システムソフトウェアと
オペレーティング・システム
88－3
（２００１．７．２６）

実行中プログラム部分入替え法における入替え時間の短縮法

山本 淳† 谷口 秀夫‡

†九州大学大学院システム情報科学府 ‡九州大学大学院システム情報科学研究院

〒 812-8581 福岡県福岡市東区箱崎 6-10-1
TEL 092(642)3867
Email：†yamamoto@swlab.csce.kyushu-u.ac.jp ‡tani@csce.kyushu-u.ac.jp

あらまし

動作中のソフトウェアを変更できれば，共同利用計算機の無停止運転や個人利用計算機の利便性の向上が可能になる．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，入替え時間の観点から評価を行い，入替え条件を緩和した場合，入替え時間が長大化することを明らかにした．本論文では，入替え時間が長大化する問題への対処として，入替え時間を短縮する方法について述べる．具体的には，入替え可能状態にあるプロセスを早期に停止させる．また，入替え不可状態にあるプロセスを優先的に実行させる．さらに，両者の方法について実装および評価を行い，その方法の有効性を示す．

キーワード

プログラム入替え，プロセス，プロセススケジューリング

Methods of Reducing the Exchanging Time on Mechanism for Exchanging Parts of Executed Program

Atsushi YAMAMOTO† and Hideo TANIGUCHI‡

Graduate School of Information Science and Electrical Engineering, Kyushu University

〒 812-8581 6-10-1 Hakozaki, Higashi-ku, Fukuoka, Japan
TEL 092(642)3867
Email：†yamamoto@swlab.csce.kyushu-u.ac.jp ‡tani@csce.kyushu-u.ac.jp

abstract

We have proposed the mechanism for exchanging parts of executed program. We have described the evaluation of exchanging time on some situation. And we have described that the exchanging time has increased under the relaxed conditions of exchange. In this paper, we describe two methods of reducing the exchanging 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 are unable to exchange gives priority. We also report the evaluation of the exchanging time and the availability of the two methods.

keywords

exchanging program, process, process scheduling

1　まえがき

計算機の高性能化と低価格化により，計算機の普及が進んでいる．計算機が提供するサービスは，ハードウェアのみで提供されることは少なく，サービスの提供には，ソフトウェアが深く関与している．一方，計算機の利用形態の多様化に伴い，不具合の改修，新しい機能の追加，および携帯端末における使用環境の変化に合わせた機能縮小のために，サービス機能の追加や変更および削除が頻繁に必要になっている．多くの場合，ソフトウェアの変更は計算機の停止を必要とする．しかし，共同利用している計算機では無停止運転が望まれており，個人利用の計算機でも利便性向上のためには，ソフトウェア変更による計算機の停止は避けたい．そこで，動作中のソフトウェアを変更できれば利便性が向上し，さらに，ソフトウェアの一部分を変更できれば，変更時間は僅かになり，サービスに与える影響を抑制できる．

プロセスとして走行しているプログラムを変更する方法については，プロセスを冗長構成にしてプロセス単位で入れ替える方法 [1] がある．この方法では，プロセス単位で入れ替えるため，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る．一方，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] も報告されている．しかし，この方法は，入替えを行うサービスプログラムが，入替えの契機を考慮し，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，入替え契機を意識したサービスプログラムの作成を必要としている．さらに，マイクロカーネルに基づいた動的機構を有する OS に関する研究 [3] がある．しかし，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，OS 自体の性能が劣る．また，その動的再構築や機能拡張は，OS を対象にしたものであり，アプリケーションを対象にしてない．

我々は，プロセスとして走行しているプログラムの一部分を変更する方法として，次の大きな 2 つの特徴を持つ方法を提案した [4]．1 つは，サービスプログラムが入替えの契機を意識する必要がない点である．もう 1 つは，入替え対象の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も考慮している点である．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法を示し [5]，入替え要求から入替え処理を終了するまでの時間 (以降，入替え時間と呼ぶ) の定式化を行った [6]．しかし，いずれの場合も，プロセスが走行しているプログラム部分の実行状態が走行中の時のみを入替え不可としており，実行状態が未使用または呼出中の時には入替え可能としている．このため，入れ替えるプログラム部分が満たすべき入替え条件は 6 つあり，入れ替えるプログラム部分の作成が難しいといえる．

そこで，本論文では，プロセスが走行しているプログラム部分の実行状態が走行中または呼出中の時は入替え不可とし，実行状態が未使用の時のみを入替え可能とする方法を説明する．これにより，入れ替えるプログラム部分が満たすべき入替え条件は 2 つになり，入替え条件を緩和できる．しかし，この入替え条件の緩和により，入替え時間が長大化する．そこで，入替え時間の長大化を抑制する方法を示し，実装と評価により，その方法の有効性を示す．

2　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 つの課題を挙げその対処法を示した．

(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) 入替えの制御法
(5) サービスへの影響の軽減法

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　プログラム実行状態の分類

図 1 に示すように，プログラム部分の実行状態は 3 つの状態に分類できる．すなわち，入替え対象のプログラム部分に対して，以下の 3 状態である．

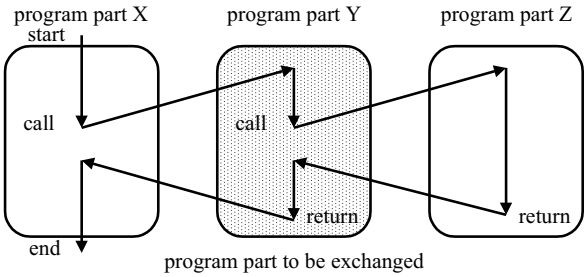

図 1 プログラム部分の実行状態の関係

(1) 未使用：実行する前，または，実行を終えた状態
(2) 走行中：実行中の状態
(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，走行中の状態は，内部変数値が過渡的であるため，入替えは不可能である．

以降では，各プログラム部分をモジュールと呼び，入替え対象のプログラム部分を入替え対象モジュールと呼ぶ．

2.2 入替え条件

入替え前後での整合性を保つため，入れ替えるモジュールは以下の入替え条件を満たす必要がある．

(条件 1) 呼出しと返却値のインタフェースの一致
(条件 2) 処理の矛盾の回避
(条件 3) 戻り先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，入替えの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プログラム実行状態と入替え条件の関係を表 1 に示す．

表 1 プログラム実行状態と入替え条件の関係

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
(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
(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
(7)	未使用かつ走行中かつ呼出中	入替え不可

2.3 入替えの制御法

モジュールを入れ替える処理は，以下の 3 つの部分からなる．

(1) 入替え可能状態への移行処理
(2) 入替え可能状態の保持処理
(3) 入替え処理

入替え可能状態への移行処理とは，入替え不可状態にある各プロセスを入替え可能状態へ移行させる処理である．この処理に関しては，特に制御は行ってない．入替え可能状態の保持処理とは，入替え可能状態にある各プロセスの状態を保持させる処理である．すなわち，入替え要求時に既に入替え可能状態にあるプロセス，または，入替え可能状態へ移行したプロセスの状態を保持させる．具体的には，入替え不可状態へ遷移しようとするプロセスの走行を遷移の直前で停止させる．

入替え可能状態の保持処理を行うことによって，入替え不可状態であるモジュールを実行中のプロセスは存在しなくなる．すなわち，有限時間内で入替え対象モジュールの実行状態を入替え可能状態へ遷移させることができる．ただし，入替え不可状態であるモジュール内で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時は例外的な場合であり，入替えを行うことはできない．

入替え処理は，入替え対象モジュールの実行状態が入替え可能状態へ遷移した後，モジュールの内容をメモリ上で上書きする処理である．

表 2　入替え可能状態の違いによる相対的特徴

入替え可能状態	入替え条件	入替え時間
未使用 or 呼出中	多い (6 条件)	短い
未使用	少ない (2 条件)	長い

3　入替え条件の緩和

3.1　基本方式の問題点

2.2 節で述べたように，入替え前後での整合性を保つために入替えの内容が満たすべき入替え条件は，プログラム実行状態に依存する．表 1 に示したように，入替え可能状態を未使用または呼出中の状態とする場合，(条件 1) から (条件 6) の 6 条件を考慮して，入れ替えるモジュールを記述しなければならない．これは，本入替え法の利用者にとって大きな制約である．これに対して，入替え可能状態を未使用の状態のみとした場合，入替え条件を (条件 1) と (条件 2) の 2 条件に緩和でき，本入替え法の利便性が向上する．しかし，この場合には入替え時間が長大化する．なぜなら，入替え不可状態であるモジュール処理時間の総和が長くなるためである．表 2 に，入替え可能状態の違いによる入替え条件と入替え時間の相対的な特徴を示す．特に，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) の場合，その入替え時間は，入替え不可状態であるモジュール処理時間の総和と共有プロセス数の積の値に影響されるため，この入替え条件の緩和により，入替え時間が著しく長大化する [7]．さらに，入替え対象モジュールを共有してないプロセス (以降，他プロセスと呼ぶ) による入替え時間への影響時間も長大化する [8]．したがって，入替え条件の緩和による入替え時間の長大化を抑制する必要がある．

3.2　対処法

3.2.1　基本的な考え方

入替え条件の緩和による入替え時間の長大化の抑制は，2.3 節で述べた入替え可能状態への移行処理において，入替え不可状態にあるプロセスの走行確率を増加させるようにプロセスの走行を制御することで実現できる．図 2 に，入替えの可否を基準としたプロセスの分類を示す．入替え不可状態にあるプロセスの走行確率を増加させることによって，入替え不可状態にある全プロセスが早期に入替え可能状態へ移行できる．すなわち，入替え可能状態への移行処理を早期に終了でき，入替え時間を短縮できる．したがって，以下の 2 つの対処法が有効である．ここでは，文献 [9] に示した 2 つの対処法の改善を図る．

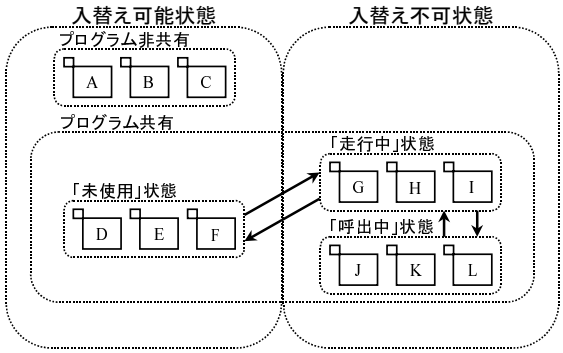

図 2　入替え可否を基準としたプロセス分類

3.2.2　プロセス早期停止法

プロセス早期停止法とは，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を早期に停止させる．プロセス早期停止法における文献 [9] の制御 (以降，(制御 1) とする) は，モジュール間移動を契機として，入替え不可状態から入替え可能状態へ遷移したプロセスを停止させるものである．このため，入替え要求時に既に入替え可能状態にあるプロセスについては，入替え可能状態の保持処理により停止させられるまでの間，その走行が許されている．

そこで，入替え要求時に既に入替え可能状態にあるプロセスについては，入替え要求を契機として，即座にその走行を停止させる (以降，(制御 2) とする) 方式を提案する．一方，入替え可能状態にあるプロセスには，図 2 に示したように，入替え対象のプログラムを共有しているプロセスと共有してないプロセスがある．このうち，後者については，入替えとは全く無関係なプロセスである．そこで，(制御 2) の制御対象を，前者のみとした制御を (制御 2-1) とする．また，両者ともに制御対象とした制御を (制御 2-2) とする．表 3 に，プロセス早期停止法の制御方式を示す．

表 3 プロセス早期停止法の制御方式

制御方式	制御内容
(早期停止 1)	(制御 1)
(早期停止 2)	(制御 1)+(制御 2-1)
(早期停止 3)	(制御 1)+(制御 2-2)

3.2.3 プロセス優先実行法

プロセス優先実行法とは，入替え不可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，入替え不可状態にあるプロセスを優先的に実行させる．プロセス優先実行法における文献 [9] の制御 (以降，(制御 A) とする) は，モジュール間移動を契機とする入替え可否の判別処理において，入替え不可状態にあると検出されたプロセスを当該 READY キューの先頭につなぎ変えるものである．このため，入替え要求後にいずれかのプロセスがモジュールに突入または脱出するまでの間，入替え不可状態にあるプロセスが走行する保証はない．そこで，入替え要求を契機として，入替え不可状態にある全プロセスに対して，(制御 A) における READY キューのつなぎ変え処理を行う (以降，(制御 B) とする) 方式を提案する．

さらに，次の工夫も行う．入替え不可状態であるモジュールに WAIT 状態になる処理が含まれる場合，入替え不可状態にあるプロセスの中でも，入替え可能状態への移行時間が長いプロセスほど，より優先的に走行させた方が効率が良い．そこで，READY キューのつなぎ変え処理において，入替え不可状態にあるプロセスを当該 READY キューの先頭につなぎ変える (以降，(処理 1) とする) のではなく，当該プロセスよりも入替え可能状態への移行時間が短いプロセスの直前につなぎ変える (以降，(処理 2) とする) ようにする．表 4 に，プロセス優先実行法の制御方式を示す．

4 評価と考察

文献 [9] では，入替え可能状態を未使用または呼出中の状態とする場合について，(早期停止 1) および (優先実行 1) を適用することによって，入替え時間の長大化を抑制できることを示した．そこで，ここでは，入替え条件を緩和するため，入替え可能状態を未使用の状態のみとする場合について，2 つの対処法において提案した新方式を適用した場合の測定結果を示し，旧方式との比較評価の観点から，提案した新方式の有効性について考察する．

表 4 プロセス優先実行法の制御方式

制御方式	制御内容	キュー処理
(優先実行 1)	(制御 A)	(処理 1)
(優先実行 2)	(制御 A)+(制御 B)	(処理 1)
(優先実行 3)	(制御 A)+(制御 B)	(処理 2)

4.1 測定条件

測定に用いたテストプログラムの処理の流れを図 3 に示す．module Y が入替え対象モジュールである．また，入替え可能状態は未使用の状態のみとする．

各モジュール処理時間を表 5 に示す．各モジュール処理時間の総和を 10 秒，module Y の処理時間を一定 (1 秒) とし，typeA から typeH の 8 つの場合を想定した．表 5 のような設定にしたのは，入替え可能状態である module X の処理時間と入替え不可状態である module Y と module Z の処理時間の総和の違いによって，それぞれ，プロセス早期停止法とプロセス優先実行法の新方式の効果がどのように異なるかを明らかにするためである．

テストプログラムには，入替え可能状態を未使用の状態のみとすることによって，入替え時間の長大化が起こり易いプロセッサ処理プログラムを用意した．プロセッサ処理プログラムは，各モジュールにおいて，表 5 で設定した処理時間の間，特定の変数を 1 加算する処理を繰り返すプログラムである．一方，プロセッサ処理プログラムには，WAIT 状態になる処理が含まれないため，プロセス優先実行法の新方式 (優先実行 3) の有効性を評価できない．そこで，テストプログラムとして，各モジュール

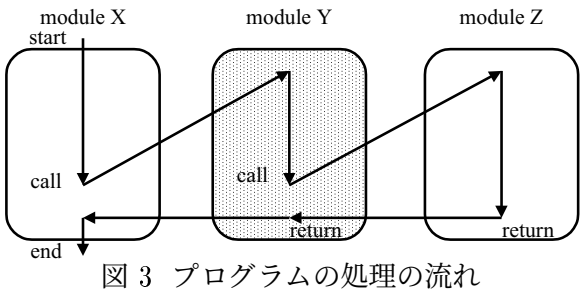

図 3 プログラムの処理の流れ

表 5 プログラムの各モジュール処理時間 (秒)

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

において，プロセッサ処理と入出力処理の処理時間比が CPU:I/O=1:1 であるような均等処理プログラムも用意した．各モジュールでは，表 5 で設定した処理時間を 1 とすると，プロセッサ処理 (1/4)，入出力処理 (1/2)，プロセッサ処理 (1/4) を順に行う．プロセッサ処理は特定の変数を 1 加算する処理であり，入出力処理は sleep 処理である．上記の理由により，プロセス早期停止法の評価にはプロセッサ処理プログラムを，プロセス優先実行法の評価には均等処理プログラムを使用した．

各プロセスは，疑似乱数を用いて不定間隔で起動し，共有プロセス数が 1 から 10 までの場合について，module Y に対して入替えを行い，その入替え時間を測定した．測定は 20 回ずつ行い，それらの平均値を測定結果とした．また，他プロセスの共存が，新方式の効果にどのような影響を与えるかを明らかにするため，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，特定の変数を 1 加算する処理を繰り返すプログラムである．測定は，BSD/OS ver2.1 が走行する計算機 (Pentium 90MHz) で行った．

4.2 プロセス早期停止法

図 4 と図 5 に，表 3 に示したプロセス早期停止法の 3 つの制御方式を適用した場合の入替え時間の測定結果を示す．図 4 は，他プロセスが存在しない場合，図 5 は，他プロセスが 1 つ同時走行している場合の測定結果である．各図では，各 type での特徴がよくわかるように，typeA，E，H の場合の測定結果のみを示している．typeA1，E1，H1 と typeA2，E2，H2 と typeA3，E3，H3 が，それぞれ，(早期停止 1) と (早期停止 2) と (早期停止 3) の各 type における測定結果である．

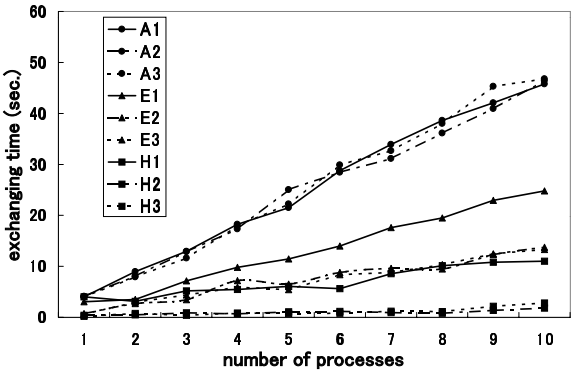

図 4 プロセス数と入替え時間の関係
(プロセッサ処理プログラム，他プロセスなし)

まず，図 4 から以下がわかる．

(1) Ai(i=1，2，3) は同様であるが，E1 や H1 に比べ E2 と E3 や H2 と H3 は入替え時間が短くなっている．すなわち，新方式の効果は，typeE と typeH の場合に大きく，typeA の場合には小さい．これは，新たな制御 (制御 2-1) と (制御 2-2) の制御対象が入替え要求時に入替え可能状態にあるプロセスであることによる．typeA の場合，新方式の効果があまり見られないのは，総処理時間 10 秒に対して，module X の処理時間は 1 秒と短く，入替え要求時に module X を実行中のプロセス，すなわち，入替え可能状態にあるプロセスの存在する確率が非常に低いためである．module X の処理時間が 5 秒，8 秒と長い，typeE，typeH では，新方式の効果が大きく，それぞれ，旧方式に対して，平均 45%，平均 86%入替え時間を短縮できた．したがって，新方式の効果は，入替え対象プログラムの総処理時間に対して，入替え可能状態であるモジュール処理時間の占める割合が高いほど，大きいといえる．

(2) E2 と E3 や H2 と H3 は大差なく，新方式 (早期停止 2) と (早期停止 3) の効果に違いが見られない．これは，(制御 2-1) と (制御 2-2) の違いが，その制御対象に他プロセスを含めるか否かの違いだけであることによる．他プロセスなしの場合，両方式の効果

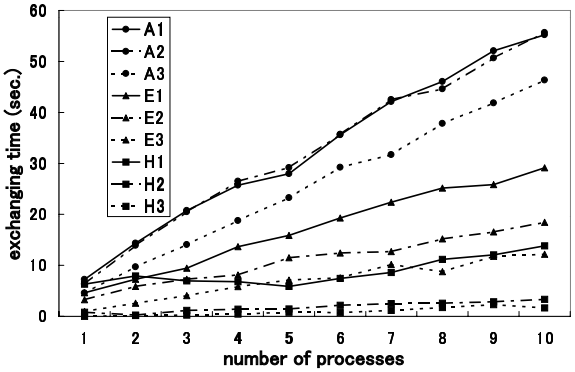

図 5 プロセス数と入替え時間の関係
(プロセッサ処理プログラム，他プロセスあり)

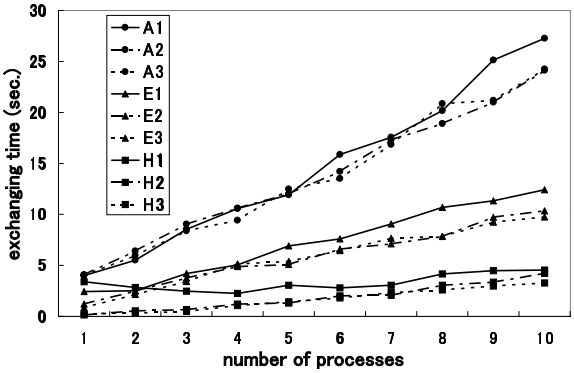

図 6 プロセス数と入替え時間の関係
(均等処理プログラム，他プロセスなし)

が等しいのは明らかである．

次に，図 4 と図 5 の比較から以下がわかる．

(3) (早期停止 1) と (早期停止 2) の入替え時間は図 5 の方が図 4 に比べ長くなっているの対して，(早期停止 3) の入替え時間は図 4 と図 5 において等しい．これは，他プロセスは (制御 2-2) の制御対象であり，(制御 1) および (制御 2-1) の制御対象外であることによる．他プロセスが入替え時間に与える影響時間は，入替え不可状態であるモジュール処理時間の総和に依存するため，その総和が長いほど，(早期停止 1) と (早期停止 2) の入替え時間は一様に長くなっている．これに対して，(早期停止 3) は，図 4 と図 5 における入替え時間が等しいことから，他プロセスが入替え時間に与える影響を完全に除去できるといえる．しかし，これは一方で，入替えとは全く無関係な他プロセスが停止していることを暗に意味する．

4.3 プロセス優先実行法

図 6 と図 7 に，表 4 に示したプロセス優先実行法の 3 つの制御方式を適用した場合の入替え時間の測定結果を示す．図 6 は，他プロセスが存在しない場合，図 7 は，他プロセスが 1 つ同時走行している場合の測定結果である．各図では，各 type での特徴がよくわかるように，typeA，E，H の場合の測定結果のみを示している．typeA1，E1，H1 と typeA2，E2，H2 と typeA3，E3，H3 が，それぞれ，(優先実行 1) と (優先実行 2) と (優先実行 3) の各 type における測定結果である．

まず，図 6 から以下がわかる．

(1) 新方式の効果は，typeE と typeH の場合に大きく，typeA の場合には小さい．これは，入替え不可状態であるモジュール処理時間の総和の占める割合が高い場合，新たな制御 (制御 B) によらずとも，入替え要求後に入替え不可状態にあるプロセスの走行する確率が高いためである．typeA の場合，総処理時間 10 秒に対して，module Y と module Z の処理時間の総和は 9 秒と長く，入替え要求時に module Y や module Z を実行中のプロセス，すなわち，入替え不可状態にあるプロセスの存在する確率は非常に高い．このため，新方式の効果はほとんど見られない．module Y と module Z の処理時間の総和が，5 秒，2 秒と短い，typeE，typeH では，旧方式に対して，それぞれ，平均 18〜21%，平均 47〜53%の入替え時間の短縮であり，新方式の効果が大きくなっている．したがって，新方式の効果は，入替え対象プログラムの総処理時間に対して，入替え不可状態であるモジュール処理時間の総和の占める割合が低いほど，大きいといえる．

(2) 新方式 (優先実行 2) と (優先実行 3) の効果に違いが見られる．(処理 1) ではなく，(処理 2) を行うことによって，短縮できる入替え時間は，入替え不可状態であるモジュールに含まれる入出力処理の処理時間の総和に依存する．このため，短縮時間は，入替え不可状態であるモジュール処理時間の総和が長いほど，長くなっている．

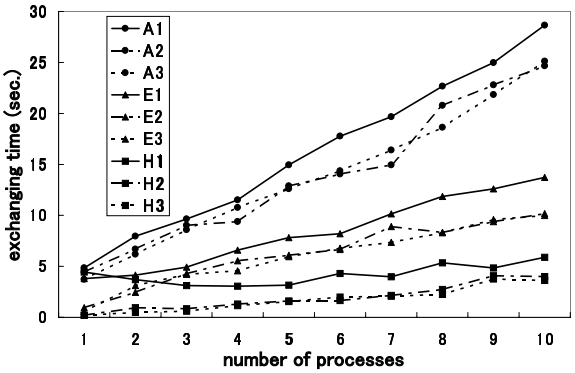

図 7 プロセス数と入替え時間の関係
(均等処理プログラム，他プロセスあり)

次に，図 6 と図 7 の比較から以下がわかる．

(3) 図 7 の方が図 6 に比べ各 type における新方式と旧方式の入替え時間の差分が広がっている．これは，(制御 B) により，他プロセスが入替え時間に与える影響を軽減できるためである．(制御 A) では，入替え要求後にいずれかのプロセスがモジュールに突入または脱出するまでの間，他プロセスの走行が発生する可能性がある．これに対して，(制御 B) では，入替え要求を契機として，入替え不可状態にある全プロセスに対して，READY キューのつなぎ変え処理を行うため，他プロセスの走行の発生を抑制できる．

5 あとがき

実行中プログラムの部分入替え法について，入替え条件を緩和した場合，入替え時間が長大化する問題への対処として，入替え可能状態にあるプロセスを早期停止する方法と，入替え不可状態にあるプロセスを優先実行する方法について述べた．具体的には，プロセス早期停止法として，入替え要求を契機に即座に入替え可能状態にあるプロセスを停止させる方式を提案した．また，プロセス優先実行法として，入替え要求を契機に，入替え不可状態にある全プロセスに対して，READY キューのつなぎ変え処理を行う方式を提案した．

さらに，両者の方法について実装および評価を行い，従来方式との比較評価の観点から，提案方式の有効性を示した．具体的には，プロセス早期停止法の提案方式の効果は，入替え対象プログラムの総処理時間に対して，入替え可能状態であるモジュール処理時間の占める割合が高いほど，大きいことを明らかにした．提案方式は，最大 86%入替え時間を短縮できた．また，提案方式により，他プロセスが入替え時間に与える影響を完全に除去できることを示した．一方，プロセス優先実行法の提案方式の効果は，入替え対象プログラムの総処理時間に対して，入替え不可状態であるモジュール処理時間の総和の占める割合が低いほど，大きいことを明らかにした．提案方式は，最大 53%入替え時間を短縮できた．また，従来方式よりも，他プロセスが入替え時間に与える影響を軽減できることを示した．

今後の課題として，2 つの対処法の共存による評価および提案した入替え時間の短縮法による他プロセスへの影響の明確化がある．

参考文献

[1] H.Muramatsu, M.Date, H.Yoshida, M.Kitaoka, and N.Kurobane, “Operating System SXO for Continuous Operation,” IFIP 92, Vol.1, pp.615-621, 1992.

[2] B.Snead, F.Ho, and B.Engram, “Operating System Features Real Time and Fault Tolerance,” Computer Design, pp.177-185, 1984.

[3] 盛合敏, 徳田英幸, “次世代 OS のためのマイクロカーネルトレイアーキテクチャ,” 情処研報, Vol.97, No.56, pp.1-6, 1997.

[4] 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.

[5] 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.

[6] 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.

[7] 中島雷太, 谷口秀夫, “入替え条件を緩和した実行中プログラム部分入替え法の評価,” 情処研報, Vol.98, No.71, pp.45-52, 1998.

[8] 中島雷太, 谷口秀夫, “実行中プログラム部分入替え法の評価 —他プロセスの影響—,” 情処コンピュータシステム・シンポジウム論文集, Vol.98, No.15, pp.79-86, 1998.

[9] 中島雷太, 谷口秀夫, “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,” 情処学論, Vol.41, No.6, pp.1734-1744, 2000.