able to exchange gives priority. Furthermore, we describe the availability of the two methods.

1. まえがき

近年，計算機の普及によって，計算機を利用する環境が急激に変化しつつある．このような状況においては，計算機の提供するサービスも，その計算機の利用環境に素早く対応することが望ましい．また，計算機を利用した長期的なサービスの提供では，社会に与える影響を考えると，その保守作業などは，速やかに完了することが望ましい．たとえば，銀行系システムに代表される大規模なオンライントランザクション処理システムにおいては，通信回線の集線による通信コスト低減を目的にホスト計算機と端末の間を端末制御装置で接続している．ホスト計算機に比べ台数が多い端末制御装置では，装置コストを抑制する必要がある．このため，システムの 24 時間連続運転を実現する際，ホスト計算機に比べ，端末制御装置での対処は限られてしまう．プログラムの不具合などによりプログラムを変更する処理は，主に毎朝の端末制御装置立ち上げを契機にホスト計算機から情報を受信しパッチにより対処しているが，24 時間連続運転では立ち上げ契機を失ってしまう．このため，サービス提供中のプログラム変更が必要になっている．また，急増している WWW サーバでも 24 時間連続運転が通常であり，安価な WWW サーバでもサービス提供中にセキュリティホールを埋めるためのプログラム変更が可能になれば，利便性が向上する．

このような背景から，計算機上で実行中のプログラムの処理を停止することなく，機能の一部分を動的に入れ替える実行中プログラム部分入替え法を提案した．

同様の関連研究として，プロセスを冗長構成にすることでプロセス単位で入れ替える方法[1)]が提案されている．この方法では，プロセスを 1 つの単位として入れ替えるため，少しの変更であってもプロセス全体を入れ替える必要があり，現プロセスの走行を継続させることはできない．また，プロセスを走行させたままプログラムの一部分を入れ替える方法[2)]も提案されている．この場合には，入替えを行うプログラムが入替

† 九州大学大学院システム情報科学研究科
Graduate School of Information Science and Electrical Engineering, Kyushu University

えの契機を考慮し，オペレーティングシステム（以降「OS」と略す）に依頼する必要がある．このため，入替え処理工数の増加を招いたり，入替え契機を意識したプログラム作成が必要となる．また，同一のプログラムによって生成された複数のプロセスが動作する場合には，プログラム処理構造が複雑となり，入替え法の利便性を大きく低下させる．さらに，マイクロカーネルに基づいたOS構成法によって，OSに柔軟な拡張性を与える研究がある[3]．しかし，残念ながらマイクロカーネルモデルは，モノリシックカーネルモデルに比べて実行効率が悪い[4),5)]．プロセススケジューリング機構をはじめとするOS機能を対象としたOS上での動的再構築や機能拡張に関する研究[6)~10)]もあるが，アプリケーション（以降「AP」と略す）を対象とはしていない．

これらの関連研究に対し，我々が提案したプログラム入替え法は，次の大きな2つの特徴を持つ[11)]．1つは，サービスプログラムが，入替え契機を意識する必要がない点である．もう1つは，入替え対象プログラム部分が入替え対象でない別のプログラム部分を呼び出している場合にも，プログラムの入替えを可能としている点である．さらに，複数のプロセス間で共有されたプログラム部分の入替えも考慮している[12)]．これまでに，基本方式を提案し，実装，評価することによって，その有効性を示した．また，入替え要求から入替え処理を終了するまでの入替え処理時間（以降「入替え時間」と呼ぶ）の観点から評価を行い，定式化も行った[13)]．具体的には，主に入出力処理を行うプログラム（以降「入出力処理プログラム」と呼ぶ）に対する入替え時間を測定評価した．

本論文では，さらに主にプロセッサ処理を行うプログラム（以降「プロセッサ処理プログラム」と呼ぶ）に対する入替え時間の評価を行い，入出力処理プログラムに比べ，プロセッサ処理プログラムに対する入替えは，入替え時間が大きく増加することを示す．また，入替え対象であるプログラム部分を共有しないプロセス（以降「他プロセス」と呼ぶ）が同時走行すると，さらに入替え時間が増加することも示す．入替え対象プログラム部分を入れ替えるには，そのプログラム部分が実行されていない必要があるため，入替え対象プログラム部分への処理移行を停止させる．この結果，サービスに対して少なからず影響が生じる．したがって，増加した入替え時間の短縮化を図る必要がある．そこで，入替え時間が増加する問題への対処として，入替え時間を短縮する手法を提案し，実装と評価によりその有効性を示す．以降，2章では，提案した入替え法の基本方式について述べる．3章では，基本方式が持つ問題点として，入替え時間が増加する問題を示し，その問題への対処として，入替え時間を短縮する手法について述べる．具体的には，入替え処理を行うことができる状態（以降「入替え可能状態」と呼ぶ）にあるプロセスを早期に停止させる．また，入替え可能状態でないプロセスを優先的に実行させる．4章では，両手法についての測定結果を示し，その有効性を示す．また，既存ソフトウェアを擬似したプログラムによる評価の結果を示す．5章にて本論文をまとめる．

2. 入替え法

2.1 基本方式

実行中プログラムの一部分を入れ替える機能を実現するための課題として，これまでに以下の5つ課題を示しその対処法を示した[11)]．

(1) プログラム実行状態の分類
(2) 入替え条件
(3) プログラム状態の管理法
(4) 有限時間内での入替えを保証する制御法
(5) 入替え対象プログラムのサービスへの影響を軽減する方法

ここでは，本論文の内容と特に関連が深い項目(1)，(2)および(4)について簡単に述べる．

2.1.1 プログラム実行状態の分類

プログラム部分の実行状態は，**図1**に示すように3つの状態に分類することができる．入替え対象となるプログラム部分Yに対し，3状態は以下のようになる．

(1) 未使用：実行する前，あるいは，実行を終えた状態
(2) 走行中：実行中の状態

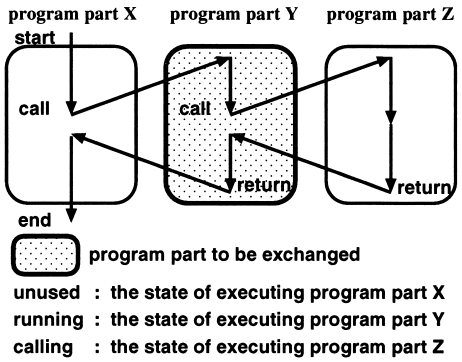

図1 プログラム部分の実行状態の関係
Fig. 1 Relation between states and program part to be exchanged.

表 1 プログラム部分状態と入替え条件の関係
Table 1 Relation between program part condition and restriction for exchange.

通番	実行状態	入替え条件
(1)	未使用	(条件 1)，(条件 2)
(2)	走行中	入替え不可能
(3)	呼出中	(条件 1)，(条件 2) (条件 3)，(条件 4) (条件 5)，(条件 6)
(4)	未使用かつ走行中	入替え不可能
(5)	未使用かつ呼出中	(条件 1)，(条件 2) (条件 3)，(条件 4) (条件 5)，(条件 6)
(6)	走行中かつ呼出中	入替え不可能
(7)	未使用かつ走行中 かつ呼出中	入替え不可能

(3) 呼出中：別のプログラム部分を呼び出している状態

3 状態のうち「走行中」の状態では，内部変数値などが過渡的な状態であるためプログラム部分を入れ替えることはできない．

以降では，各プログラム部分を「モジュール」と呼び，入替え対象となるプログラム部分を「入替え対象モジュール」と呼ぶこととする．

2.1.2 入替え条件

新たに入れ替えるモジュールを記述する際，入替え前後での整合性を保つためには，以下の条件を満たす必要がある．

（条件 1） 呼び出しと返却のインタフェースの一致
（条件 2） 処理の矛盾（ループカウンタのリセットなど）の回避
（条件 3） 戻り値のアドレスを変更しないこと
（条件 4） 静的リンクの場合，メモリ上の外部変数の参照アドレスが同じであること

また，変更する内容に応じてプログラム個別に以下の対処が必要である．

（条件 5） 外部変数の値が入替えの前後で同じであることの保証が必要か否か
（条件 6） アドレス渡しによる外部変数や内部変数の参照や更新がどのように行われているか

多くの場合，プログラムの一部分である入替え対象モジュールは，複数のプロセスによって共有されている．このような場合も含めたモジュールの実行状態と入替え条件の関係は，**表 1** のようになる．

入替え対象モジュールが複数のプロセスによって共有されている場合，そのモジュールの状態は，通番 (4) から (7) に示すように，同時に複数の状態になりうる．この場合入替え条件は，より厳しい条件に影響される．たとえば，通番 (5) の場合，入替え対象モジュールの実行状態が「未使用」のときの入替え条件は，(条件 1) と（条件 2）の 2 つである．これに対し，「呼出中」のときの入替え条件は（条件 1）から（条件 6）の 6 つである．このことから，「呼出中」のときの入替えを考慮することによって，通番 (5) の場合の入替え条件は，(条件 1）から（条件 6）の 6 つとなる．同様に，通番 (6) の場合は，入替え対象プログラム部分が「走行中」の場合の入替えを考慮することによって，入替え条件は入替え不可能となる．

2.1.3 有限時間内での入替えを保証する制御法

前述の入替え条件は，有限時間内での入替えの完了を保証してはいない．そこで，入替え要求が来ると有限時間内で入替えを実行し，完了できるようにする必要がある．入替え可能を保証することは，「入替え対象モジュールが有限時間内で走行中の状態でなくなること」を保証することと同値である．そこで，入替え要求があると，入替え対象モジュールに処理を移行しようとしているプロセスは，その処理を一時的に停止させ，その他のプロセスを走行させ続ける．その後，入替え対象モジュールが入替え可能状態であることを検知した直後に入替えを実施する．つまり，図 1 において，入替え要求後にプログラム部分 X からプログラム部分 Y，あるいは，プログラム部分 Z からプログラム部分 Y に処理を移行しようとしたプロセスについては，その処理を一時的に停止させる．その後，プログラム部分 Y を走行しているプロセスがなくなった直後に入替えを実行する．これにより，有限時間内での入替えを保証する．ただし，入替え対象モジュール内で停止しているプロセスが存在する場合，その停止中のプロセスが有限時間内にプログラム実行を再開する保証がないときは，この限りではない．

3. 基本方式の問題点と対処法

文献 12) では，入出力処理プログラムに対する入替え時間の評価を行っている．

そこで，本論文では，プロセッサ処理プログラムに対する入替え時間についての評価を行う．評価は文献 12) と同一条件下で行った．これは文献 12) の測定結果との比較が重要だからである．以降で具体的にその測定条件と評価について述べる．

3.1 測 定 条 件

測定に用いたテストプログラムの流れを**図 2** に示す．モジュール Y が入替え対象モジュールである．

入替え可能条件は，入替え対象モジュールであるモジュール Y の実行状態が「未使用」または「呼出中」

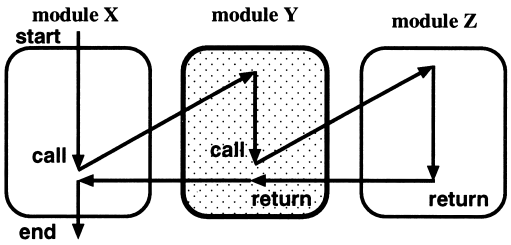

図 2 プログラムの処理の流れ
Fig. 2 Flow of test program.

表 2 プログラムの各モジュール処理時間（秒）
Table 2 Processing times of test program (sec.).

type	module X	module Y	module Z
A	1	1	8
B	1	2	7
C	1	3	6
D	1	4	5
E	1	5	4
F	1	6	3
G	1	7	2
H	1	8	1

の状態のときに入替え可能であるとする．

各モジュールの処理時間の関係を**表 2** に示す．各モジュールの処理時間の総和を 10 秒，モジュール X の処理時間を一定（1 秒）とした A から H の 8 つの場合を想定している．表 2 のような設定にしたのは，入替え対象モジュールであるモジュール Y の処理時間の違いによって入替え時間にどのような影響があるのかを明らかにするためである．各プロセスは擬似乱数を用いて不定間隔で起動し，モジュール Y を共有するプロセス数が 1 から 10 までの場合についてモジュール Y に対して入替えを行い，その入替え時間を測定している．

測定には，プロセッサ処理プログラムとして，表 2 で設定した処理時間だけ特定の変数を 1 加算する処理を行うプログラムを用意した．また，テストプログラムのどのモジュールも共有しないプロセス（以降「他プロセス」と呼ぶ）の存在が，入替え時間にどのような影響を与えるのかを明らかにするために，他プロセスが 1 つ同時走行している場合の入替え時間についても測定を行った．他プロセスは，プロセッサ処理プログラムと同様に，特定の変数を 1 加算する処理を繰り返すプログラムの実行プロセスである．測定は，20 回の平均値を測定結果として記録した．

なお，文献 12) では，入替え対象モジュールを含むプログラムの状態把握を，入替え要求直前に開始する場合と入替え対象プロセスの起動時に開始する場合について測定し，評価を行っている．しかし，入替え要求直前に入替え対象モジュールを含むプログラムの状態把握を開始する場合，プロセスの走行状況による測定結果への影響が大きい．そこで，本論文では，測定結果が示す原因や効果を分析しやすくするために，状態把握は入替え対象プロセスの起動時に開始するようにした．

測定評価は，OS として BSD/OS ver2.1 が動作している Pentium 90 MHz の計算機を用いて行った．また，入替え機能は，既存のカーネルプログラムは無変更，かつ依存性を最小限に抑える方針で実装されている．これにより，BSD/OS 以外の他システムへの移植に必要となる工数を少なくすることができる．

入替え機能は，6 つのシステムコールで構成される．詳細を以下に示す．

(1) **状態監視開始** 指定するプログラムを実行しているプロセスに対して，当該プロセスの状態把握の開始を要求する．
(2) **状態監視終了** 指定するプログラムを実行しているプロセスに対して，当該プロセスの状態把握の終了を要求する．
(3) **モジュール突入** プロセスがあるモジュールの実行を開始したことを告げる．
(4) **モジュール脱出** プロセスがあるモジュールの実行を終了したことを告げる．
(5) **入替え要求** 指定するプログラムの指定するモジュールの入替えを要求する．
(6) **タイムアウト時間設定** タイムアウト時間[12)]を設定する．

また，本実装は，複数プロセスがプログラムモジュールを共有し，同時に複数のモジュールに対して入替えの要求が生じる状況を想定している．これにより，すべての AP プロセスの走行環境に対応できる．

3.2 結果と考察

測定結果を**図 3**，**図 4**，**図 5** に示す．図 3 は，文献 12) の図 9 からの抜粋であり，入替え時間の目盛りを図 4 や図 5 と同一にしている．

各図において，共有プロセス数が 10 のときの各 type での入替え時間に着目する．図 3 の入出力処理プログラムに対する入替えでの入替え時間を 1 としたとき，図 4 のプロセッサ処理プログラムに対する入替えでは 6〜10 倍の入替え時間がかかっている．また図 5 から，他プロセスが 1 つ同時走行している場合には 7〜12 倍の入替え時間になり，いっそう遅くなることが分かる．さらに，プロセッサ処理プログラムに対する入替えの場合，共有プロセス数が多くなるほど入替え時

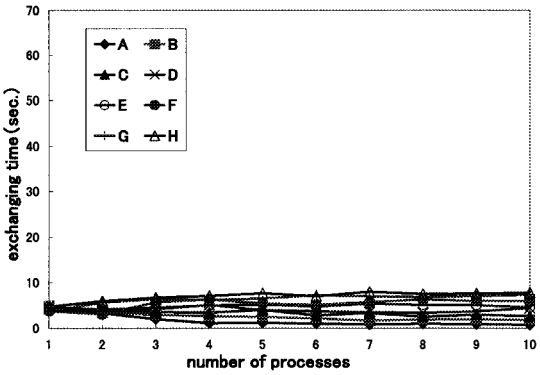

図 3 プロセス数と入替え時間の関係
（入出力処理プログラムに対する入替え）
Fig. 3 Number of processes vs. exchanging time (exchange of I/O processing program).

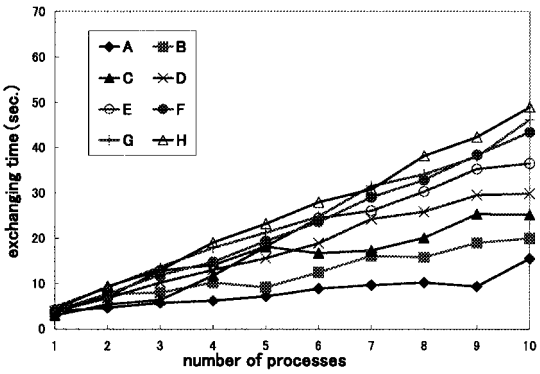

図 4 プロセス数と入替え時間の関係
（プロセッサ処理プログラムに対する入替え）
Fig. 4 Number of processes vs. exchanging time (exchange of processor processing program).

間が長くなる．

次に，他プロセスが入替え時間に与える影響に着目する．入替え対象モジュールの処理時間の総和 T_m の総処理時間に占める割合が，十分に高い場合の入替え時間について考察する．他プロセスが同時走行していない場合の入替え時間 T_o は，プログラムを共有しているプロセス数を n とすると，平均的には，

$$T_o = \left(T_m \times \frac{1}{2}\right) \times n \qquad (1)$$

となる．一方，各プロセスの READY 状態での待ち時間 T_w は，タイムスライス時間を T_t とすると，平均的には，

$$T_w = T_t \times \left(n \times \frac{1}{2}\right) \qquad (2)$$

となる．他プロセスが走行する場合，T_w ごとに他プロセスの走行（時間：T_t）が発生する．したがって，

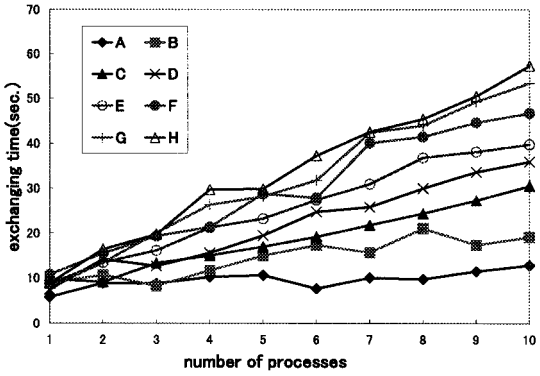

図 5 プロセス数と入替え時間の関係
（プロセッサ処理プログラムに対する入替え，他プロセスあり）
Fig. 5 Number of processes vs. exchanging time (exchange of processor processing program, another process exist).

他プロセスが入替え時間に与える影響時間 T は，

$$T = \frac{T_o}{T_w} \times T_t = \frac{\left(T_m \times \frac{1}{2}\right) \times n}{T_t \times \left(n \times \frac{1}{2}\right)} \times T_t = T_m \quad (3)$$

となる．

以上のことから，プロセッサ処理プログラムに対する入替えや，他プロセスが同時走行している状況下での入替えでは，入替え時間が大幅に増加してしまうという問題点がある．

プロセッサ処理プログラムに対する入替えを行う場合，入替え対象モジュールを共有する各プロセスは，共有プロセス数が多くなるほど，プロセススケジューリングにより，Ready 状態での待ちが発生する．プロセッサ処理プログラムに対する入替えに要する入替え時間が，入出力処理プログラムに対する入替えに要する入替え時間に比べて，大きく入替え時間が増加する原因は，Ready 状態での待ち時間による影響である．

3.3 対 処 法

入替え時間の短縮化に際しては，プロセッサ処理プログラムに対する入替えで要する入替え時間を，入出力処理プログラムに対する入替えで要する入替え時間に近づけることが目標となる．これは，入替え対象モジュールを共有する各プロセスの Ready 状態での待ち時間を短縮することによって実現できる．また，入替え時間を短縮化する対処法は，他プロセスの有無に影響を受けることなく効果があることが望ましい．

入替え時間を短縮するには，入替え対象モジュールが入替え可能でない状態を早期に脱出することが必要である．このため，入替え対象モジュールが入替え可能状態でない場合には，入替え対象モジュールを走行

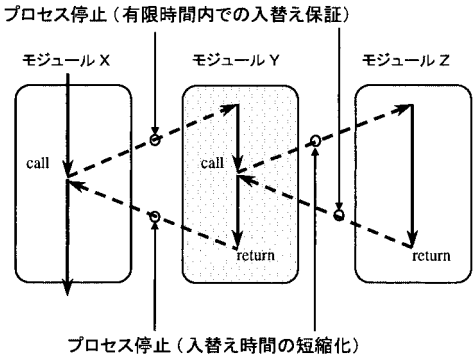

図6 プロセス早期停止の様子
Fig. 6 The situation of process suspension.

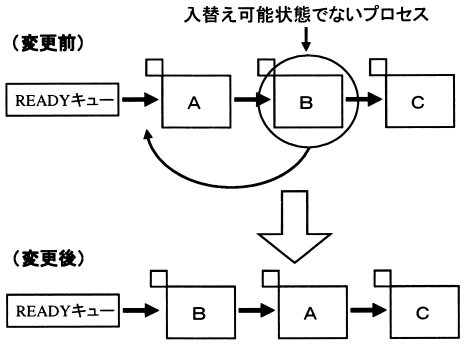

図7 ready キューのつなぎ変えの様子
Fig. 7 Exchange of a connection of ready queue.

しているプロセス，すなわち，入替え可能状態でないプロセスのみを走行させることによって，早期に入替え可能状態に遷移することができる．そこで，以下の2つ対処法を示す．

(1) **入替え可能状態にあるプロセスの早期停止** これは，入替え要求時に，すでに入替え対象モジュール以外のモジュールを走行中のプロセス，すなわち入替え可能状態にあるプロセスに着目した対処法である．具体的には，入替え可能状態にあるプロセスの処理を，一時的に強制停止する．これにより，入替え可能状態でないプロセスが走行する比率が増加し，入替え可能状態への遷移が従来よりも早くなると推察する．

図6 にプロセスの早期停止の様子を示す．入替え対象モジュールはモジュールYとする．ここで，入替え可能条件は，モジュールYの実行状態が「未使用」または「呼出中」の状態のときに入替え可能であるとしている．このとき，各プロセスにおいて，入替え可能でない状態から入替え可能な状態に遷移した直後に，その処理を一時的に停止させる．すなわち，図6において，モジュールYからモジュールZ，あるいはモジュールYからモジュールXに処理を移行しようとしたときがこれにあたる．なお，図6では，2.1.4項「有限時間内での入替えを保証する制御法」によるプロセスの一時停止の様子についても，あわせて示している．

(2) **入替え可能状態でないプロセスの優先実行** 入替え可能状態でないプロセスの優先実行とは，入替え可能状態でないプロセスを優先的に走行させることである．これにより，早期に入替え可能な状態に移行できる．図6において，モジュールYを実行していることが検出されたプロセスが処理の対象となる．具体的には，入替え可能状態でないことが検出できた時点で，当該プロセスがつながっているreadyキューについて，当該プロセスをreadyキューの先頭につなぎ変えることで優先的に走行させるようにする．**図7** にreadyキューへのつなぎ変えの様子を示す．入替え要求後，入替え対象モジュールを走行中のプロセス，すなわち入替え可能状態でないプロセスが存在するか否かを確認する．存在しない場合はただちに入替えが行われる．存在する場合には，そのプロセスについて，その時点でのreadyキューへのつながりをキューの先頭に変更する．これにより，対象プロセスの優先実行を実現する．

上述の対処法では，プロセスの処理の流れを制御するため，以下の2点について注意が必要である．

(1) 入替え可能状態にあるプロセスの早期停止は，デッドロックを引き起こす可能性がある．具体的には，入替え対象モジュールにイベント待ちをともなう処理が含まれ，入替え可能状態にあるプロセスが，そのイベントの駆動契機を与えるような場合には，デッドロックが生じる可能性がある．

(2) 入替え可能状態でないプロセスの優先実行は，プロセス実行の優先度逆転を起こす可能性がある．このため，1つのサービスが複数プロセスによって構成され，各プロセスの優先度によって実行順序を決めているような場合には，不具合が生じる可能性がある．

4. 評価と考察

本章では，提案した入替え時間を短縮する手法を適用した場合について，測定した入替え時間の結果を示

し，提案した手法の有効性について考察する．

4.1 測定条件

提案した対処法の効果を明らかにするために，測定は前章で述べた条件とまったく同じ条件下で行った．

前述の評価では，入出力処理プログラムとプロセッサ処理プログラムに対する入替え時間を評価した．一般的に，既存ソフトウェアは，入出力処理とプロセッサ処理の両方をともなう．したがって，既存ソフトウェアに本入替え法を適用した場合，入替え時間は，入出力処理プログラムに対する入替え時間とプロセッサ処理プログラムに対する入替え時間の間の値になると推察できる．この推察を裏付けるため，既存ソフトウェアに本入替え法を適用した場合の評価として，擬似プログラムを作成し，その擬似プログラムに対する入替え時間を測定した．具体的には，プロセッサ処理と入出力処理の処理時間比が，BSD-UNIX の sort コマンドと同等の擬似プログラムにおける入替え時間を測定した．sort の処理時間比は，ライブラリ rand() で生成した正の整数 100000 個からなるファイル（整数の区切りは改行）を 2 種類用意し，“sort file1 file2 -o outfile”の処理時間を測定することにより算出した．結果はほぼ CPU : I/O = 1 : 1 であった．そこで，各モジュールの処理時間が CPU : I/O = 1 : 1 である擬似プログラムを作成し，それに対する入替え時間を測定し評価を行った．

4.2 プロセス早期停止の効果

図 8，**図 9** に，入替え可能状態であるプロセスの早期停止による対処を行った場合の入替え時間の測定結果を示す．それぞれ，図 8 は他プロセスが存在しない場合，図 9 は他プロセスが 1 つ同時走行している場合の入替え時間の測定結果である．各図では，各 type での特徴がよく分かるように，type A，E，H の場合の測定結果のみを実線で示し，本手法を適用しない場合の測定結果である図 4，図 5 の type A，E，H を type A’，E’，H’ として破線で示す．

図 8，図 9 のそれぞれの測定結果から分かる特徴について，以下に type 別に述べる．

(1) type A の場合，本手法による効果があまり見られない．この場合，総処理時間 10 秒に対して，入替え対象モジュールである module Y の処理時間は，1 秒と短い．このため，入替え要求時に，各プロセスがすでに入替え可能状態である確率が非常に高い．本手法は，入替え要求後，入替え対象モジュールから他のモジュールに処理を移行しようとしたプロセスのみが処理の対象である．つまり，すでに入替え可能状態であるプロセスについては処理の対象とはならない．したがって，type A のように，入替え対象モジュールの処理時間が総処理時間に対して非常に短い場合は，本手法の効果は現れにくくなる．

(2) type E の場合，本手法の効果が大きく，最大 25%，平均でおよそ 20%程度入替え時間を短縮することができた．この場合，module Y の処理時間とその他のモジュールの処理時間の総和は等しい．このため，入替え要求時に，入替え可能状態でないプロセスが存在する確率は，type A の場合に比べて高い．このことから，入替え可能状態でないプロセスの数が多いほど本手法の効果は大きくなり，入替え時間の短縮につながることが分かる．

(3) type H の場合，type E と同様に本手法の効果が見られる．しかし，平均でおよそ 12%程度の入替

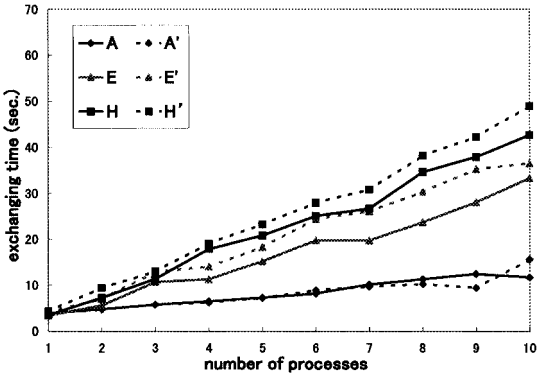

図 8 プロセス数と入替え時間の関係
（プロセッサ処理プログラムに対する入替え）
Fig. 8 Number of processes vs. exchanging time (exchange of processor processing program).

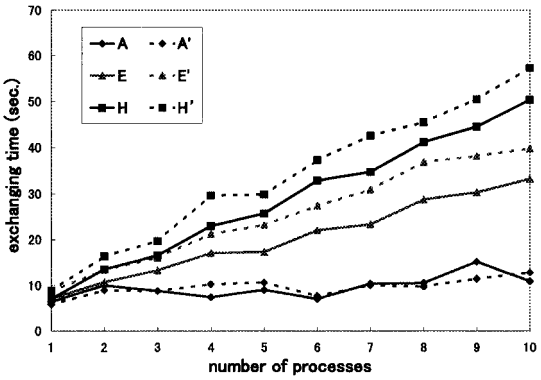

図 9 プロセス数と入替え時間の関係
（プロセッサ処理プログラムに対する入替え，他プロセスあり）
Fig. 9 Number of processes vs. exchanging time (exchange of processor processing program, another process exist).

え時間の短縮であり，type E の場合ほどの効果は得られていない．この場合，総処理時間 10 秒に対して，module Y の処理時間は 8 秒と長い．したがって，type E の場合よりも，入替え要求時に入替え可能状態でないプロセスが存在する確率は高い．また，入替え可能状態でないプロセスの数も多いと考えられる．しかし，この場合，前述のとおり module Y の処理時間が大半を占めるため，本手法を適用しない場合の入替えにおいて，入替え対象モジュール以外を走行しているプロセスが入替え時間に与える影響は，それほど大きくない．したがって，本手法を適用しても，type E の場合ほどの効果は出ないといえる．

また，図 8，図 9 を比較することで，他プロセスが入替え時間に与える影響について，以下の特徴があることが分かる．

(4) 図 8，図 9 において，同じ type における入替え時間の短縮率は，ほぼ同程度である．つまり，他プロセスの有無によって，本手法の効果に違いが生じることはないといえる．これは，本手法の適用対象が，入替え対象モジュールを共有するプロセスであり，他プロセスは適用対象外であるためと推察する．

以上のことから，本手法は，入替え対象モジュールを含むプログラムの総処理時間に対して，入替え対象モジュールの処理時間の割合が 50%程度であるときに最も効果があるといえる．評価では，最大で 25%，平均で約 20%処理時間を短縮することができた．これは，入出力処理プログラムに対する入替えに要する入替え時間に比べて増加した処理時間について，最大で約 36%の短縮にあたる．また，他プロセスの有無によって，入替え時間の短縮率に大きな違いはなく，本手法の適用により，他プロセスに影響を受けることなく入替え時間の短縮化を図ることができる．ただし，残念ながら，本手法では，他プロセスが入替え時間に与える影響を取り除くことはできないといえる．

4.3 プロセス優先実行の効果

図 10，**図 11** に，入替え可能状態でないプロセスの優先実行による対処を行った場合の入替え時間の測定結果を示す．それぞれ，図 10 は他プロセスが存在しない場合，図 11 は他プロセスが 1 つ同時走行している場合の入替え時間の測定結果である．また，各図では，各 type での特徴がよく分かるように，type A，E，H の場合の測定結果のみを実線で示し，本手法を適用しない場合の測定結果である図 4，図 5 の type A，E，H を type A’，E’，H’ として破線で示す．

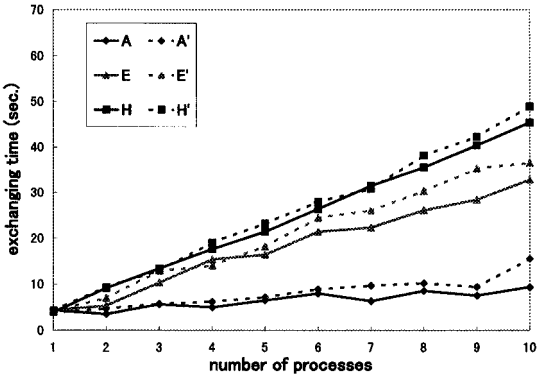

図 10 プロセス数と入替え時間の関係
（プロセッサ処理プログラムに対する入替え）
Fig. 10 Number of processes vs. exchanging time (exchange of processor processing program).

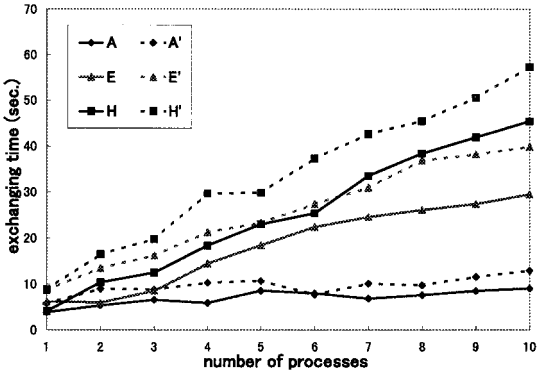

図 11 プロセス数と入替え時間の関係
（プロセッサ処理プログラムに対する入替え，他プロセスあり）
Fig. 11 Number of processes vs. exchanging time (exchange of processor processing program, another process exist).

図 10 の測定結果から分かる特徴について，type 別に以下に述べる．

(1) type A の場合，プロセス数が多くなると，本手法の効果が大きくなっている．入替え時間を最大で 40%程度短縮している場合もある．この場合，総処理時間 10 秒に対して，入替え対象モジュールである module Y の処理時間が 1 秒と非常に短い．このため，入替え要求時に，入替え可能状態でないプロセスの存在確率は低い．これは，共有プロセスに占める入替え可能状態でないプロセスの割合が非常に低いことを意味する．このため，本手法を適用することによって，少ない入替え可能状態でないプロセスを優先的に実行することができ，その効果は，プロセス数の増加にともなって大きくなると推察する．

(2) type E の場合，本手法の効果が見られる．本手法を適用しない場合に比べて，最大 20%程度の時間短縮となっている．この場合，module Y の処理時間とその他のモジュールの処理時間の総和は等しい．したがって，入替え要求時に入替え可能状態であるプロセスと，そうでないプロセスの割合もほぼ同等であると推察する．このため，type A に比べ，本手法の対象プロセスの数は多い．したがって，本手法の効果は小さくなると推察する．

(3) type H の場合も同様に本手法の効果が見られる．本手法を適用しない場合に比べて平均で約 12%程度の時間短縮となっている．ただしこの場合，総処理時間 10 秒に対して，module Y の処理時間は 8 秒と大半を占める．これは，入替え要求時に，共有プロセスに占める入替え可能状態でないプロセスの割合が非常に高いことを意味する．したがって，type E 以上に本手法の対象プロセスの数は多くなり，効果が小さくなると推察する．

次に，図 10，図 11 のそれぞれの type H，H' について着目することで，他プロセスが入替え時間に与える影響について，以下の特徴があることが分かる．

(4) 図 10 と図 11 を比較したとき，図 11 の方が，入替え時間の短縮率が 10〜30%程度高い．このことから，他プロセスが存在する場合入替え時間の短縮率が高くなるといえる．これは，本手法の適用によって，他プロセスが入替え時間に与える影響を取り除くことができるからであると推察する．なぜなら，本手法の適用対象は，入替え対象モジュールを共有するプロセスのうち，入替え要求時に入替え可能状態でないプロセスであり，他プロセスは適用対象外である．このため，適用対象プロセスが本手法によって優先的に実行されている間は，他プロセスの処理が実行されることはなく，他プロセスが入替え時間に影響を及ぼすことはない．したがって，他プロセスが同時走行している状況において，本手法を適用することは，効果的であるといえる．しかしながら一方で，入替えとは無関係であるプロセスの処理が一時的に停止してしまうという問題点もある．

以上のことから，本手法は，入替え対象モジュールを含むプログラムの総処理時間に対する入替え対象モジュールの処理時間の割合が低く，かつ共有プロセス数が多い場合に最も効果があるといえる．評価では，最大 40%程度入替え時間を短縮することができた．これは，入出力処理プログラムに対する入替えに要する入替え時間に比べて増加した処理時間について，最大で約 42%の短縮にあたる．また，他プロセスが同時走行しているような状況下での入替えでは，本手法を適用することで，他プロセスが存在しない場合と同等の入替え時間で処理を行うことができ，他プロセスが入替え時間に与える影響を取り除くことができるといえる．しかし一方で，入替えとは無関係であるプロセスの処理が一時的に停止してしまうという問題点も明らかにした．

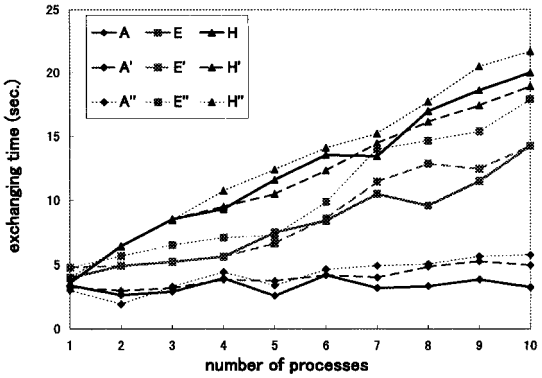

図 12 プロセス数と入替え時間の関係
（sort シミュレーションプログラムに対する入替え）
Fig. 12 Number of processes vs. exchanging time (exchange of sort simulation program).

4.4 既存ソフトウェア擬似プログラムの評価

測定結果を**図 12** に示す．type A，E，H はプロセスの優先実行を適用した場合の入替え時間，type A'，E'，H' はプロセスの早期停止を適用した場合の入替え時間，type A''，E''，H'' はどちらの対処法も適用しなかった場合の入替え時間をそれぞれ指している．

図 12，図 8，図 9 から各測定結果は，プロセッサ処理プログラムに対する入替え時間ののほぼ半分になっており，推察を裏付ける結果となっている．また，プロセスの早期停止，プロセスの優先実行を適用した場合の効果も，4.2 節「プロセス早期停止の効果」，4.3 節「プロセス優先実行の効果」で述べた特徴と同様の特徴を見ることができる．入替え時間は，総処理時間に対して 2 倍程度であり，大きな影響を与えていないといえ，既存システムに対しても十分に適用することができるという結論を得ることができた．

5. あとがき

実行中プログラムの部分入替え法について，入替えを行うプログラムの処理の主体が，入出力処理であるかプロセッサ処理であるかの違いによって，入替え時間が大きく異なる問題を明らかにした．また，他プロセスが同時走行している場合は，入替え時間がさらに

増加することを示し，定式化を行った．

そして，この問題への対処として，入替え時間を短縮する手法を提案した．具体的には，入替え可能状態にあるプロセスを早期停止する手法と，入替え可能状態でないプロセスを優先実行する手法を提案した．

また，実装および測定を行い，次のことを明らかにした．まず，入替え可能状態にあるプロセスを早期停止する手法では，入替え対象モジュールを含むプログラムの総処理時間に対して，入替え対象モジュールの処理時間が 50%程度であるとき，最も効果がある．評価では，最大 25%程度入替え時間を短縮することができた．これは，プロセッサ処理プログラムに対する入替えにおいて，入出力処理プログラムに対する入替えに要する入替え時間に比べて増加した処理時間の約 36%にあたる時間の短縮である．次に，入替え可能状態でないプロセスを優先実行する手法では，入替え対象モジュールを含むプログラムの総処理時間に対して，入替え対象モジュールの処理時間の割合が小さく，共有プロセス数が多いときに最も効果がある．評価では，最大 40%程度入替え時間を短縮することができた．これは，プロセッサ処理プログラムに対する入替えにおいて，入出力処理プログラムに対する入替えに要する入替え時間に比べて増加した処理時間の約 42%にあたる時間の短縮である．また，両手法は，他プロセスの影響を受けることなく入替え時間の短縮を図ることができる．特に，入替え可能状態でないプロセスを優先実行する手法は，他プロセスが入替え時間に与える影響も取り除くことができる．

さらに既存ソフトウェアへの適応に関する評価として，プロセッサ処理と入出力処理の処理時間比が，BSD-UNIX の sort コマンドと同等の疑似プログラムに対する入替え時間の評価を行った．結果として，入替え時間は，せいぜい総処理時間の 2 倍程度であるという結果が得られた．

今後の課題として，提案した入替え時間を短縮する手法が，同時走行中の他プロセスの処理に与える影響を明らかにすることがある．また，本論文で提案した手法により，増加した処理時間に対して最大で 42%短縮することができたが，より短縮することが可能な方法を検討する．

参 考 文 献

1) Muramatsu, H., Date, M., Yoshida, H., Kitaoka, M. and Kurobane, N.: Operating System SXO for Continuous Operation, *IFIP 92*, Vol.1, pp.615–621 (1992).
2) Snead, B., Ho, F. and Engram, B.: Operating System Features Real Time and Fault Tolerance, *Computer Design*, pp.177–185 (Aug. 1984).
3) Pardyak, P. and Bershad, B.N.: Dynamic Binding for an Extensible System, *Proc. 2nd USENIX Symposium on Operating Systems Design and Implementation* (*OSDI*), pp.201–212 (1996).
4) Condict, M., Michell, D. and Reynolds, F.: *Optimizing Performance of Mach-based System By server Co-location, A Detailed Design*, OSF Research Institute One Cambridge Center (1993).
5) 中島達夫：マイクロカーネルアーキテクチャに関する考察，日本ソフトウェア科学会第 14 回大会論文集，pp.497–500 (1997).
6) Tokuda, H., Nakajima, T. and Rao, P.: RT-Mach: Towards a Predictable Real-Time System., *Proc. USENIX Mach Workshop* (1990).
7) Oikawa, S., Sugirura, K. and Tokuda, H.: Adaptive Object Management for a Reconfigurable Microkernel., *Proc. International Workshop on Object Orientation in Operating Systems* (1996).
8) Moriai, S. and Tokuda, H.: Dynamic Loadable Object Support for Real-Time Mach Kernels., *Proc. International Conference on Worldwide Computing & Its Applications '97* (1997).
9) 柏木一彦，最所圭三，福田 晃：動的構築機能を有するオペレーティングシステム・サーバについて，日本ソフトウェア科学会第 16 回大会論文集，pp.97–100 (1999).
10) 林 幸弘，柴山悦哉：コンパイルタイムリフレクションによる OS 拡張の記述，日本ソフトウェア科学会・コンピュータシステム，Vol.16, No.5, pp.72–77 (1999).
11) 谷口秀夫，伊藤健一，牛島和夫：プロセス走行時におけるプログラムの部分入替え法，信学論（D-I），Vol.J78-D-I, No.5, pp.492–499 (1995).
12) 谷口秀夫，後藤真孝：走行中のプロセス間で共有されたプログラムの部分入替え法，信学論（D-I），Vol.J80-D-I, No.6, pp.495–504 (1997).
13) 谷口秀夫，後藤真孝：実行中プログラムの部分入替え法における入替え時間の評価，信学論（D-I），Vol.J82-D-I, No.8, pp.998–1007 (1999).

(平成 11 年 12 月 13 日受付)
(平成 12 年 4 月 6 日採録)

中島　雷太（正会員）

平成 10 年九州大学工学部情報工学科卒業．平成 12 年九州大学大学院システム情報科学研究科修士課程修了．同年ソニー（株）入社．オペレーティングシステムに興味を持つ．

谷口　秀夫（正会員）

昭和 53 年九州大学工学部電子工学科卒業．昭和 55 年同大大学院修士課程修了．同年日本電信電話公社電気通信研究所入所．昭和 62 年同所主任研究員．昭和 63 年 NTT データ通信（株）開発本部移籍．平成 4 年同本部主幹技師．平成 5 年九州大学工学部助教授．平成 8 年九州大学大学院システム情報科学研究科助教授．博士（工学）．オペレーティングシステム，分散処理に興味を持つ．著書「オペレーティングシステム」（昭晃堂）．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員．