B-014　メモリアクセス特性を用いたメモリ管理ポリシ選択の自動化
Automatic Policy Selection for Memory Management based on Access Characteristics

川原 葵 † 小林 良岳 † 中山 健 † 前川 守 †
Aoi Kawahara Yoshitake Kobayashi Ken Nakayama Mamoru Maekawa

1. はじめに

大量のメモリを消費し, 様々なメモリアクセス特性を持つアプリケーションが複数同時実行される環境において, 単一のメモリ置換アルゴリズムでは著しいパフォーマンスの低下が起こる可能性がある. そこで, システム内に複数のメモリ置換アルゴリズムを用意し, 個々のアプリケーションに適応することが考えられる. しかしこの時, メモリ置換アルゴリズムの選択方法が問題となる.

現在までに, 複数のページングアルゴリズムを持ち, 個々のアプリケーションに対して個別のページングアルゴリズムを割り当てる事が可能な機構が作成されている [1]. しかし, この機構では手動でプロセスとページングアルゴリズムを結びつける設定をしなければならず, 動作中のアプリケーションのメモリアクセス特性も知ることができない.

そこで, 本研究ではメモリ置換アルゴリズムを自動選択するために, メモリアクセスを監視するシステムを実装し, アクセスパターンを検出する手法について考察を行う.

以降は 2 章でシステム概要について述べる.3 章ではシステムの評価を行い, 4 章で関連研究,5 章でまとめと今後の課題を述べる.

2. システム概要

システムの構成を図 1 に示す. 本システムはメモリアクセスの監視とログを記録する memory moniter, 取得したログを解析する access analyzer, 解析結果を元にページングアルゴリズムを切替える policy changer で構成される.

2.1 メモリアクセス監視 (memory monitor)

メモリアクセス監視はページフォールトの記録を行う. メモリモニタによる監視を行うにはまず監視対象であるプロセスのプロセス ID を登録する. これによって任意のプロセスについて, 監視を行う事を可能とする. 監視対象であるプロセスのページフォールト割り込みが検出された時に, ログ記録処理を呼び出す. この処理ではページフォールトが起きた時刻と論理アドレスを記録する. また, ログはタスクごとに保持する事が可能である. ログは一時的にカーネル領域に蓄えられ, 一定量蓄積された段階でユーザ領域に書き出される. 現在はファイルへ情報を出力するようになっている.

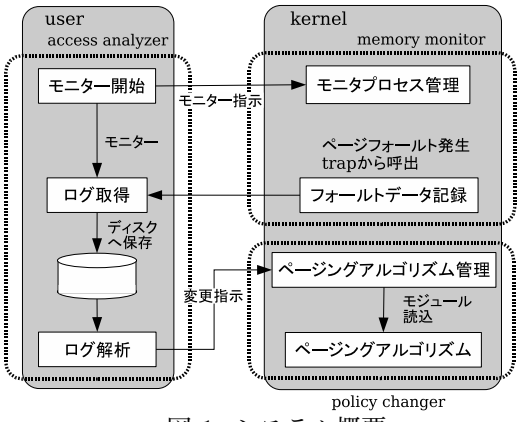

図 1: システム概要

2.2 ログ解析 (access analyzer)

ログ解析は記録されたデータを元にページングアルゴリズムを決定する. ページフォールトの発生回数が増え続けた時, 使用中のページングアルゴリズムが不適切であるとし, ログの解析を行う. 使用中のページングアルゴリズムが適切であるとしたプロセスについては記録したログを破棄する. またページングアルゴリズムを変更した場合も破棄する.

特性を解析する為には全てのメモリアクセスを記録する事が理想ではあるが, その記録および計算コストは膨大である. またページフォールトの発生したアドレスからメモリアクセス特性を類推する事は可能である [2]. 以上の理由からページフォールトのみの解析を対象とした.

2.3 ページングアルゴリズム管理 (policy changer)

動的に割り当てられるページングアルゴリズムをモジュールとして管理する. ログの解析の結果にしたがい, これらのモジュールはプロセスと関連づけられる. 典型的なメモリ参照特性を想定したメモリ管理ポリシは OS に最初から用意しておく. またユーザ (開発者, 使用者) が作成したメモリ管理ポリシを導入・削除する事も可能である.

3. 評価

本システムの目標はメモリアクセス特性を用いてメモリ管理ポリシを選択することである. そこで実際にアプリケーションを実行し, ページフォールトの発生領域を調査した. また本システムによるアプリケーションへの影響を調べるために, ログの記録にかかるオーバヘッドを計測した.

3.1 実装環境

実験は,PentiumIII 500MHz, メモリ 256MB の FreeBSD 4.8-RELEASE にメモリ監視機構を組み込んで行った. 但し, ページフォールトをより多く発生させるために, コンパイルオプションでシステムが利用可能な物理メモリを 64MB に制限してある.

3.2 ページフォールト監視

ログの取得によって得られたグラフを以下に示す. ここではユーザアプリケーションの代表的な機能を持つと考えられるソフトを用いた. 以下のグラフは横軸がモニター開始からの時間経過, 縦軸はページフォールトの起きたアドレスである.

†電気通信大学大学院情報システム学研究科情報システム設計学専攻

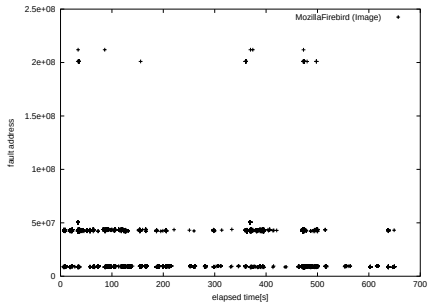

図 2: 画像 (jpeg) 主体のページ閲覧時

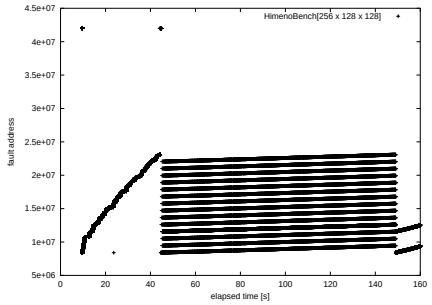

図 3: 姫野ベンチ実行時

Web ブラウズ (画像閲覧) Web ブラウザ MozillaFirebird を利用し, 画像主体のページを閲覧した時のメモリアクセス状況のグラフを図 2 に示す.

科学技術計算 行列計算によって処理速度を計る姫野ベンチを利用し, サイズが 256 x 128 x 128 である行列の計算を行った時のメモリアクセス状況のグラフを図 3 に示す. このグラフは全体の処理のうち特徴的な部分の詳細図である.

3.3 ログ記録オーバヘッド

ログデータの出力処理について, 処理にかかる時間を測定した (表 1). フォールトデータはカーネル領域からファイルまたはユーザ領域のバッファに記録される. データ量は一回の出力処理で処理するログ数である.

4. 考察

4.1 ページフォールト監視

実験からページフォールトは, アプリケーション起動直後に多く発生し, その後に機能特有のメモリアクセスパターンが現れる. またページフォールト発生量も機能ごとに特徴的な分布をしめす事が判った. ログ解析は起動中のフォールトデータを無視する事で, 誤りを減らす事ができる. またページフォールトの発生領域だけでなく発生回数も解析に利用できると考えられる.

ブラウザのようなアプリケーションでは表示している内容によってもページングの起こるパターンが変化する事が判った. よって常に動的に切替える様にすると頻繁にページングアルゴリズムの変更が起こってしまう。この問題を解決するためにアプリケーションのページフォールトの統計などを取りデータベース化する事でアルゴリズム変更を制御できるのではないかと考えられる.

表 1: データ出力にかかる時間

出力先	データ量 [個]	処理時間 [μ s]
FILE	80	396.6
	1000	402.3
buffer	80	4.42
	1000	4.41

4.2 ログ記録オーバヘッド

実験の結果から, 出力データのサイズによる処理時間の大きな差は無い事が判った. ログの記録処理において, ページフォールト発生量が多少にかかわらず, 他アプリケーションへの影響は少ない事が確認できた. また出力先がファイルの場合はユーザ領域バッファへの書き込みの 100 倍の時間がかかっており, ログ解析にはバッファ出力のみを使う必要がある事が判った.

5. 関連研究

APEX[3] は組み込み機器向けのメモリ管理システムである. ハードウェア的にメモリアクセスパターンを検出する事ができ, パターンを分類する. 分類に対し一つずつ専用のメモリモジュールを持ち, これを割り当てる事でページングアルゴリズムの切替えを行っている. この機構をメモリ量の大きなコンピュータ上で実現した場合, ハードウェアの性能がボトルネックになる可能性がある.

ACME[4] は複数のメモリ管理ポリシを使い, 効率の良いキャッシングを行う事を目的としている. しかし, 管理ポリシそのものを入れ替える訳ではなく, 全てのポリシに対してキャッシングの計算を行い統計を取った結果を用いている. この方法での問題点として, 全ポリシの計算を行う計算, 評価に使うデータの保存のスペースなどのコストが大きくなってしまう点があげられる.

6. まとめと今後の課題

本研究ではメモリ置換アルゴリズムを自動的に選択するために, メモリアクセスを監視する機構を実装し, さらに特徴を検出する手法について考察した. 今後はより詳細なログ解析を行い, 特徴を決定する機構を提案する.

参考文献

[1] 唐野 雅樹, 小林 良岳, 結城 理憲, 中山 健, 前川 守, 複数タスクの特徴に動的適合可能なマルチポリシメモリ管理システム, Swopp2001 情報処理学会研究報告, 2001-OS-88, pp. 99-106, 2001.

[2] 安田 裕, 小林 良岳, 中山 健, 前川 守, 最適なメモリ管理ポリシの自動割り当てに関する研究, SPA2004 日本ソフトウェア科学会, 2001-OS-88, pp. 99-106, 2001.

[3] Peter Grun, Nikil Dutt, Alex Nicolau, APEX:Access Pattern based Memory Architecture Exploration, ISSS'01, pp.25-32, 2001.

[4] Ismail Ari, Ahmed Amer, Robert Gramacy, Ethan L. Miller, Scott A. Brandt, and Darrell D. E. Long, ACME: adaptive caching using multiple experts, In Proceedings of the 2002 Workshop on Distributed Data and Structures, 2002.