Vol.2010-HPC-125 No.9
2010/6/17

# ユーザレベル実装 遠隔メモリページングシステムにおける ページ置換アルゴリズムの評価

斉藤 和広[†,*] 緑川 博子[†] 甲斐 宗徳[†]

筆者らはクラスタにおける遠隔マシン上のメモリを利用して大容量メモリを提供する遠隔メモリページングシステムが，OS とは独立のユーザレベルソフトウェアとして実装することで高速かつ安定した動作が得られることを示してきた．遠隔メモリを利用する上で，遠隔スワップは最もクリティカルな処理であるため，OS の仮想メモリと同様に効率のよいページ置換アルゴリズムが求められる．しかし,ユーザレベルソフトウェアでは従来の OS で利用されてきたメモリアクセス情報を利用するページ置換アルゴリズムをそのまま用いることが難しい．擬似的なメモリアクセス情報を用いることで実装可能ではあるが，このコストや性能は十分に評価されていない．そこで今回，擬似的なメモリアクセス情報ビットとして参照ビットと変更ビットをユーザレベルで実装，評価を行い，実用可能であることを示した．更に，これを利用したユーザレベル実装のページ置換アルゴリズムとして Clock，NRU，擬似 LRU と，メモリアクセス情報を利用しない低コストのシンプルアルゴリズム，スワップイン履歴を用いたページ置換アルゴリズム，ランダムアルゴリズム，FIFO を実装し，比較・評価を行った．

# Evaluation of Page Replacement Algorithms for User-level Remote Memory Paging System

Kazuhiro SAITO[†1] Hiroko MIDORIKAWA[†2] and Munenori KAI[†2]

The authors already revealed that a user-level remote memory paging system, which provides a larger size of memory beyond that of local physical memory by using remote memory distributed over cluster nodes, is higher performance and more stable than a kernel-level one. It requires an efficient page replacement algorithm because the remote page swap is critical in performance for using the remote memory. However, an user-level software can be hardly apply its page replacement algorithm using memory access information in OS. If it uses pseudo memory access information, it can be applied the efficient page replacement algorithm. But this cost and performance are not evaluated sufficiently. In this paper, we implement and evaluate pseudo memory access bits, and reveal that it can be used actually. Furthermore, we compare page replacement algorithms using them and not using, the Clock algorithm, the NRU and pseudo-LRU, Simple algorithm, page replacement algorithm using swap-in history, the Random algorithm and the FIFO.

## 1. はじめに

### 1.1 背景

64bitOS の普及により，飛躍的に大きなアドレス空間が利用可能となり，大容量データを扱う様々な応用が容易に実行できるようになってきた．OS における仮想メモリ機構では物理メモリサイズを超えるデータを扱う場合に，ローカルハードディスク上のスワップ領域にページ単位でスワップすることで，物理メモリサイズを超えた仮想メモリを実現している．しかし近年，ローカルハードディスクのデータ転送速度を越える通信性能を持つネットワークが出現し，遠隔マシンのメモリを仮想メモリとして利用する遠隔メモリページングの研究がなされるようになってきた[1][2].

筆者らは，クラスタにおける遠隔マシン上のメモリを利用し，ローカル物理メモリサイズに制限されずに大容量メモリを提供する分散大容量メモリシステム DLM(Distributed Large Memory)を構築，評価してきた[3]．DLM は大容量データを扱う逐次処理のためのシステムで，並列化の困難なアプリケーションや，並列・分散処理の専門知識を持たないユーザにとって恩恵がある．さらに DLM は，OS スワップシステムに組み込む他の多くのカーネルレベル実装の遠隔メモリページング手法とは異なり，OS とは独立のユーザレベルソフトウェアとして実装したことで，高速かつ安定した動作が得られることを示してきた．

遠隔メモリページングにおいてクリティカルな処理となるのは遠隔スワップである．そのため，OS の仮想メモリ機構のスワップと同様に，遠隔スワップの回数を減らすためにも計算ノードのメモリに配置しておくページの選択が非常に重要となり，効率のよいページ置換アルゴリズムが求められる．従来から OS 仮想メモリにおけるページ置換アルゴリズムは非常に多くの研究がなされてきていて，これらのページ置換アルゴリズムのほとんどは OS で利用されることを前提として考えられてきている．例えば，理論上最適なアルゴリズム OPT に最も近いとされる LRU（Least Recently Used）や，FIFO を改良した Clock などのページ置換アルゴリズムは，メモリアクセスの時刻やアクセスビットを用いてページへのアクセスが最近かそうでないかを判別しているが，これらのメモリアクセス情報は，メモリ管理機構 MMU や OS カーネルか

[†]成蹊大学工学研究科情報処理専攻
Graduate School of Engineering, Seikei University
[*]現在，KDDI 株式会社
Presently with KDDI corporation

ら提供されている[5]. ユーザレベル実装のシステムではこれらを直接利用することができないため，これらを利用するページ置換アルゴリズムをそのままでは利用できない．そのため，ユーザレベルでメモリアクセス情報を用いるページ置換アルゴリズムを実装するためには，擬似的にメモリアクセス情報を実装する必要があるが，これを実装しないアルゴリズムよりも高コストになる．しかしこのようなユーザレベル実装のシステムにおいて，擬似的なメモリアクセス情報を用いたアルゴリズムと，低コストで単純なアルゴリズムの性能評価は十分にされていない．

そこで本稿では，低コストのページ置換アルゴリズムとして，これまで DLM で利用してきたシンプルアルゴリズム[3](Simple)とスワップイン履歴を用いたページ置換アルゴリズム[4](Swap-in history)，更に新たにランダムアルゴリズム(Random)と FIFO を，また擬似的なメモリアクセス情報を利用するページ置換アルゴリズムとして Clock，NRU，擬似 LRU [5]をユーザレベルソフトウェアである DLM に実装することで比較・性能評価を行う．

### 1.2 関連研究

多くの遠隔メモリページングシステムはカーネルレベル実装であり，OS のスワップ機構を利用するため，OS のスワップ機構で用いられるページ置換アルゴリズムをそのまま利用している．また，これらの研究では OS のスワップ機構のページ置換アルゴリズムを改良しようとする試みもされていない．

Teramem[1]はカーネルレベル実装ではあるが, OS のスワップ機構とは独立にローダブルモジュールとして実装しているため,自由にページ置換アルゴリズムを設計することができる．論文[1]では FIFO と擬似 LRU を Teramem に実装し，GNU sort で比較し，後者が優れているとしている．

ユーザレベルソフトウェアである JumboMem[2]は擬似 NRU を利用していて，最近フェッチされたページ，変更されたページをローカルに維持しようとするポリシーを用いている．しかし実装方式や他のアルゴリズムとの性能比較は言及されていない．

## 2. 評価するユーザレベル実装ページ置換アルゴリズム

本章では，ユーザレベルで実装するページ置換アルゴリズムとして，メモリアクセス情報を用いない低コストのページ置換アルゴリズムと，メモリアクセス情報を利用するアルゴリズムを擬似的なメモリアクセス情報の実装とともに述べる．なお，DLM のメモリ管理の単位を DLM ページと呼ぶが，以降これをページと呼ぶこととする．

### 2.1 メモリアクセス情報を用いない低コストのページ置換アルゴリズム

メモリアクセス情報を利用しないページ置換アルゴリズムとして, DLM で利用する Simple と Swap-in history を，また今回新たに Random と FIFO の計４つを実装する．

Simple はスワップアウトページ探索時に，ページ単位で管理されている DLM のアドレス空間の最初のページから探していき，最初に発見した計算ノードにマップされたページをスワップアウトページとして選択する．次回以降は前回選択したページからスタートして同様に探索するアルゴリズムである. Simple は非常に低コストであり，配列アクセス等の連続アクセス時には効率的な選択を行う．しかし，ランダムアクセス等の非連続アクセスや，連続アクセスの開始のタイミングによっては非効率な選択を行ってしまう可能性がある．

スワップイン履歴を用いた Swap-in history は，各ページが持つスワップインの履歴を利用することで, Simple の非効率的な選択を回避するアルゴリズムである. これは，スワップを発生させたページの前回のスワップイン後にスワップインしたページは，今回のスワップインでもすぐに必要となる可能性が高いページであるという仮定に基づいている．これを実現するために，スワップ発生時に各ページがそれぞれのスワップイン履歴を保持し，前回のスワップインの直後にスワップインしたページはスワップアウトページ選択しない方式を取っている．

Random は，ローカルにあるページから乱数によってスワップアウトページを選択するアルゴリズムである．

FIFO は最初にスワップインしたページをスワップアウトページとして選択する. そのために，スワップインした順に並べたページのキューを利用した．

### 2.2 擬似メモリアクセス情報の実装

ユーザレベル実装では，アルゴリズムと共にメモリアクセス情報である参照ビットや変更ビットも擬似的に実装しなければならない．これには，OS のメモリ管理で利用されるページ単位のアクセス保護（mprotect）を利用する．擬似参照ビットのセットは，メモリ割り当て時に Read/Write を不可にし，アクセス時に SIGSEGV ハンドラで擬似参照ビットを 1 にし Read/Write 可にすることで実現が可能である．擬似変更ビットも同様で，SIGSEGV ハンドラ内でアクセスの読み書きを調べ，読み込みなら擬似変更ビットは 0 のままでページを Read 可に，書き込みなら擬似変更ビットを 1 にしてページを Read/Write 可にする．

擬似参照ビットには定期的なクリアが必要となる．Clock のように，スワップアウトページ選択時に探索とともにクリアしていくアルゴリズムもあれば, NRU のように選択時とは別の定期的なクリアが必要な場合もある．このようなスワップアウトページ選択と独立した疑似参照ビットの定期的クリアには，いくつかの方針が考えられ，ここでは以下の２つを提案する．第１に，一定時間が経過したときにクリアする「タイマー方式」である．ユーザレベルでは SIGALRM を利用して実現可能である．タイマー方式はスワップの頻度などのアプリケーションの違いによるクリアのタイミングの変化が起こらないという特徴がある．しかし，DLM ではユーザプログラムを実行するスレッド（計算スレッド）でのクリア処理となるため，シグナル割り込みによって計算が中断してしまう．第２に，スワップが一定回数発生したときにクリアする「ス

ワップ回数方式」である．これは DLM では計算スレッドとは別のスレッド（通信スレッド）で処理を行うことが可能なので，計算と平行して処理を行うことが出来る．また，スワップ頻度によってクリア間隔の実際の時間が変化するという特徴を持つ．

### 2.3 擬似メモリアクセス情報を用いるページ置換アルゴリズム

前述した擬似メモリアクセス情報ビットを用いて, 新たに Clock, NRU, 擬似 LRU を DLM に実装した. 以下, ユーザレベル実装の Clock を U-Clock, 同様に NRU を U-NRU, 擬似 LRU を U-pLRU と呼ぶこととする．

U-Clock は FIFO に参照ビットを付加することで, 最近アクセスしていないページを優先するよう改良したアルゴリズムである．本実装の擬似参照ビットは，メモリ割当て時に 0 で設定され，スワップアウトページ選択時にスワップイン順にページの擬似参照ビットを見て，1 のページはクリアし，0 のページはスワップアウトされる．

U-NRU は擬似参照ビット R と擬似変更ビット M を用いてページを４つに分類する．

- クラス１：R=0, M=0（最近のアクセスがなく，変更もされていない）
- クラス２：R=0, M=1（最近のアクセスはないが，変更されている）
- クラス３：R=1, M=0（最近アクセスされたが，変更はされていない）
- クラス４：R=1, M=1（最近アクセスされて，変更もされている）

スワップアウト選択時に上記のクラスを上から順に見ていき，ページの存在するクラスの中からランダムに選択する．また今回，擬似参照ビットの定期的なクリアの方式別に２つの U-NRU を実装し，スワップ回数方式のものは U-NRUs，タイマー方式のものは U-NRUt と呼ぶことにする．

U-pLRU は Linux kernel2.6.11 のページフレーム回収機構で採用されている擬似 LRU[6]を参考にしていて，２本の LRU リスト（アクティブリストと非アクティブリスト）と擬似参照ビットを利用する．スワップアウトページは非アクティブリストの先頭から見ていき，擬似参照ビット 1 のページはアクティブリストへ移動し，0 のページを選択する．このとき非アクティブリストが空だと，アクティブリストのページを非アクティブリストへ移動し，擬似参照ビットをクリアする．

## 3. 性能評価

### 3.1 評価環境

様々なページ置換アルゴリズムを実装した DLM を，それぞれ複数のアプリケーションに適用して評価した．アプリケーション実行に利用した環境は，表 1 の東京大学情報基盤センターの T2K[10]で，10Gbit/s のネットワーク（IP/Myrinet）でソケットを利用した．なお計測には計算ノード１台とメモリサーバ１台の計２台を用いた．

用いたアプリケーションは，姫野ベンチマーク C 言語版[7]の ELARGE サイズ（513x513x1025 サイズ，15GB），Cluster3.0[8]，NAS Parallel Benchmark (NPB)の C 言語版である NPB2.3-omni-C[9]の最大クラスである C の IS（1.6GB），FT（7GB），CG（1.2GB），MG（3.6GB），SP（1.3GB）である．Cluster3.0 は階層クラスタリングで遺伝子解析を行うアプリケーションで，メモリの動的割り当て・解放を多く繰り返す特徴がある．今回はサンプルとしてある遺伝子データの demo.txt を用いた．

計測は，各アプリケーションにおいて，様々なローカルメモリ率（プログラム全体で利用する全 DLM データのメモリサイズに対する計算ノードの利用メモリサイズ）別に実行した．なお，DLM ページサイズは 1MB を用いている．

表 1　東大 T2K の実験環境

| | Calculation Node |
|---|---|
| machine | HP ML150 G2 |
| CPU | Xeon 2.8GHz x 2CPU HyperThread |
| memory | 1GB |
| Cache | L2 : 1MB/CPU |
| OS | Linux kernel2.6.20-1.2320.fc5 x86_64 |
| NIC | Broadcom 5721 PCI-Express Gigabit NIC |
| Network | 1GbitEthernet |

### 3.2 擬似メモリアクセス情報ビットの評価

#### 3.2.1 擬似メモリアクセス情報ビットのオーバヘッド

U-Clock と U-NRUs，U-NRUt，U-pLRU は，擬似ビットのセットが必要となる．この処理には SIGSEGV ハンドラ呼び出しと mprotect 関数の呼び出し，ページ状態の変更（リスト移動等）がある．これらのうち前者２つの処理にかかる時間を，シグナルハンドラ内で mprotect 関数を呼び出すだけのテストプログラムを用いて計測した．これには，R/W 不可の 100MB のデータを用意し，1MB 単位でアクセスすることで SIGSEGV ハンドラ呼び出し時間と mprotect 関数の処理時間を計測する．その結果から出した１回の SIGSEGV ハンドラ呼び出し時間は 14 $\mu$ s，mprotect 関数の処理時間は 3 $\mu$ s と非常に短い時間であることがわかった．

#### 3.2.2 ローカルメモリ率別の時間成分

姫野ベンチマークの ELARGE で, タイマー方式である U-NRUt をクリア間隔 500ms で設定した DLM を用いて，ローカルメモリ率を様々に変えて実行した．この時の各実行時間における成分を図 1 に示す. 時間成分は擬似ビットクリア総時間(clear bit)，擬似ビットセット総時間（set bit），遠隔スワップに要する総時間（swap），その他の計算時間（calc）となっている．なお,擬似ビットセット・クリアと遠隔スワップの際に発生するシグナルハンドラ呼び出し時間はその他の計算時間に含まれている．図 1 のように，ローカルメモリ率が下がると，ローカルマシンが保持するページ数が少なくなるため，遠隔スワップの頻度が高くなり，遠隔スワップ処理の占める割合も大きくなる．また擬似参照ビットの処理は，ローカルメモリ率の減少とともにセット・クリアするページ数が少なくなるため，全体に占める割合が非常に少なくなる．

### 3.2.3 アプリケーション別の時間成分

時間成分の割合はアプリケーションによっても大きく異なり，図 2 はその一例で，ほぼ同じローカルメモリ率における U-NRUt(500ms)の姫野ベンチマーク, NPB FT.C, Cluster3.0 の時間成分である. FT.C は遠隔スワップが占める割合が少ないが, Cluster3.0 は実行時間の多くが遠隔スワップで占められていることがわかる.

### 3.2.4 スワップ回数方式のクリア間隔

スワップ回数方式の U-NRUs は，遠隔スワップの頻度によって擬似参照ビットのクリア間隔が変化する．例えばスワップ頻度が高まるとクリア間隔が短くなる．これは，アプリケーション毎に，また同じアプリケーション内でも実行フェーズの変化によってクリア間隔が変化する．図 3 は姫野ベンチマークにおける U-NRUs のスワップ 10 回，50 回，100 回の実際のクリア間隔の平均時間で，各スワップ回数は実質的に図のようなクリア間隔となり，ある程度タイマー方式と対応付けることができる．

### 3.2.5 ページ置換アルゴリズム別の時間成分

図 4 は，4 つのアルゴリズム（Simple，Swap-in history，U-NRUs のスワップ 50 回，U-NRUt の 500ms）の時間成分で，NPB FT.C のローカルメモリ率 77%と NPB SP.C のローカルメモリ率 55%で実行した結果である．なお，U-NRUs は擬似参照ビットクリアを計算と並列に実行するため，グラフのアプリケーション実行時間の中にクリア時間が含まれない．もしユーザプログラムを実行しているスレッドにクリア待ち時間が発生した場合，その待ち時間は擬似ビットセットの総時間に加算されている．

図 4(a)が NPB FT.C の結果で, 擬似メモリアクセス情報を用いる U-NRUs と U-NRUt は遠隔スワップの処理時間を，アクセス情報を用いない Simple と Swap-in history の半分以下まで減少させている．しかしアプリケーション実行全体から見ると，遠隔スワップが占める割合が少ないため短縮した時間も短く，擬似ビットセット・クリアの時間も無視できない大きさとなる．このため，実行時間自体は Swap-in history とあまり変わらない結果となった．また，擬似メモリアクセス情報を用いるアルゴリズムは，用いないものに比べその他の計算時間（calc）も長くなってしまっている．これは擬似ビット処理に用いる SIGSEGV ハンドラ呼び出し時のユーザモードとカーネルモードの切り替えによるオーバヘッドや，擬似ビット処理時のページ状態の変更等によるキャッシュ効果の低下が原因と考えられる．なお，FT.C における U-NRUs と U-NRUt の遠隔スワップは約 4 万回であるのに対し，擬似ビットセットは U-NRUs では約 18 万ページ，U-NRUt では約 54 万ページと多くの SIGSEGV ハンドラ呼び出しが行われている．図 4 (b)の SP.C はこれによってその他の計算時間が顕著に長く，遠隔スワップの処理時間は U-NRUs と U-NRUt が Simple より少し短くなっているが，その他の計算時間がより長くなっている影響で，全体の実行時間は Simple よりも長くなっている．

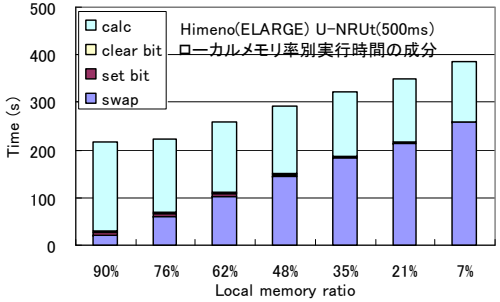



図 1　姫野ベンチマークのローカルメモリ率別時間成分

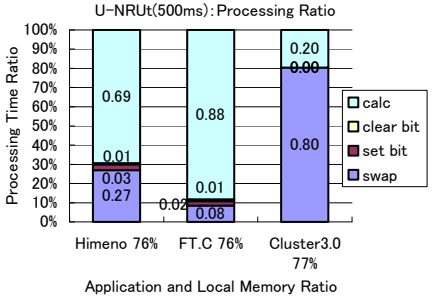



図 2　U-NRUt の各アプリケーションにおける時間成分

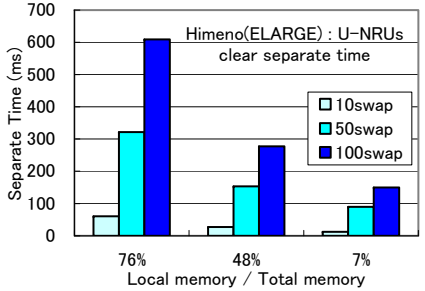
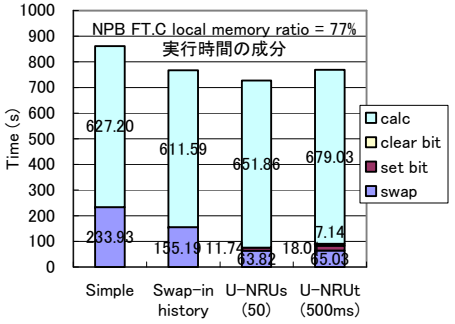
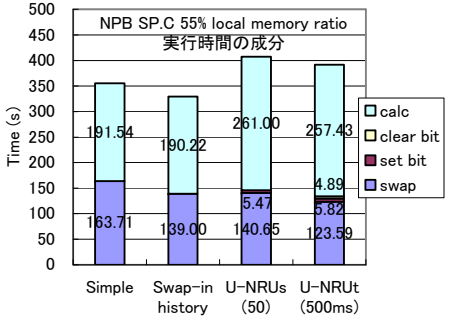



図 3　姫野ベンチマークにおける U-NRUs の平均クリア間隔




(a) NPB FT.C の実行結果




(b) NPB SP.C の実行結果

図 4　アルゴリズム別時間成分

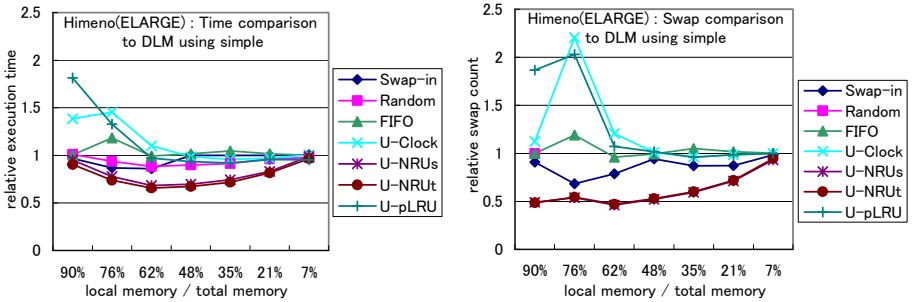
### 3.3 ユーザレベル実装ページ置換アルゴリズムの評価

今回評価に用いたページ置換アルゴリズムは８つで，Simple，Swap-in history，Random，FIFO，U-Clock，U-NRUs，U-NRUt, U-pLRU を DLM に実装し，前述の７つのアプリケーションに適用して実行した．結果は，Simple を基準とした相対値で表していて，1 より小さければ性能が良い．

図 5 は姫野ベンチマークで８つのアルゴリズムをローカルメモリ率別に実行した結果で，各ローカルメモリ率の Simple に対する相対値を表している．(a)は実行時間の相対値で，(b)が遠隔スワップ回数の相対値である．なお擬似参照ビットのクリア間隔は，U-NRUs はスワップ 100 回，U-NRUt は 1000ms で計測した．基本的にローカルメモリ率が大きいほど Simple との遠隔スワップ回数の相対値が大きい．これは，ローカルメモリ率が大きいと当然ローカルにあるページ数が多くなるため，スワップアウトページ選択の幅が増え，アルゴリズムの差が大きくでるためである．しかし実際の遠隔スワップ回数は少ないため，相対値の差ほどの変化がなく，実行時間は遠隔スワップ回数ほどの差は生まれない．反対にローカルメモリ率が小さいとスワップアウトページ選択の幅が少なくなり，アルゴリズムの差が小さくなる．このため,以後はこの差が比較的大きく出た中程度のローカルメモリ率に焦点を絞って評価する．

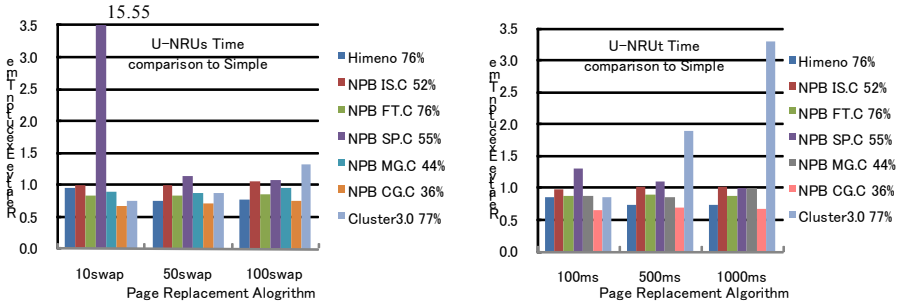
次に，U-NRU のスワップ回数方式とタイマー方式のクリア間隔を変えて，様々なアプリケーションで実行したときの評価を行った．図 6 は，U-NRU のクリア間隔別に６つのアプリケーションで実行したときの Simple に対する実行時間の相対値をグラフにしたものである．(a)はスワップ回数方式である U-NRUs を，クリア間隔のスワップ 10 回，50 回，100 回で実行した結果で，(b)はタイマー方式である U-NRUt を，クリア間隔を 100ms，500ms，1000ms に変えて実行した結果である．また,アプリケーションの横にあるパーセンテージはローカルメモリ率である．この結果から，アプリケーションによってクリア間隔を短くしたほうが性能の良いものと，逆にクリア間隔が長いほうが良いものとあることがわかる．特に顕著なのが NPB SP.C と Cluster3.0 で，SP.C はスワップ回数方式の 10 回で 15.55 倍と実行時間が激増しているが, 100 回では Simple とほぼ変わらない実行時間となった．また Cluster3.0 はタイマー方式の 1000ms で 3.8 倍と大きくなっているが，100ms とクリア間隔を短くするとむしろ Simple よりも性能が良くなる．このように，U-NRU はクリア間隔をアプリケーションによって変えることで性能悪化を抑え，高い性能を出すことが出来るが，アプリケーション毎にパラメータチューニングが必要となるという欠点がある．

図 6 から，どのアプリケーションでも比較的性能が高かったクリア間隔として，U-NRUs は 50 回，U-NRUt は 100ms をパラメータとして設定し，８つのページ置換アルゴリズムで７つのアプリケーションを実行したときの結果を図 7 に示す．なお(a)が Simple に対する実行時間の相対値，(b)が遠隔スワップ回数の相対値である．Swap-in history は今回用いた全てのアプリケーションで Simple よりも性能が向上していて，安定性が高い．更にパラメータチューニングも必要がないことからも，実用性が高いと言える．Random は適当にスワップアウトページを選んではいるが，アプリケーションによっては最近ページにアクセスしたかどうかが重要ではない場合や，時間的もしくは空間的ローカリティがない場合などがあり,このような場合には，アクセスビットや連続性を利用する場合よりもむしろ性能が向上する．今回用いたアプリケーションでは Cluster3.0 や NPB CG.C がこれに該当している．しかしそうでない場合，例えば NPB MG.C のように，性能が悪化する場合もある．FIFO は比較的安定した結果を出しているが，Simple や他のアルゴリズムに比べ性能が低い．U-Clock は FIFO に擬似参照ビットを付与することでアルゴリズムの質は向上していて，NPB FT.C や SP.C，Cluster3.0 では遠隔スワップ回数が大きく減少している．しかし擬似参照ビット処理によるオーバヘッド等により，実行時間は FIFO や Simple とあまり変わらない結果で，むしろ姫野ベンチマークのように悪化しているものもある．U-NRU の２つは，比較的性能が良く，姫野ベンチマークや NPB FT.C では Swap-in history よりも性能が良い．しかし，NPB SP.C のように性能が悪化する場合もあり，安定性には欠ける．またクリア方針の違いによっても性能は異なり，姫野ベンチマークの場合はスワップ回数方式である U-NRUs のほうが高い性能を出しているが, Cluster3.0 の場合はタイマー方式である U-NRUt のほうが良い．この結果や上で述べたようにパラメータチューニングが実用上難しいと考えられる．U-pLRU は NPB MG.C や FT.C，Cluster3.0 などでは比較的性能が良いが，姫野ベンチマークや NPB IS.C，CG.C では性能が悪化している．今回は実装方法としてかなり簡単化してアクセス状態を２段階にしてしまったために，あまりアルゴリズムの良さが引き出せなかったためだと考えられ，U-pLRU に関しては改良の余地がある．



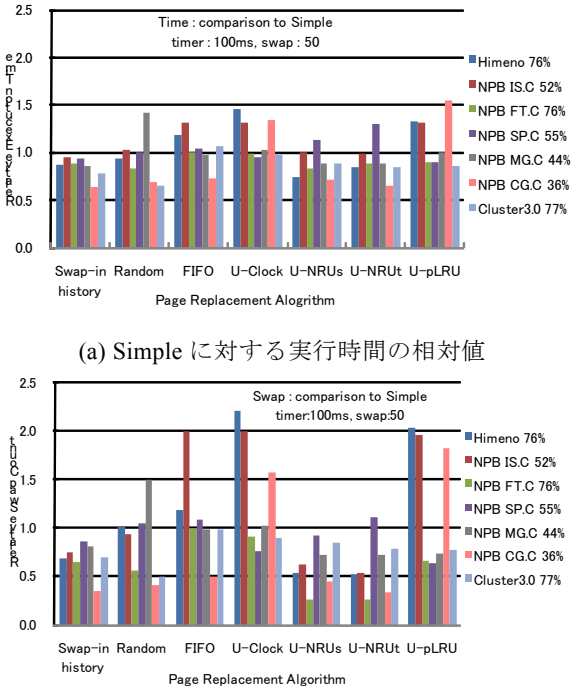

(a) 実行時間の相対値　　(b) 遠隔スワップ回数の相対値

図 5　姫野ベンチマークにおける様々なアルゴリズムの実行結果




(a) スワップ回数方式 U-NRUs　　(b)タイマー方式 U-NRUt

図 6　U-NRU のクリア間隔別実行時間の相対値




(a) Simple に対する実行時間の相対値

(b) Simple に対する遠隔スワップ回数の相対値

図 7　様々なページ置換アルゴリズムにおける実行結果

## 4.　おわりに

今回，ユーザレベルソフトウェアで，OS やハードウェアとは別の擬似的なメモリアクセス情報として擬似参照ビットと擬似変更ビットを実装し，さほど大きなオーバヘッドではなく実用レベルで利用できることがわかった．また擬似参照ビットのクリア方針としてスワップ回数方式とタイマー方式を提案した．それぞれに特徴があり，アプリケーションによってはクリア方針を変更することで性能が向上することもあった．

更にこの擬似メモリアクセス情報ビットを用いて，ユーザレベルソフトウェアにおける様々なページ置換アルゴリズムを実装し性能を明らかにし，これらが実用可能であることを示した．今回新たに実装したアルゴリズムは，ランダムアルゴリズム，FIFO，Clock，NRU，擬似 LRU で，特に NRU は擬似参照ビットのクリア方式別に２種類実装した．この中でも特にユーザレベル実装の NRU の性能が高かったが,実用上パラメータ設定が難しく，安定性に欠けることがわかった．一方メモリアクセス情報を用いない低コストのスワップイン履歴を用いたページ置換アルゴリズム Swap-in history は他のアルゴリズムに対し安定して比較的高い性能が出ることがわかった．

**謝辞**　この研究の一部は，文科省戦略的研究基盤形成支援事業，及び科研費基盤研究（C）（No.21500062）「大規模データ処理のための高速仮想メモリシステムの研究」の助成を受けています．

## 参考文献

1) 山本，石川，“テラスケールコンピューティングのための遠隔スワップシステム Teramem”，情処論文誌 ACS Vol. 2, No. 3 , pp.121-126 (2009, 9)
2) S. Pakin and G. Johnson, “Performance Analysis of a User-level Memory Server”, IEEE International Conference on Cluster Computing, pp.249-258 (2007)
3) 緑川，黒川，姫野，“遠隔メモリを利用する分散大容量メモリシステム DLM の設計と 10GbEthernet における初期性能評価”，情処論文誌 ACS, Vol.2, No.4, pp.15-36 (2009.12)
4) 斉藤，緑川，甲斐，"遠隔メモリページングにおけるスワップイン履歴を用いたページ置換アルゴリズムの初期評価"，情処学会，HPC 研究会 Vol.2009-HPC-120, No.8, pp.1-6 (2009, 6)
5) William Stallings, “Operating Systems Internal and Design Principles Fifth Edition”, Person Education Inc.（2005）
6) D. P. Bovet and M. Cesati, “Understanding the LINUX KERNEL”, O’Reilly Media Inc.,（2006）
7) (2010) Himeno Benchmark web site [Online]. http://accc.riken.jp/HPC/HimenoBMT/index.html
8) (2010) Cluster3.0 web site [Online]. http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/
9) (2010) NPB2.3-omni-C web site [Online]. http://phase.hpcc.jp/Omni/benchmarks/NPB/index.html
10) (2010) HA8000 クラスタシステム [Online]. http://www.cc.u-tokyo.ac.jp/ha8000/