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

2008－OS－108 (19)
2008／4／24

プロセスの初期動作に着目した使用資源量の見積もり

芝 公仁
龍谷大学

オペレーティングシステムは，個々のプロセスの特性に合わせて資源割り当てを行う必要がある．そのため，多くのシステムでは，プロセスの動作に応じて，資源割り当ての方針を決めている．しかし，プロセスの動作を確認してから方針を決める方法では効率化に限界がある．本稿では，プロセスの初期動作からプロセスの動作の特性を得る手法について述べる．また，本手法に基づき構築したプロセスが使用するメモリ量を予測するシステムについても述べる．

Estimation of Needed Resources from Initial Behaviors of Processes

Masahito Shiba
Ryukoku University

Operating systems must provide resources to processes according to the characteristic of the processes. In many systems, operating systems monitor activities of the processes and select policies of providing resources to the processes. However, the way that an operating system decides a policy after a process has run has limited effect. In this paper, a method of estimating characteristic of a process from the initial behavior of the process is described. A system for estimating needed memory size of processes based on the proposed method is also described.

1 はじめに

計算機上で動作するアプリケーションには様々な種類があり，その動作も多様化してきている．これらのアプリケーションが効率的に動作できるように適切に計算機資源を割り当てることは，オペレーティングシステムの重要な役割であるが，アプリケーションの資源に対する要求はそれぞれ異なるため，単一の固定的な資源管理手法を用いることは，資源利用効率の低下を招く．個々のアプリケーションの動作に応じて資源管理手法を変更できることが望ましいが，各アプリケーションの資源使用特性は実行時に決まることも多く，予め静的に情報を保持し対応することは困難である．

個々のプロセスの動作に合った資源割り当てを行うことで効率化が図れるため，既存のシステムでは，動作しているプロセスを監視し，それによって得られた情報から資源割り当ての方法を決定することが多い [1, 2]．この場合，プロセスが動作した後に対応が行われるため，効率化には限界がある．本稿では，このような問題に対応するための，プロセスの初期動作から当該プロセスがどのような資源要求の特性を有するかを予測する手法について述べる．

提案手法では，機械学習の技術を用いて，プロセスの初期動作の情報から当該プロセスがどのような特徴を持ったプロセスであるかを識別する．本手法には，以下に示すような利点がある．

- 初期動作のみからプロセスの特徴が分かる
- 少ない情報から予測することができる
- 様々な資源やその管理手法に適用可能である

プロセスの初期化処理の動作から，将来どのように動作するかを予測できると，それに応じた資源管理を行うことができる．例えば，新たに起動したプロセスが大量のメモリを必要とするものであると予測できれば，ディスクに書き出されていないバッファを直ちに書き出し，空きメモリを作成する準備を予め行っておき，起動したプロセスを効率よく動作させることが可能である．

提案手法では，プロセスの動作を始終監視するのではなく，起動後から定められた期間のみの情報を使用するため，必要なデータ量が少ない．また，単純に過去の履歴を使用するのではなく，識別器によってプロセスの動作の特徴を予測するため，未知のプロセスにも対応することが可能である．また，

システムの環境の変化にも適応できるといった利点もある．

提案手法は，特定の資源に特化したものではなく，様々な資源に対して適用できる汎用的な手法である．また，既存の資源管理手法との親和性も高い．既存の手法では，プロセスの動作からその特徴を確認した後，それに適した資源管理を行うが，提案手法を用いると事前にプロセスの特徴を知ることができるため，早い段階から既存の有用な資源管理手法を適用することができる．すなわち，提案手法と既存の手法を組み合わせることによって，効率化を図ることが可能である．

以下，本稿では，2章で提案手法とそれに基づくシステムの構成について述べ，続けて，3章で提案手法のメモリ資源への適用について述べる．また，4章で，システムの性能および識別の精度について評価を行う．

2　システムの概要

効率的な資源管理を行うためには，個々のプロセスの特徴に応じて資源を割り当てる必要がある．これを効果的に行うには，できるだけ早い時期にプロセスの特徴を検出することが重要である．提案手法では，プロセスの動作を監視し，それによって得られた情報から当該プロセスの特徴を調べ，得られた特徴を考慮し資源管理を行う．具体的には，プロセスの起動後，初期化処理の動作状況から当該プロセスがどのような特徴を持ったプロセスかを識別し，その特徴に応じた資源割り当てを行い，効率的な資源利用を実現する．本章では，提案手法によるプロセスの特性の識別と，これを行うシステムの構成ついて述べる．

2.1　システム構成

提案手法では，次の3つのモジュールによって，プロセスの特徴を調べ，それに基づきプロセスを分類する．また，その結果をオペレーティングシステムの資源管理で活用する．

- プロセス特性管理モジュール
- プロセス特性識別器
- プロセス特性データベース

これらの関係を図1に示す．プロセス特性管理モジュールは，プロセスの動作を監視し，それによって得られた特性情報を記録する．また，プロセス特性識別器からの要求に応じて，記録した特性情報を渡す．このように，プロセス特性管理モジュールは，プロセスの特性情報を取得・保持するが，どのような情報を取得･保持するかは，どのようなプロセス特性の識別を行うかによって決まる．例えば，メモリ管理のための識別を行う場合，各プロセスのメモリ使用状況を監視し，また，スケジューリングのための識別を行う場合，各プロセスの実行状況を監視する．プロセスを監視することによって得られるデータをすべて記録することは，CPU負荷や必要になるメモリ量が非常に大きくなる．提案手法では，プロセスの動作状況のすべてを監視の対象とするのではなく，プロセスの初期動作のみを対象としている．すなわち，プロセスが生成された後，定められた期間内の動作に関する情報のみが対象となる．

プロセス特性識別器は，プロセスの特性情報をもとに，当該プロセスがどのような特徴を持っているかを識別する．特徴の識別には，サポートベクターマシンを用いる．すなわち，予め，プロセスの特徴的な動作をもとにプロセスのクラスを定めておき，特性情報から各プロセスがどのクラスに属するかを実行時に分類する．例えば，メモリ資源に関しては，多くのメモリを確保しようとするプロセスのクラス，少ないメモリで動作するプロセスのクラスなどを定義することができる．

プロセス特性データベースには，プロセスがどのクラスに属するかをプロセスの特性から識別するための知識が格納される．サポートベクターマシンでは，識別するための知識は，サポートベクターである．提案手法では，実際に動作したプロセスの初期動作の特性と当該プロセスがどのクラスに属するかを学習データとして，サポートベクターの算出を行う．

2.2　サポートベクターマシン

プロセスがどのような特徴を持っているかの識別に使用しているサポートベクターマシン [3, 4] は，機械学習を用いた識別手法のひとつである．教師あり学習を行うため，本システムでは，過去のプロセ

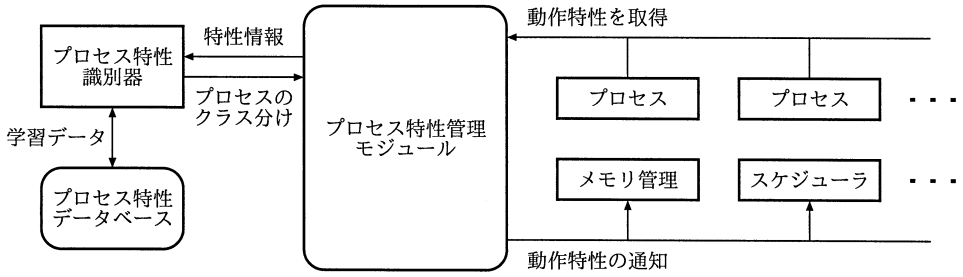

図 1　システムの構成

スの挙動の履歴を用いて学習データを作成し，識別器を構成する．

プロセスの特徴の識別にサポートベクターマシンを使用する利点として，次の 3 つがあげられる．

- 特徴の識別を行うために必要になるデータ量が少なくてよい
- システム構成時には動作を確認できない未知のプロセスに対応可能である
- 様々な資源に適用できる汎用性があり，また，新たな資源への対応も容易である

プロセスの資源利用の特性情報は，非常に複雑な情報であり，単純に履歴をとる手法では，履歴が膨大な量となる．履歴のデータ量の増加は，システムの資源の消費し，特徴の検出処理も負荷が高くなるため，資源管理のための手法としては不適当である．

提案手法では，機械学習の技術を用いることによって，プロセスの特徴を少ない情報をもとに調べることが可能である．また，サポートベクターマシンでは，超平面での 2 値分類が基本的な処理となり，必要なデータのみを用いて分類処理を行うため，比較的低負荷の処理で特徴の識別を行うことができる．

また，機械学習を用いると，過去に動作させたことのないプロセスにも対応できるという利点がある．様々なアプリケーションが開発され使用される環境においては，予めすべてのプロセスについて調べておくことはできないため，未知のプロセスにも対応できることは重要である．

サポートベクターマシンは，様々な用途に使用できる汎用の仕組みであり，これを利用する提案手法も，特定の資源に限られることはなく，様々資源に対して適用できる手法である．すなわち，プロセスの資源使用に関する情報を何らかの値として表現することができれば，それをもとに，プロセスの特徴を調べることができる．また，特徴によってプロセスを分類するための手法であり，それをもとにどのような資源管理を行うかは自由であるため，既存の資源管理手法と組み合わせて使用することもできる．

3　メモリ資源への適用

メモリ管理はオペレーティングシステムにおいて，最も重要な役割のひとつである．メモリの割り当てによって，ファイルのキャッシュが有効に機能したり，スワップアウトによって処理速度が低下したりする．すわなち，どのようにメモリ管理を行うかは，プロセスの実行速度にも影響する重要なものである．

既存の手法では，プロセスがどのようにメモリ資源を使用するかを監視し，その結果を用いて，メモリ管理の方法を決定するという手順であった．しかし，この場合，プロセスの挙動を確認してから対応するため，必ずしも効率的であるとはいえない．そこで，提案手法を用いてプロセスの初期動作から当該プロセスが将来どの程度のメモリ量を必要とするかを見積もる．

以下，本章では，提案手法のメモリ管理への適用，および，これに基づいたプロセスのメモリ使用量を予測するシステムについて述べる．

3.1　メモリ使用の特性

使用メモリ量の特性を調べるために，実際に広く利用されているアプリケーションのメモリに関する特性を調べる実験を行った．本実験では，Linux-2.6.23.13 上で libc-2.6.1 を使用するプロセスを実行し，使用メモリ量を確認している．

getty，bash，tgif，emacs の 4 つのプロセスにつ

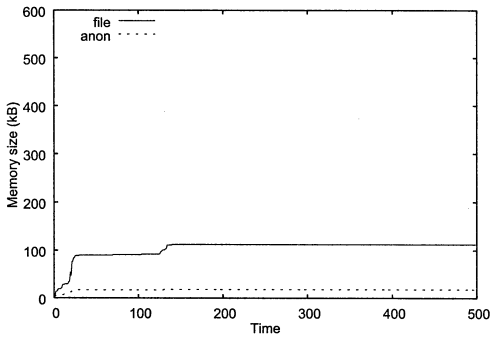

図 2　getty の使用メモリ量

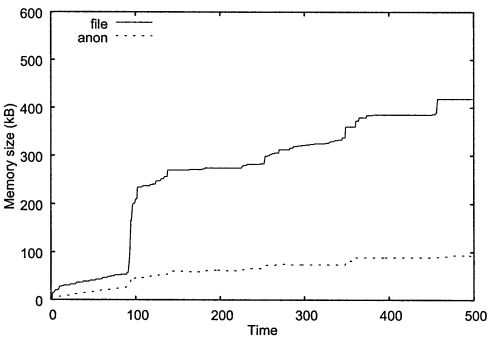

図 4　tgif の使用メモリ量

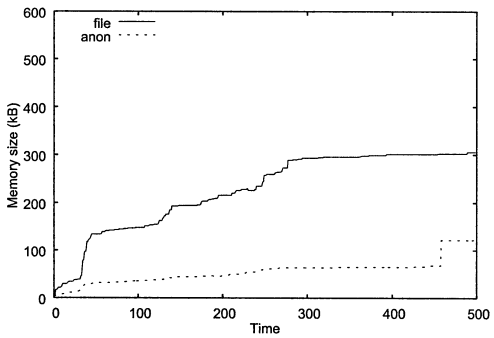

図 3　bash の使用メモリ量

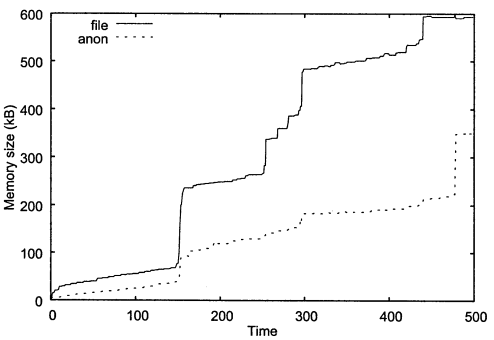

図 5　emacs の使用メモリ量

いて，起動後の使用メモリ量の変化を図 2, 図 3, 図 4, 図 5 に示す．グラフの横軸は，仮想的な経過時間であり，当該プロセスがシステムコールを発行するたびに増加するカウンタの値である．グラフの縦軸は，使用メモリ量である．各グラフにおいて，次の 2 つの使用メモリ量が示されている．

- file
  ファイルがマップされたメモリ領域
- anon
  ファイルに対応付けられていないメモリ領域

file の領域において，マップされるファイルの多くは，当該プロセスの実行ファイルや，使用している共有ライブラリのファイルである．また，anon は，スタックやヒープなどに使用される領域である．

各グラフから分かるように，4 つプロセスのすべてで，file のメモリ量が anon のメモリ量より多くなっている．これは，プロセス起動直後は，実行ファイルや共有ライブラリのファイルがマップされたメモリが多く，プロセスに実行に使用するデータやスタックのためのメモリが少ないためであると考えられる．

getty は，キャラクタデータの入出力が主な処理となるアプリケーションである．getty では，他のものに比べ早い段階でメモリの確保が終了し，file のメモリと anon のメモリの両方で，増加が止っている．また，増加後の使用メモリ量も他のものに比べ少ない．

bash も，getty と同様に，キャラクタデータの入出力が主な処理となるアプリケーションである．ただし，単純な機能のみを有する getty に対して，bash は実現している機能が多く，また，必要とする共有ライブラリも多い．そのため，getty に比較し，多くのメモリを使用している．また，グラフから，getty に比べ，必要なメモリの確保に長い時間を要していることが分かる．

tgif は，getty や bash とは異なり GUI を持つ．X Window System のクライアントとして動作するウ

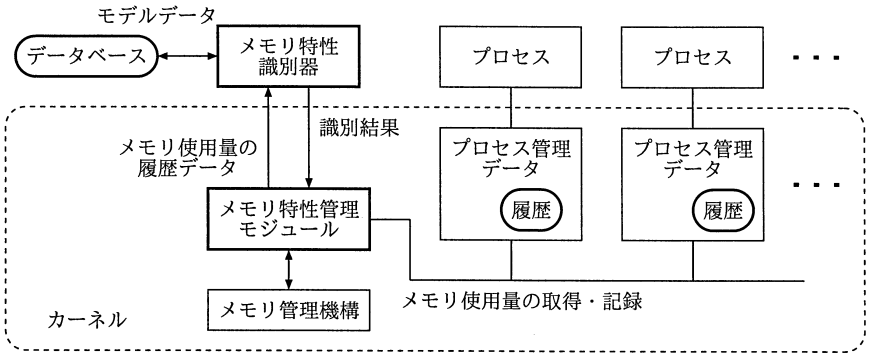

図 6　メモリ使用特性の識別

インドウアプリケーションであり，その動作は複雑で，使用する共有ライブラリも多い．そのため，ファイルのマップされたメモリを多く使用している．

emacs は，tgif と同様に，X Window System のクライアントとして動作するため，多くの共有ライブラリをマップするためのメモリを必要とする．また，emacs は，拡張機能を多く有するため，tgif に比べ，初期化時の処理内容が複雑で，ファイルの読み出しも多い．その結果，使用するメモリ量は，4 つのアプリケーションの中で最も多くなっている．

このように，アプリケーションは，それぞれ，使用するメモリ量が異なるだけでなく，初期化動作時の使用メモリ量の変化に特徴が見られる．提案手法を用いると，このような特徴から，プロセスが使用するメモリ量を予測することが可能になる．プロセスの起動時に，そのプロセスが将来要求するメモリ量を予測することができると，リアルタイム処理や QoS 制御などが要求される環境において，効果的な資源管理を行うことができると考えられる．

3.2　システム構成

提案手法を用いて，プロセスが初期化動作時に使用するメモリ量の変化を入力とし，当該プロセスが将来使用するメモリ量について予測するシステムを構築した．本システムの構成を図 6 に示す．図に示されるように，本システムでは，カーネル内あるいはカーネル外に配置される以下のモジュールが協調して動作する．

- メモリ特性管理モジュール
- 履歴データ
- メモリ特性識別器
- データベース

本システムでは，各プロセスが使用しているメモリ量を取得するために，カーネル内に使用メモリ量を取得するメモリ特性管理モジュールを組み込んでいる．使用メモリ量の取得は，プロセスがシステムコールを発行したときに行われる．取得された使用メモリ量は履歴として保持されるが，履歴に残す回数は予め決められており，この回数を越えた場合，システムコールが発行されても使用メモリ量の取得は行われない．すなわち，プロセス生成後，定められた回数分の使用メモリ量の変化が記録されることになる．この履歴データは，カーネルが持つプロセス管理用のデータ内に保持される．また，メモリ特性管理モジュールは，履歴データをメモリ特性識別器に渡す機能も持つ．

履歴データをもとに，資源量の見積もりを行う処理は，ユーザプロセスとして動作するメモリ特性識別器で行われる．ユーザプロセスとして実現することによって，識別処理という複雑な処理をカーネルの外で行うことが可能になる．識別処理は，通常のカーネル内の処理と比較すると長い時間を要する処理である．ユーザプロセスとして実現することによって，カーネルの処理を軽減できるといった利点もある．また，適切に優先度を設定することによって，そのときどきに応じて，識別処理を適当にスケジューリングし実行することができる．

識別処理は機械学習によって実現されるため，知識となるモデルデータが必要になる．これは，識別器からアクセス可能なデータベースで保持される．

プロセスがどのように識別されるかは，このモデルデータによって決まるが，これは，使用するアプリケーションの種類や，アプリケーションが使用するライブラリに影響される．そのため，使用する環境ごとにモデルデータを作成する必要がある．しかし，識別処理とモデルデータの作成は同様の手順で行うことができるため，モデルデータの作成は大きな手間にはならない．また，識別結果が正しかったか否かは，初期化完了後，プロセスが使用するメモリ量がどのように変化したかを調べれば，簡単に判断することができる．そのため，必要に応じて，実際にシステムを稼働させながら学習を行い，モデルデータを更新することも可能である．このようにしてモデルデータを動的に更新すると，システムの変化に動的に適応することが可能となる．

4 評価

本章では，3章で述べたプロセスのメモリ使用の特性を識別するシステムの評価を行う．3章で述べたように，プロセスの初期動作における使用メモリ量の変化は，当該プロセスが将来使用するメモリ量を予測するのに役立つ．そこで，識別器への入力をプロセスが使用しているメモリ量の変化とする．また，各プロセスがメモリを何バイト使用するかを値で予測するのではなく，予測使用メモリ量に応じて4つにプロセスを分類する．

4.1 性能評価

識別器への入力データとなる使用メモリ量の変化に関するデータはカーネル内で取得・保持されるが，この処理は，プロセスがシステムコールを発行したときに，当該システムコールの処理と同時に行われる．すなわち，必要な回数分の履歴が作成されるまでは，システムコールが発行されるたびに，メモリ使用量を取得するための処理が行われることになる．メモリ使用量取得の処理時間を計測するために，本システムを用いた場合と用いなかった場合の2つで，システムコールの処理時間を計測した．本実験には，Core Solo U1300（1.06GHz）を搭載するPCで，Linux-2.6.23.13を動作させた環境を用いた．また，システムコールにはunameを用いている．それぞれの場合について，システムコールを1000回呼び出したときに要する時間を表1に示す．

表1 システムコールの処理時間

システムの使用	処理時間	増加率
なし	0.8032 ms	0.000
あり	0.8381 ms	0.043

本システムを用いることで，システムコールの処理時間が増加していることがわかる．増加時間は，約4.3％であるが，unameのような単純なシステムコールではなく，ファイルやデバイスの入出力をともなうようなシステムコールに対しては，十分に短い処理時間で識別のためのデータを取得できると考えられる．

また，本システムを用いると，システムコールに必要な処理時間が増加するだけではなく，識別のためのデータをカーネル内に保持するためのメモリが必要になる．このデータは，カーネルが保持するプロセスのアドレス空間を管理するデータ内に保持される．どの程度のメモリが本システム用に必要になるかは，何回分の履歴を保持するかによって決まる．基本的には，多くの履歴を保持し，それを識別処理に使用すると，識別精度の向上が期待できる．しかし，多くの履歴を保持した結果，使用可能なメモリが不足することは性能の低下を招くため，各システムにおいて，履歴の保持量を適切に設定することが必要である．実際にどの程度の履歴があれば，十分な精度で識別が可能になるかについては，次節で評価を行う．

4.2 識別の精度

初期動作からプロセスの使用メモリ量の特性をどの程度の精度で識別できるかを評価するための実験を行った．本実験では，3章と同様の環境を用いている．また，サポートベクターマシンには，libsvm-2.82.0[5] を用いた．

本実験の手順を以下に示す．

(1) 実際にプロセスを動作させ，初期動作時のメモリ使用の特性，および，初期動作完了後の使用メモリ量を取得
(2) (1)で得られたデータのうち無作為に選びだした半分を用いて学習を行う．

表 2　メモリ使用量の 5 数要約

	(a)	(b)	(c)
Min.	38	16	103
1st Qu.	154	51	228
Median	404	189	603
3rd Qu.	1960	555	2482
Max.	12415	5675	15845

(3) 学習で得られたモデルデータを用いて，残りの半分のプロセスの識別を行う．

手順 (1) では，1013 個のプロセスの特性を得ている．ただし，本実験でプロセス動作させた際，メモリの空き容量は十分にあり，スワップアウトは発生していない．プロセスのメモリ使用の特性情報としては，次の 3 つメモリ量を取得している．

(a) ファイルがマップされた領域のメモリ量
(b) ファイルがマップされていない領域のメモリ量
(c) ファイルがマップされた領域のメモリ量とされていない領域のメモリ量の合計

各プロセスが初期化処理終了後に実際に使用しているメモリ量は，(a)，(b)，(c) のそれぞれで，表 2 に示すように分布している．本システムでは，使用メモリ量の見積もりを行うことが目的であるが，使用メモリ量に応じ 4 つのクラスにプロセスを分類する．このとき，各クラスの使用メモリ量の閾値には，表 2 の値を用いた．すなわち，ファイルがマップされた領域のメモリ量の変化をメモリ使用特性のデータとする場合，154，404，1960 が閾値となる．手順 (1) では，このようにして定めた閾値をもとに，各プロセスがどのクラスに属するかを調べている．すなわち，手順 (1) の段階で，1013 個のプロセスすべてについて，使用メモリ量の履歴と属するクラスのデータが収集されたことになる．

手順 (2) では，収集したデータを識別器の学習に使用するものと，識別の精度を評価するための 2 つのグループに分けている．また，学習用グループのデータを用いて学習を行い，モデルデータを作成している．すなわち，使用メモリ量の履歴と，それが 4 つのクラスのうちどれに属するのかの情報を与え，サポートベクターを算出する．

手順 (3) では，学習に使用しなかった残りの半分の識別精度評価用グループのデータを用いて，識別精度を調べる．具体的には，使用メモリ量の履歴から，識別器によってどのクラスに属するかを識別し，その結果と実際にどのクラスに属するものであるかの比較を行う．実際のクラスと同一のクラスに識別されれば正解とし，他の 3 つのいずれかに識別された場合は不正解とした．

(a)，(b)，(c) のそれぞれについて，識別器への入力した履歴の数と識別が正しく行われた割合の関係は，図 7，図 8，図 9 のようになった．いずれの場合も，入力履歴数が増加すると精度が上がっている．入力履歴数が少ない場合精度が非常に低いが，入力履歴数が十分あれば，90 %程度の精度で正しく識別することができている．

(a) では，ファイルがマップされているメモリの量のみで識別を行っている．メモリにマップされるファイルは，アプリケーションの実行ファイルや使用している共有ライブラリのファイルである．したがって，この場合の使用メモリ量の変化は，実行ファイルのサイズやどのような共有ファイブラリを使用しているのかを反映したものとなる．すなわち，実行ファイルや使用する共有ライブラリの種類が識別処理に使用されている情報であるといえる．

(a) では，履歴数が 1 個であると，30 %弱の精度でしか正しく識別を行えていない．本実験では 4 つのクラスへの分類を行っているため，識別処理を無作為に行った場合，25 %の精度を期待できる．したがって，30 %弱の精度は，ほとんど識別が行われていない状態だといえる．しかし，入力履歴数が増加すると，それに連動して精度が上昇している．

(b) では，ファイルがマップされていないメモリの量のみで識別を行っている．ファイルがマップされていないメモリは，スタックやヒープに使用されるものである．これらのメモリの使用量の変化は，プロセスがどのように動作しているかに依存する．したがって，(b) では，プロセスの動作をもとに識別を行っていると考えることができる．

(a) と比較し，(b) では，入力履歴数が少ない場合，識別の精度が高い．例えば，入力履歴が 1 個の場合，精度は 50 %弱であり (a) に比べ高い値となっている．当然のことながら，50 %の精度は十分では

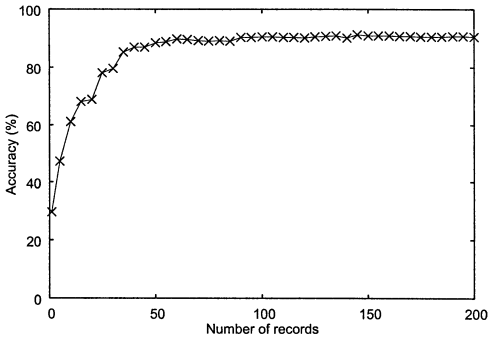

図 7 (a) での識別の精度

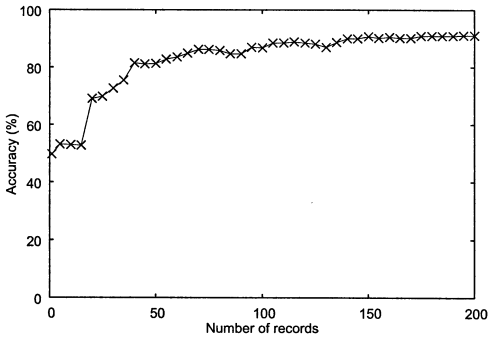

図 8 (b) での識別の精度

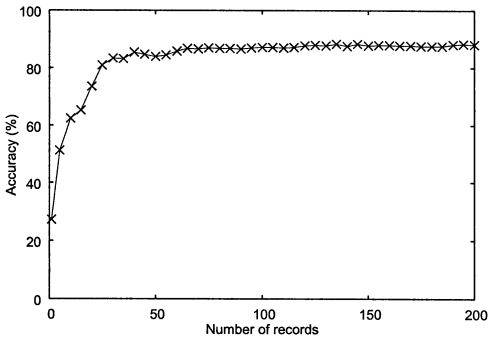

図 9 (c) での識別の精度

ないが，入力履歴数が増加すれば，(a) と同様に十分な精度で識別を行えるようになる．しかし，(a) に比べると，入力履歴数の増加に対する精度の上昇は緩やかである．

(c) では，(a) と (b) の両方の情報を使用して識別処理を行っている．すなわち，使用するファイルの情報とどのように動作しているかの両方の情報をもとに識別を行っている．(c) でも，履歴数が 100 個のとき 87.1 %の精度で識別を行えており，プロセスがどの程度のメモリを必要とするかを予測するのに，プロセスの初期動作の情報を利用することは有用であることが分かる．

5 おわりに

本稿では，プロセスの初期動作を観察することで得られる情報から，プロセスの特徴を識別する手法について述べた．また，本手法に基づいた，プロセスが初期化処理時に使用するメモリ量から将来使用するメモリ量を予測するシステムについても述べた．プロセスの初期動作からのプロセス特性の予測は，オペレーティングシステムの資源管理にとって有用である．特に，プロセスの動作によって資源割り当ての方法を切り替える既存の手法と組み合わせることで，より効率的な資源管理を実現できると考えられる．

参考文献

[1] Kim, J. M., Choi, J., Kim, J., Noh, S. H., Min, S. L., Cho, Y. and Kim, C.-S.: A Low-Overhead, High-Performance Unified Buffer Management Scheme That Exploits Sequential and Looping References, *OSDI*, pp. 119–134 (2000).

[2] Smaragdakis, Y., Kaplan, S. and Wilson, P.: EELRU: simple and effective adaptive page replacement, *SIGMETRICS '99: Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, New York, NY, USA, ACM, pp. 122–133 (1999).

[3] Nello Cristianini, John Shawe-Talylor 著, 大北剛 訳: サポートベクターマシン入門, 共立出版 (2005).

[4] 小野田崇: サポートベクターマシン, オーム社 (2007).

[5] Chih-Chung Chang and Chih-Jen Lin: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.