社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－OS－103 (13)
2006／8／1

プログラムの動作を利用形態に適応させる適応制御機能

仁科 匡人， 谷口 秀夫

岡山大学大学院自然科学研究科

計算機の普及に伴い利用者も増大し，計算機の利用形態は多岐に広がっている．このため，利便性を高めるためには、それぞれの利用形態に適応した制御を行う機能が求められる．そこで，オペレーティングシステムを利用形態に適応させる機能として，適応制御機能を提案する．本機能は，利用者のプログラムの利用履歴に基づきプログラムの動作を制御する機能である。ここでは，オペレーティングシステムの中心的機能の 1 つであるプロセスの生成と終了の情報を利用し，プログラムの動作を利用形態に適応させる機能について述べる．

A mechanism which adapts program execution for its purpose

Tadato NISHINA, Hideo TANIGUCHI

Graduate School of Natural Science and Technology, Okayama University

Computers become popular and have been used in various situations. Thus, a computer has to process various requests from many users. Especially, operating system should be adapted for the demands of user programs to improve usability. Therefore, we propose a mechanism which adapts program execution for its purpose. The mechanism uses execution history for controlling programs. We focused on process creation and process termination in this article.

1. はじめに

計算機の低価格化により，計算機は，様々な場面で利用されるようになった．さらに，計算機の普及にともない，利用者が増大し，計算機の利用形態は，多種多様になっている．利用形態の増加により，利用者のオペレーティングシステム(OS)に対する要求は，様々に変化している．たとえば，各利用者により，高速に動作してほしいプログラムと動作が多少遅くてもよいプログラムは異なる．

しかし，既存の OS では，どんな利用者に対しても同様なプログラムの実行制御を行なっていることが多い．このため，利用者によっては，高速に動作してほしいプログラムの動作が遅くなるといったように，OS が利用者にとって好ましくないプログラムの実行制御を行うことがある．このよ

うな背景から，OS に対し利用形態への高い適応性が求められる．

ここでは，OS に適応性を持たせる機能として適応制御機能を提案する．具体的には，適応制御機能は，利用者のプログラムの利用履歴を考慮し，プログラムの実行を制御する．プログラムの利用履歴を獲得するために，プログラムの動作を監視する．プログラムの動作には，プロセスの生成や終了，プロセス間通信やメモリ利用，CPU 利用といったものがある．プログラムの利用履歴に基づき，頻繁に利用されるプログラムはスーパバイザモードへの走行モード変更，メモリ常駐化およびプロセスの事前生成を行い，高速に動作させる．一方，利用頻度の低下したプログラムはユーザモードへの走行モード変更，メモリ非常駐化およびプロセス生成契機の変更を行う．これにより，OS は，プログラムの動作を利用者のプログラムの利用形態に適応させることが可能になる．

ここでは，適応制御機能の設計方針と基本的な機能について述べる．

2. 適応制御機能

2.1. 目的

それぞれの利用者により，計算機の利用形態は様々である．このため，OS は，全ての利用者に同様なプログラムの動作制御法を提供するのではなく，各利用者に適応したプログラムの動作制御法を提供する必要がある．そこで，適応制御機能は，上記の要求を満たすために OS に適応性を持たせることを目指す．

適応制御機能は，利用者のプログラムの利用履歴を記録し，プログラムの利用履歴に基づいて OS にプログラムの動作を制御させる．これにより，OS は，利用者の利用履歴を考慮したプログラムの実行制御を行うことができる．つまり，OS は，各利用者に対し，適切なプログラムの実行制御法を提供することができる．

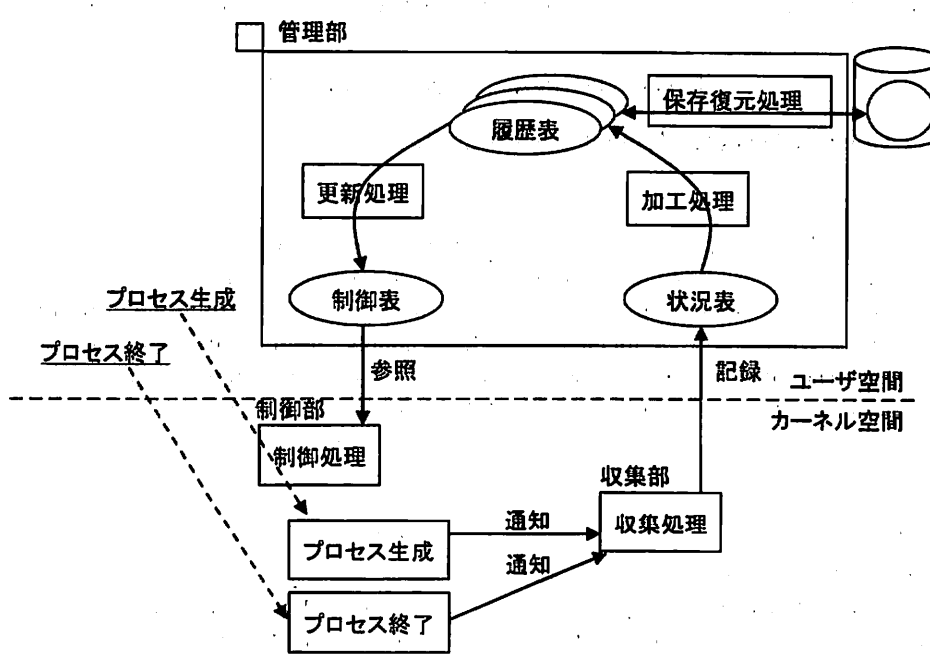

図1 適応制御機能の概要

ここでは，OS の中心的機能の 1 つであるプロセスの生成と終了について述べる．

2.2. 要求

本機能への要求として，以下のものがある．

(要求 1) 処理オーバヘッドの抑制

(要求 2) 適応制御のために利用する項目の増減時の工数抑制

適応制御機能により，OS 処理の負荷が増加することは好ましくない．このため，(要求 1)は重要である．この要求を満足するには，本機能の実装において，カーネル内外の機能分担や管理表の操作を工夫する必要がある．

適応制御のために利用する項目(例えば，プロセス生成，プロセス終了，メモリ領域確保，メモリ領域解放)は，システム動作の違いにより変化する．したがって，(要求 2)は，この変化への対応を容易にするものである．

2.3. 基本機能

適応制御機能は，処理内容により大きく 3 つの処理部に分類できる．また，各処理部が利用する管理表も大きく 3 つに分類できる．この様子を図 1 に示し，以下に説明する．

(1) 収集部：プログラムの生成や終了に関する情報を逐次記録する．

(2) 管理部：適応制御を行うための情報を収集し加工し更新する．また，加工した 情報の保存や復元も行う．

(3) 制御部：更新された最新の情報に基づき，プログラムの動作を制御する．

これらの処理部において，情報の逐次記録やプログラムの動作制御は，カーネルの他の処理部分と深く関係する．このため，収集部と制御部のみをカーネル内部に配置し，処理オーバヘッドの抑制(要求 1)を進める．さらに，負荷が大きな情報加工処理を管理部で行うようにすることで，収集部と制御部の処理内容を簡素化する工夫も行っている．

表1 管理表の役割

	管理表名	役割	表の単位
(A)	状況表	プロセスに関する情報を時系列で保持	システム
(B)	履歴表	プログラム毎に情報を分離加工し、プログラムの動作履歴を把握できる形で情報を保持	プログラム
(C)	制御表	制御に必要な観点から整理更新した情報を保持	システム

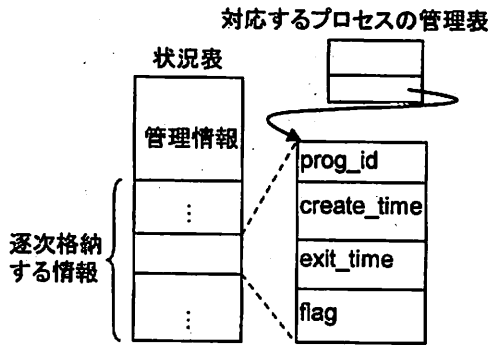

図2 状況表の構造

次に，各管理表の役割を表 1 に示し，以下に説明する．

(A) 状況表

状況表は，プログラムの生成や終了に関する情報をそのまま時系列で保持する．この管理表は，システムに 1 つ存在し，各エントリは，1 つのプロセスに対応する．この様子を図 2 に示す．状況表は，管理情報と逐次に記録する情報からなる．管理情報は，逐次に記録する情報について記録済のエントリの範囲や次の記録位置などの情報を保持する．逐次に記録する情報のエントリは，各プロセスに対応し，以下の情報を保持する．

(a) prog_id：プログラム識別子

(b) create_time：プロセス生成時刻

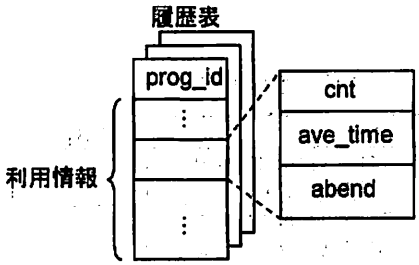

図3 履歴表の構造

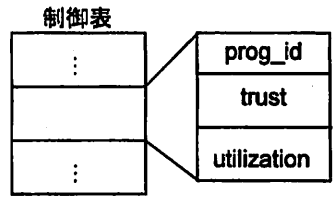

図4 制御表の構造

(c) exit_time：プロセス終了時刻
(d) flag：エントリの状態(未使用，記録中，記録済)

prog_id と create_time は，プロセス生成時，exit_time は，プロセス終了時に記録する情報である．プロセス終了時の exit_time への記録処理を高速化するため，当該エントリへのポインタ情報を対応するプロセスの管理表に持たせることにした．なお，flag は，エントリ操作を制御するための情報である．

(B) 履歴表

履歴表は，状況表の情報に基づき，加工し更新された情報を保持する．この管理表は，プログラム毎に１つ存在する．この様子を図３に示す．履歴表は，プログラム識別子とプログラムの利用情報からなる．

プログラムの利用情報は，1 エントリが以下の情報を保持する．

(a) cnt：一定時間(T1)内に動作(起動と終了)した回数
(b) ave_time：一定時間内のプログラムの平均動作時間
(c) abend：一定時間内の異常終了回数

なお，利用情報の各エントリは，一定時間(T1)の情報を保持しており，時系列に並んでいる．

(C) 制御表

制御表は，制御に必要な観点から整理更新された情報を保持する．この管理表は，システムに１つ存在し，各エントリは，１つのプログラムに対応する．この様子を図４に示す．制御表は，１つのエントリが１つのプログラムに対応し，以下の情報を保持している．

(a) prog_id：プログラム識別子
(b) trust：信頼度(プログラムの信頼性を示す値)
(c) utilization：利用度(プログラムがどの程度利用されているかを示す値)

信頼度や利用度についての算出方法およびこの情報に基づく制御法については，2.4.2. 項で説明する．

なお，これらの管理表は，主に管理部によって更新されるため，いずれも管理部が保有(ユーザ空間に配置)する．これにより，データ授受の効率化を図る．

本機能に関する処理の開始時と終了時には，初期化処理と終了処理を行う．具体的には，初期化処理では，各管理表の領域確保を行い，履歴表の内容を外部記憶装置から復元する．さらに，復元した履歴表に基づき更新処理を行い，制御表を作成する．なお，更新処理については後述する．また，終了処理では，履歴表の内容を外部記憶装置へ書き込んで永続化を行い，各管理表の領域開放を行う．

2.4. 各制御部

2.4.1. 収集部

収集部は，カーネル内部で動作するため，処理の単純化が必要である．このため，以下の処理とした．

(1) プログラム生成の通知により，状況表の新エントリを確保し，必要な情報を記録

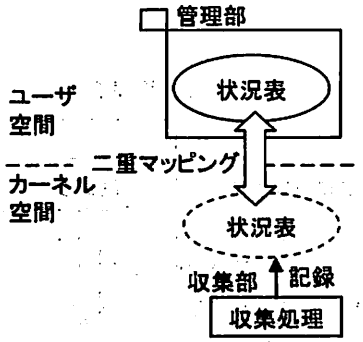

図5 収集部

する．このとき，次の情報記録位置を更新する．

(2) プロセス終了の通知により，当該エントリに必要な情報を記録する．このとき，当該エントリの検索を簡素化するため，先にも述べたように対応するプロセスのプロセス管理表に状況表のエントリ番号を持たせることとした．

また，収集部は，状況表を介して管理部と情報の授受を行う．収集部と管理部の連携を高速に行なうために，以下の対処を行なう．

(A) 状況表の二重マッピング

収集部と管理部の情報授受にかかるオーバヘッドを削減するため，状況表を管理部カーネル空間にもマッピングする．この様子を図 5 に示す．これにより，収集部と管理部の情報授受を高速に行なうことが可能になる．

(B) 状況表操作の簡素化

状況表は，収集部と管理部の両者から操作される．収集部と管理部の状況表への操作を表 2 に示し，以下に説明する．

収集部は，情報記録位置，未使用エントリの flag, 記録中エントリの flag を更新し，その他の項目については参照のみである．一方，管理部は，利用しているエントリの範囲の開始位置および記録済エントリの flag を更新し，その他の項目については参照のみである．このように，両処理部は，同じ項目を更新することがない．これにより，排他制御をする必要が無く状況表の操作が簡素化できる．

表2 状況表の操作

項目	収集部	管理部
(A) 情報記録位置	更新	参照のみ
(B) 利用しているエントリの範囲の開始位置	参照のみ	更新
(C) 未使用エントリのflag	更新	参照のみ
(D) 記録中エントリのflag	更新	参照のみ
(E) 記録済エントリのflag	参照のみ	更新

2.4.2. 管理部

管理部の処理は，主に以下の 3 つの処理に分割できる．

(1) 加工処理：状況表の情報を履歴表の情報に加工する処理

(2) 更新処理：制御表の情報を更新する処理

(3) 保存復元処理：履歴表の情報を保存および復元する処理

加工処理は，プロセス毎の情報を時系列に保持する状況表に基づき，プログラム毎に情報を整理加工し，履歴表を作成する．なお，この処理は，一定の時間間隔(T1)で周期的に行う．処理の具体的内容を以下に示す．

(A) 状況表から同じプログラムの情報を集める．

(B) エントリ数を数える．(cnt)

(C) 各エントリの動作時間を計算する．それらを合計した値と(B)で求めた動作回数(cnt)を利用して，時間間隔(T1)での平均動作時間を算出する．(ave_time)

(D) 異常終了の回数を数える．(abend)

上記の処理により，プログラム識別子と cnt, ave_time および abend の組み合わせが出来上がる．この情報を利用して，履歴表を更新するときに 3 つの場合がある．

(場合 1) 既に履歴表に登録してあるプログラムの情報は，前回の記録情報の後ろに情報を記録する．

(場合 2) 履歴表に登録されていないプログ

ラムの情報は，新たに履歴表にプログラムを登録し，その履歴表に情報を記録する．

(場合３) プログラムは履歴表に登録されているが，状況表に当該プログラムの情報が無いときは，前回の情報の後ろに cnt, ave_time, abend の全ての項目の値が 0 の情報を記録する．

更新処理は，履歴表の情報に基づき制御表を更新する．この処理は，一定の時間間隔(T2)で周期的に行う．処理の具体的内容を以下に示す．

(A) 一定の時間間隔(Ta = T1 * n)の情報を履歴表から取得し，プログラムの情報毎に合計動作回数，平均動作時間および合計異常終了回数を計算する

(B) 利用度(A)の算出は，一般的に
A = f1(合計動作回数，平均動作時間)
である．例えば，
A = 合計動作回数 * 平均動作時間
として考えることができる．

(C) 信頼度(B)の算出は，一般的に
B = f2(合計異常終了回数)
である．例えば，
B = 合計異常終了回数
として考えることができる．

(D) (A)～(C)の処理を履歴表に登録されているプログラム全てについて繰り返す

なお，(B)の処理において利用度(A)が初めて利用度閾値(S1)を超えた場合は，制御部へ当該プログラムの生成要求を通知する．この処理については，2.5.節で説明する．

保存復元処理は，本機能に関する処理の開始時と終了時に行われる．保存処理は，履歴表を外部記憶に保存する処理であり，本機能に関する処理の終了時に行われる．復元処理は，外部記憶に保存された履歴表を読み込む処理であり，本機能に関する処理の開始時に行われる．なお，復元処理が行われた直後に，復元した履歴表に基づき更新処理を行ない，制御表を作成する．

2.4.3. 制御部

制御部は，サービスプログラムからのプロセス生成の要求を契機として，制御表に基づき以下の処理を行う．

(1) 既にプロセスが生成されていれば，そのプロセスの動作を開始させる．

(2) プロセスが利用するプログラムの信頼度が低ければ，プロセスを生成しない．

(3) 上記(１)(２)以外の場合，利用度に見合った形態でプロセスの生成を要求 する．具体的な内容については，２．５節で説明する．

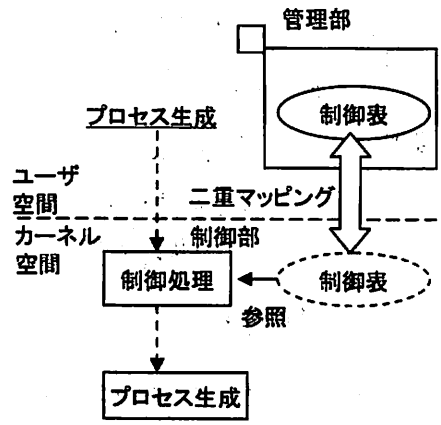

図6 制御部

また，管理部からのプロセス生成の要求を契機として，制御表に基づきプロセスの事前生成を行い，プロセスの動作を停止させておく．これにより，利用度の高いプログラムのプロセス生成を高速化する．

さらに，制御部は，管理部との情報授受に伴うオーバヘッドを削減するため，制御表をカーネル空間にもマッピングする．この様子を図６に示す．

2.5. 制御表に基づくプロセス生成法の事例

ここでは，制御表の信頼度や利用度に基づいて，プロセスの生成を制御する機構を説明する．基本的な考え方は，以下の２つである．

(1) 信頼度が低いプログラムは動作させない．

(2) 利用度が高いプログラムは，事前プロセス生成などにより，プロセス生成の高速化を図る．

具体的には，信頼度が信頼度閾値(R)より低

表3 プログラムの起動状態と閾値の関係

	(状態1)	(状態2)	(状態3)	(状態4)
プログラム走行モード	ユーザモード	ユーザモード	ユーザモード	スーパバイザモード
メモリ常駐か否か	非常駐	非常駐	常駐	常駐
プロセス生成契機	生成要求時	生成要求前	生成要求前	生成要求前

いプログラムは，プロセス生成を要求されても，プロセス生成は行わない．

また，利用度についても，やや複雑であるが，同様な処理を行う．プログラム走行モード，メモリ常駐か否か，およびプロセス生成契機の観点から，プロセスの状態を４つに分類する．この様子を表３に示す．

(状態１)は，プロセス生成要求時にプロセスの生成が行われる．生成されたプロセスはユーザモードで動作しており，メモリ常駐ではない．

(状態２)は，(状態１)とプロセス生成契機の観点のみ異なる．(状態 2)のプロセスは，プロセスの事前生成が行われる．こうすることで，(状態１)のプロセスに比べプロセスの生成を高速化することができる．

(状態３)は，(状態２)とメモリ常駐か否かの観点のみ異なる．(状態３)のプロセスは，利用するメモリ空間が常駐化されている．こうすることで，ページフォルトの発生を抑え，(状態２)のプロセスと比べ高速に動作することが可能になる．

(状態４)は，(状態３)とプロセス走行モードの観点のみ異なる．(状態４)のプロセスは，走行モード変更機構[1]を利用してスーパバイザモードで動作させる．こうすることで，システムコールやupコールを利用していたカーネルとの連携を関数呼び出しで行うことができる．このため，(状態３)に比べカーネルとの連携オーバヘッドが削減でき，さらに高速に動作することが可能になる．

したがって，(状態１)から(状態４)になるほど，利用度が高いプログラムを利用したプロセスである．

各状態間の遷移は，以下の考え方に基づく．

(１) 利用度(A)と利用度閾値(Si,Ui)の大小関係による．

(２) 利用度が高い状態へ移行する際の閾値と，利用度が低い状態へ移行する際の閾値は，必ずしも同じではない．

例えば，１つのプログラムの利用度が高くなる場合について考える．このとき，

(場合１) A ＜ S1 であれば，プロセスを(状態１)で生成する．

(場合２) S1 ＜ A ＜ S2 であれば，プロセスを(状態２)で生成する．

(場合３) S2 ＜ A ＜ S3 であれば，プロセスを(状態３)で生成する．

(場合４) S3 ＜ A であれば，プロセスを(状態４)で生成する．

とする．一方，利用度が低くなる場合について考える．このとき，

(場合１) A ＜ U1 であれば，プロセスを(状態１)で生成する．

(場合２) U1 ＜ A ＜ U2 であれば，プロセスを(状態２)で生成する．

(場合３) U2 ＜ A ＜ U3 であれば，プロセスを(状態３)で生成する．

(場合４) U3 ＜ A であれば，プロセスを(状態４)で生成する．

とする．なお，Si < Si+1 (i=1,2), Ui < Ui+1 (i=1,2) である．上記の利用度閾値の設定により，

(仮定１) Si < Ui (i=1,2,3)であれば，利用度が高い状態になりやすく，利用度が低い状態になりにくい．

であり，逆に，

(仮定２) Si > Ui (i=1,2,3)であれば，利用度が高い状態になりにくく，利用度が低い状態になりやすい．

である．したがって，利用度閾値の設定を

工夫することで，システムの利用形態に合わせた環境構築が可能である．

3. おわりに

適応制御機能について設計方針と基本的な機能について述べた．設計方針として，オーバヘッドの削減と適応制御のために利用する項目の増減時の工数抑制がある．基本的な機能については，情報収集を行う収集部，情報の管理を行う管理部，およびプログラムの動作を制御する制御部の３つの処理部により構成され，情報の収集に利用する状況表，情報の保存に利用する履歴表，プログラムの制御に利用する制御表の３つの管理表を利用することを示した．また，本機能を利用した事例として，プロセス生成方法について述べ，利用度閾値の設定を工夫することで，システムの利用形態に合わせた環境構築が可能であることを示した．

残された課題として，適応制御機能の実装がある．我々の研究室で開発を進めている***AnT***オペレーティングシステム[2] [3]に本機能を実装する予定である．

謝辞 本研究を進めるにあたり検討に御協力頂いた岡山大学大学院自然科学研究科乃村能成講師並びに田端利宏助教授に感謝いたします．

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]横山和俊，乃村能成，谷口秀夫，丸山勝巳，“応用プログラムの走行モード変更機構”，情処研報，2004-OS-95，pp.25-32，2004.

[2]http://www.swlab.it.okayama-u.ac.jp/lab/tani/research/ant/index-j.html

[3]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”，情処研報，2006-OS-103，2006．(掲載予定)