遠隔メモリアクセスのためのスワップページサイズ自動調整機構

緑川 博子*[1]，内山 丞*[2]

Adaptive Page Size Control for Remote Page Swapping

Hiroko MIDORIKAWA [*1], Joe UCHIYAMA [*2]

ABSTRACT : An adaptive page size control methodology is proposed for remote memory paging. It estimates a working set size and changes page size dynamically and adaptively to each iterative processing part of an application during it is running. It is highly effective for iterative applications with various memory access patterns.

Keywords : Cluster Computing, Remote Memory, Distributed Memory, paging, page swap, page size, working set

（Received March 29, 2012）

1. はじめに

筆者らは，ユーザ逐次プログラムがローカルホストの物理メモリサイズの制約を受けずに大容量のメモリを利用できるようにするため，クラスタ上で巨大なメモリを仮想的に逐次プログラムに提供する分散大容量メモリシステム(DLM : Distributed Large Memory)[1][2][3]を開発している。DLM では計算に必要なデータが遠隔ノードにある場合，MPI 通信等を利用して遠隔ノードとのページスワップを行う。これにより，ローカルメモリサイズを超える大きなサイズのメモリを仮想的に利用することができ，クラスタなどの複数ノードの遠隔メモリを利用して巨大な仮想メモリがあるかのようにして，大きなデータを扱う逐次プログラムをほぼそのままのコードで実行させることができる[4]．DLM による実行は，ローカルメモリだけを用いた実行に比べ，実行時間は遅くなるものの，従来のローカルハードディスクにスワップ領域を確保して実現される OS の仮想メモリシステムを利用した場合に比べ，非常に高速に実行ができることが確認されている[1].

このようなシステムでは，ローカルメモリサイズ，応用プログラムのアクセス局所性に関連して, DLM スワップページサイズ（以降ページサイズ）が性能に大きく影響する事がある。DLM システムでは，通常，通信時の転送サイズは大きい方が一度の通信でより多くのデータを送ることができて効率が良い[1]。しかし，ある特定の応用において，ローカルメモリサイズが非常に小さい場合に，ページサイズとの組み合わせによって処理性能が急激に低下するという現象が確認されている[2]。DLM システムではユーザがプログラム実行時に任意にページサイズを決定する事ができるが，この現象は応用の種類やローカルメモリサイズによって状況が異なるため，ユーザが事前に適切なページサイズをあらかじめ決定する事が難しい。

そこで，本研究では，科学技術計算における数値計算などでよくみられる繰り返し処理を持つ応用を対象に，DLM システムにおけるページサイズと各種応用プログラムの実行時間との関係や影響の調査を行った。その結果を元に，応用プログラムの繰り返し処理部分に含まれるワーキングセットサイズに注目して，実行中にページサイズを動的に変更し最適化するページサイズ自動調整機構を考案した．この機構を DLM システムに実装し，その評価を行った。

2. ページサイズの応用実行性能への影響

本研究での実験は全て，表 1 に示す東京大学の T2K-Tokyo, HA8000 クラスタシステム[5]のノード間ネットワーク 2.5GB/s のクラスタを使用して行っている。

*1：情報科学科 助教 (midori@st.seikei.ac.jp)

*2：理工学研究科理工学専攻情報科学コース修士学生

表 1　HA8000 クラスタシステム

	T2K Open Supercomputer, HA8000
CPU	AMD QuadCore Opteron 8356(2.3GHz)4CPU/ node
Memory	32GB/node (936 nodes)，128GB/node (16nodes)
Cache	L2 : 2MB/CPU (512KB/Core)，L3 : 2MB/CPU
Network	Myrinet-10G x 4, (5GB/s full-duplex) bonding2 Myrinet-10G x 2, (2.5GB/s full-duplex) bonding4
OS	Linux kernel 2.6.18-53.1.19.el5 x86_64
Compiler	gcc version 4.1.2 20070626，Hitachi Optimizing C mpicc for 1.2.7
MPI Lib	MPICH-MX (MPI 1.2)

2.1 応用プログラム実行時間とページサイズ

DLM システムを使用する各種応用実行性能とページサイズには関連性がある。その例を図 1 に示す。

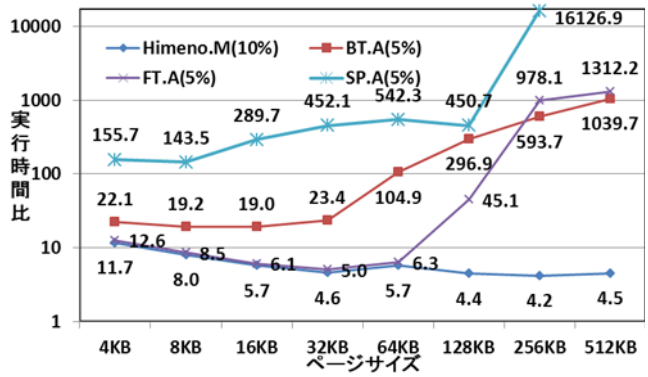

図 1 ページサイズによる実行時間への影響

図 1 は，姫野ベンチマーク[6]（サイズ M，ローカルメモリ率 10%），NAS 並列ベンチマーク NPB[7]の BT（クラス A，ローカルメモリ率 5%），NPB の SP（クラス A，ローカルメモリ率 5%），NPB の FT（クラス A，ローカルメモリ率 5%）の 4 種のベンチマークについての相対実行時間とページサイズの関係を示す。ここでいう**ローカルメモリ率**とは，応用プログラムが使用する全メモリのうちの何%が，ローカルメモリにあるかを示しており，残りの割合（%）は遠隔メモリにデータがおかれていることを示す．すなわち，ローカルメモリ率 5%とは，プログラムの使用する全メモリのうちの 5%のみがローカルメモリにあり，残りの 95%は遠隔メモリを利用して遠隔ページングを行いながら，仮想的に大きなメモリがあるかのように処理を行うということを意味する．つまり，ローカルメモリ率 5%の実行とは，ローカルメモリサイズの 20 倍のサイズの仮想メモリをプログラムから利用していることを意味する．

図 1 のグラフの横軸は計測時のページサイズを示し，縦軸は DLM を使用しないで，ローカルメモリのみを用いた通常実行(ローカルメモリ率 100%)の場合の実行時間を基準にした相対実行時間である。ここでは計測時間短縮のために応用プログラムは小規模問題サイズを用いて実験を行っているが，ローカルメモリ率などの条件がそろったときには，大規模サイズの問題の場合にも同様の現象が起こる。

姫野ベンチマークでは，通常実行に比べ，ページサイズ 512KB で 4.5 倍，ページサイズ 4KB では 11.7 倍の実行時間がかかり，ページサイズの大きい方が相対的に性能がよいことがわかる。この結果は過去の研究[1]においても示されており，大多数の応用では，一般に大きなページサイズのほうが，実行時間が短い傾向が見られる。

これに対し，一部の NAS 並列ベンチマーク（NPB）で，ローカルメモリ率が低い場合などに，全く逆の傾向が見られることがある。たとえば，ページサイズ 4KB では，BT（クラス A，ローカルメモリ率 5%）の実行時間比は 22.1，FT（クラス A，ローカルメモリ率 5%）の実行時間比は 12.6，SP（クラス A, ローカルメモリ率 5%）の実行時間比は 155.7 になってしまう．ページサイズ 512KB では，BT が 1039.7，FT が 1312.2 へ変化し，ページサイズ 256KB で SP は 16126.9 まで実行時間が増加してしまう。

図 1 で示したような実行時間の急激な増加は，どのような応用でも，ローカルメモリ率が低ければいつでも発生するというものではない。しかし，ローカルメモリサイズが小さい場合に，一部の応用プログラムで確認されており，この現象による実行時間の増加量は無視できないほどの非常に大きなものになることがある．

またページサイズを変化させる事で実行時間についてある特徴が見受けられた。その例として BT（クラス A，ローカルメモリ率 25%）での例を図 2 に示す。グラフの横軸はページサイズ，縦軸は図 1 と同様に通常実行に対する相対実行時間である。

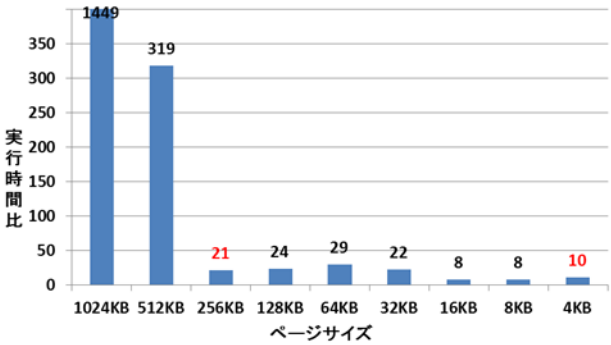

図 2　BT.A ローカルメモリ率 25%におけるページサイズ毎の実行時間

ページサイズ 1024KB，512KB の実行時間比はそれぞれ 1449，319 であるが，ページサイズ 256KB では 21 となり極端に減少する。このように，あるページサイズを境

に実行時間が大きく変化する事が他の応用，ローカルメモリ率においても確認できた。

また，ページサイズ 8KB や 4KB といった非常に小さなページサイズでは 16KB 以上のページサイズに比べてわずかながら実行時間の増加がみられた。これは，一般的に知られているように，通信サイズが小さいと毎回の通信セットアップ時間によるオーバヘッドが蓄積される事に加え, DLM システムではスワップの度にシグナルハンドラの処理を行っているので，小さいページによる通信オーバヘッドが応用の実行性能を悪化させたと考えられる。すなわち，一般に，メモリアクセスの局所性や通信コストを考えれば，小さいページよりも，大きなページにするほうが効率的である．しかし，大きなページでの転送は，平均的には，ページ内に利用しないデータも多く含まれることとなり，結果として，不要なデータの転送とローカルメモリの無駄を引き起こす可能性がある．したがって，応用のアクセスパターンやローカルメモリサイズに応じて適したページサイズを選択することが重要となる。

2.2 実行時間増加の原因についての考察

実行時間の急激な増加が見られた NPB の各種応用はいずれも for 文の多重ループを複数回繰り返す構造の応用である。こういった構造の応用では特定のワーキングデータセットに繰り返しアクセスを行うことが多い。前述のようにページベースでの管理を行う場合，ページ中には直接使用しないデータも含まれる。ローカルメモリに余裕があれば大きな影響はない。しかし，今回のようにローカルメモリが少ない場合には，計算ノードが保持できるページの枚数が非常に少なくなってしまい，ループ内でアクセスするデータ（ワーキングセット）すべてをローカルメモリ内に格納することができず，ループ実行中に限られた幾つかのページが繰り返し，出し入れされてしまう。このようなスラッシングが発生した事により，ページスワップ処理回数が急激に増加，実行時間の爆発的な増加を引き起こしたと考えられる。

3. ページサイズ自動調整機構の設計

3.1 ページサイズ自動調整機構の基本方針

スラッシングの発生が実行時間増加の原因であると考えられるが，スラッシングを起こさないための設定を逐一ユーザが行うのは非常に難しい。何故なら，ワーキングセットの大きさや適したページサイズの決定にはその応用プログラムのメモリアクセス局所性や計算ノードのメモリサイズの状況，ローカルメモリ率等が複雑に絡んでしまうためである。また一つの応用プログラムであっても，処理部分によって，適正ページサイズは異なる場合もありうる。

DLM システムはユーザレベルソフトウェアであり，OS の設定するページサイズの倍数サイズであれば，独自のページサイズの設定が可能である．そこで，これを利用し，応用プログラムに依存するワーキングセットに基づいて実行中にページサイズを最適化する事によりユーザに負担をかけることなくスラッシングを回避するページサイズ自動調整機構の設計と実装を行った。

ただし，ここで自動調整の対象とするのは，NPB のように，計算のコアとなるループ文を含みイテレーションを持つ構造の数値計算分野の応用プログラムである。

3.2 ワーキングセットの計測と判定方式

あるループ文において使用されるワーキングセットを調べ，それがローカルメモリにちょうど収まるようなページサイズに調整する事ができれば理想的だが，厳密なワーキングデータセットを求めることは，処理の負荷，情報記録のためのメモリ量の観点で現実的ではない。そこで自動調整機構では，正確なワーキングデータセットを求めるのではなく，ループ文の中でスワップインされたページ枚数を記録し，それを「およそのワーキングセットのサイズ」（ワーキングセットページ数）として使用する。

ページサイズ変更のための判定は，このワーキングセットページ数とローカルメモリのページ数の大小を比較する事で行う。ワーキングセットの方が大きければ，理想的なページサイズを以下の式で求め，理想サイズに最も近い小さいページサイズに一度に変更する。

理想ページサイズ（Byte）＝ローカルメモリサイズ（Byte）/ワーキングセットページ数　　(式 1)

ワーキングセットページ数の方が小さかった場合には，それ以前のページサイズ変更で小さくしすぎている可能性を考慮し，通信効率を上げる可能性を試すため，ページサイズを大きくする。ただし，ページサイズが無制限に上下する事で最大値と最小値を往復するような事態を避けるため，ページサイズを小さくする場合には 1 式の理想サイズに一度に下げるが，サイズを大きくする場合には段階的（たとえば 2 倍）に増やすようにする。

3.3 ページサイズ動的変更機構

一つの応用プログラムを実行した場合, 最適なページサイズは処理部分毎に異なる可能性がある。その一例を図 3 に示す。図 3 は SP（クラス B，ローカルメモリ率 10%) においてページサイズ 16KB と 32KB における処理部分

毎の実行時間を計測したものである。どちらのページサイズが適しているかは断定できるものではなく，部分毎に適したページサイズは異なっている事が分かる。このため自動調整機構では前述の判定と調整を処理部分毎に個別に行うものとする。これを実現するためには，応用プログラムの動作中に，処理部分毎にページサイズを大小に細かく変更していく必要がある。

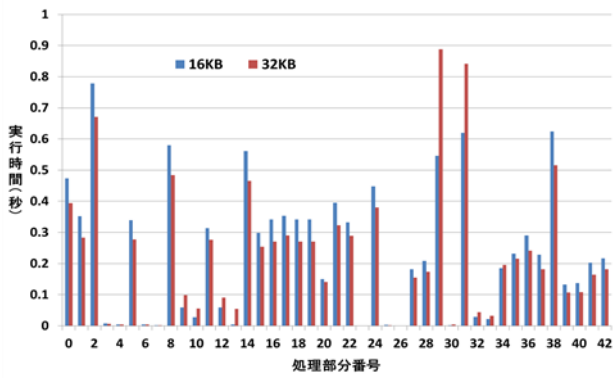

図 3　SP.B(10%)における処理毎の実行時間

ページサイズ変更手法として，連続した複数ページを一枚のページとして扱うという方法を採用する。自動調整を行う上での基準となる最少ページサイズを使ってページ管理表を構築し，メモリの管理そのものは最小ページを基本に行う。スワップ等の処理を行う際には管理している最小ページをセット単位で扱い，指定されたページが含まれる複数ページを一つのセットとし，一枚の大きなページとして扱うこととする。セット枚数は 2 の累乗倍で適宜変更することで，ページサイズの動的変更を実現する。

4.　ページサイズ自動調整機構の実装

4.1 可変ページサイズの実現

連続した複数のページを同時転送する事でページサイズを動的に変更するには，常にメモリサーバのアドレス空間上でページの連続性が保たれている必要がある。しかし，従来の DLM システムのスワップ処理方法は，計算ノードが必要とするページをメモリノードから計算ノードへ転送し，それによってメモリノード上に出来た空きスペースへ計算ノードから入れ替わりに送られてきたページを格納するというものであり，スワップ処理によってページの所在が次々に入れ替わってしまうものであった。そのため，ある程度処理を続けるとページの並びは図 4(a)のようにバラバラになってしまう。そこで今回は各ページのホームとなるメモリサーバを固定化するために図 4(b)のようなキャッシュ仕様への変更を行った。

キャッシュ仕様では全てのページをメモリノードに確保し，計算ノードではメモリノードの持つページのコピーのみを持つようにする。それによりスワップ時にはページのコピーを行う事になり，オリジナルのページは常に固定のメモリノードにおかれ，スワップのたびに移動する事がなくなる。

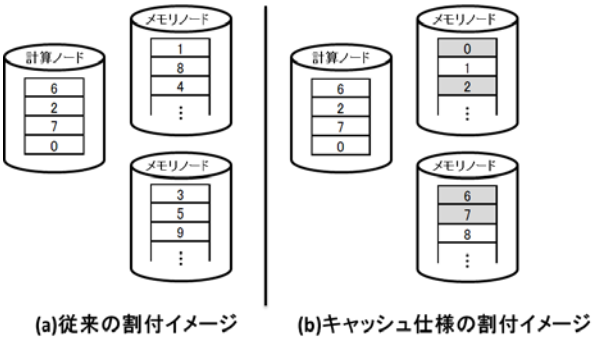

図 4　キャッシュ仕様のイメージ

こうすることで，スワップ処理を繰り返してもページの連続性が維持され，複数ページでの一括送受信が可能となる。また，応用プログラム実行中にページサイズを変更する事を考慮して各ノードへのメモリ割付は最大ページサイズに対応したセット数の倍数になるように実装し，通信ヘッダにセット枚数の情報を持たせることで，セット数変更直後に発生する複数のページサイズが混在する状況にも対応できるようにしている。セット枚数の指定は 2 の累乗で，現実装では，1～64 倍サイズの変更が可能である。具体的には 16KB を基本とした場合にはセット数 64 を指定すれば 16KB×64 枚で 1024KB のサイズが 1 ページとして扱われるようになり，16KB～1024KB の範囲でのページサイズ変更が可能となる。

4.2　ページサイズ変更時の過渡的状況への対応

セット枚数を変更してページサイズを大きくした場合，それまでの小さいページサイズ単位での送受信による影響を受ける。すなわち，ページスワップ時に，受信予定のページの一部が既に受信済み，もしくは送信予定のページの一部が存在しない，といった状況が発生する。このまま通信を行ってしまうとローカルメモリに最新のデータがあるにも関わらず，メモリサーバから古いデータを受信して上書きしたり，送信しようとしたページが存在しないといった状況が発生してしまう。そういった状況に対応するために，スワップ処理の前にページの欠けをチェックする関数を追加してある。スワップの直前にこれから操作するページの状況を調べ，スワップ時のヘッダにその状況に合わせたリクエストを書き込んで必要な部分だけを送受信できるようにしている。また，従来のスワップ処理は要求と応答が 1 対 1 対応だったが，リ

クエストをヘッダにまとめて記入することで計算ノードからの一度のページ要求でメモリサーバから必要なページのみを不連続な小さいページとして複数回受信することも行えるようにしている。

4.3 ワーキングセットサイズの計測

一つの応用プログラムが複数の繰り返しを含む処理部分に分けられるとき，その処理部分毎に個別にページサイズを最適化するために，ユーザが処理部分の前後に swapin_countstart（int id）と swapin_refresh(int id)の 2 つの関数を挿入する。図 5 の例に示すように，一つ一つの処理部分に id をつけて区別する．本機構では引数の id を基準に個別に情報を記録し，ページサイズの調整を行っていく。また，個別での調整を行うにあたり，従来のページ管理表にページ毎のスワップイン回数を記録するための変数 swapin count を追加し，また id 毎の理想ページサイズ（request setnum）を記録しておくための表を用意してある。

swapin_countstart 関数では，スワップイン回数の記録の開始と理想ページサイズへのページサイズ変更を行う。swapin_refresh 関数ではスワップイン回数の記録を終了し，ワーキングセットページ数を求めた上でスワップイン回数の記録をリセット，目標ページサイズの算出と理想ページサイズへの変更を行う。

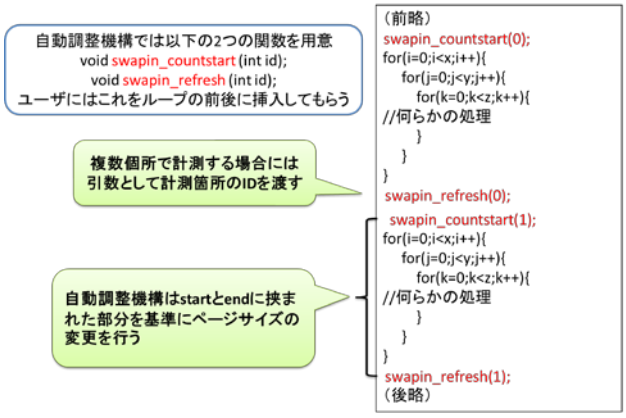

図 5　ワーキングセットサイズ計測関数の挿入例

具体的には，まず初回のイテレーションで swapin_countstart 関数を通過した時には，id に対応した理想ページサイズはまだ記録されていないため，ページサイズの変更は行われない。スワップが発生した場合には該当ページの swapin count に加算を行い，swapin_refresh 関数ではワーキングセットページ数として swapin count が 0 でないページの枚数を集計してワーキングセットページ数とし，swapin count は全て 0 に初期化する。その後，式 1 の理想ページサイズを算出し，管理表の自身の id の欄に理想ページサイズを記録し，ページサイズをその値に変更する。次に 2 度目に swapin_countstart 関数を通過した場合には，まず前回 swapin_refresh 関数が記録した理想ページサイズにページサイズを変更する。その後，前回同様にスワップイン回数の記録を行っていく swapin_refresh 関数では初回と同様の処理を行う。こうすることで，2 回目以降は同 id の swapin_refresh 関数の記録した理想ページサイズを swapin_countstart 関数が呼び出して再開する形となり，id 毎に個別にワーキングセットページ数の計測とページサイズの変更を行っていくことができる。

算出される理想ページサイズは計測時のページサイズによって計測解像度が異なってくるため，処理部分毎に繰り返し計測する事でページサイズは調整される。

5. ページサイズ自動調整機構の評価実験

ページサイズ自動調整機構の動作と効果の確認のため，評価実験を行った。ここでは 2 節で挙げたベンチマークのから NPB の BT クラス A（ローカルメモリ率 5%），SP クラス B(ローカルメモリ率 10%)を自動調整の効果が大きい例として示す．一方，スラッシングがほぼ発生しない場合の例として姫野ベンチマークのクラス M（ローカルメモリ率 10%）を示す。

5.1 BT.A ローカルメモリ率 5%での効果

NPB の BT クラス A（ローカルメモリ率 5%）での実験結果を図 6 に示す。縦軸はローカルメモリ率 100%での実行時間を 1 とした場合の実行時間比，横軸はページサイズを示す。512KB～16KB は固定ページサイズでの実行時間比，自動調整は 1024KB～16KB の範囲で自動調整を行った場合の実行時間比である。尚，自動調整機構の最小ページサイズは今回の実験では 16KB に設定している。最小ページサイズは，応用が扱うメモリ規模，クラスタのメモリ容量にもよるが，小さすぎて弊害が出ない程度にしている。

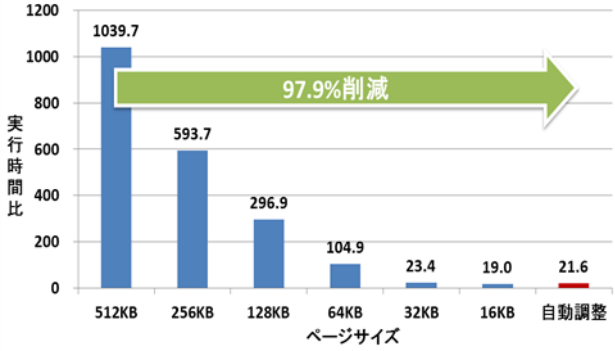

図 6　BT.A(ローカルメモリ率 5%)での自動調整の効果

512KB では実行時間比 1039.7 まで増加していたが，自動調整機構を使用することによって 21.6 まで改善する事ができ，およそ 98%実行時間を削減できた。尚，実験時間を短縮化するため，NPB の繰り返し数は 10 回にして実験を行っているが，本来の繰り返し数は 200 回なので，本来は，自動調整によりさらに大きな効果が得られると考えられる。

5.2 SP.B ローカルメモリ率 10%での効果

SP クラス B（ローカルメモリ率 10%）においても 4.1 と同様，最大 99.3%の実行時間の改善が見られた。その結果を図 7 に示す。

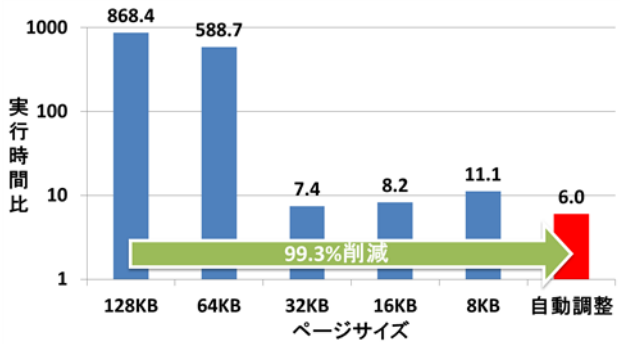

図 7　SP.B(ローカルメモリ率 10%)での自動調整の効果

図中の 128KB～8KB は各ページサイズ固定で動作させた場合の実行時間比である。128KB 固定では実行時間比 868.4 だったのに対し，自動調整を行う事で 6.0 まで実行時間が削減されている。

今回自動調整を行った事で固定ページサイズでの最良値である 32KB での 7.4 を下回る結果を得た。これは複数ヶ所で個別にページサイズの調整を行ったため，固定ページサイズで統一するよりも良い結果が得られたものと思われる。尚，今回はイテレーション 1 回あたり id0 から id42 の計 43 箇所で調整を行っている。

また，BT での実験同様，NPB の本来の繰り返し数 200 回を 10 回に減らして計測しているので，本来の 200 回の設定であれば自動調整の有無による差はより大きなものになる。

5.3 Himeno.M ローカルメモリ率 10%での効果

最後に姫野ベンチマーク（クラス M，ローカルメモリ率 10%）の実験結果を示す。これは，スラッシングが発生しにくく，調整が不要な応用においても自動調整機構がうまく動作を行うことを確認するための実験である。その結果を図 8 に示す。自動調整機構を使用した結果は，実行時間比は 4.6 であった。

姫野ベンチマークは，NPB の BT，SP，FT は異なり，メモリアクセスの局所性が高く，ワーキングセットサイズも小さいため，スラッシングが起きにくい。このため通常，ページサイズは大きい方が良い結果の出る応用である。自動調整機構を使用した結果，ページサイズを大きいまま維持できている事が確認できた。

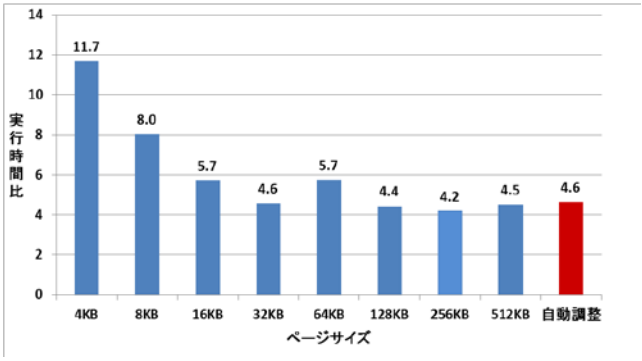

図 8　Himeno.M(ローカルメモリ率 10%)での自動調整の効果

6. おわりに

今回は計算コア部分におけるアクセスページ数をワーキングセットサイズとし，ページサイズ自動調整のヒントにする方法を提案した。限られた応用での検証ではあるが，実行時間を最大で 99%以上削減する等，非常に高い効果が得られることが確認できた。こういった自動調整機構を導入する事により，ユーザは最適なページサイズというものを気にせず DLM システムを利用することが出来，スラッシングの発生を回避する事ができる。また，方法自体は DLM システムに限らずページベースのシステムであれば応用が可能なものとなっている。

今後は，様々な応用での効果についての評価実験も引き続き行っていく予定である。また，計測を行う計算コアの部分の指定は，現在ユーザによる手動挿入に頼っているが，どの部分を一つの処理部分として取り出すか，多重ループがあった場合にどの部分を計測するべきかなどのシステマティックな計測関数挿入方針の確立が望まれる。さらにそのような方針が確立できれば，コンパイラによる自動挿入についても可能性があると考えている。

参考文献

1) 緑川博子，黒川原佳，姫野龍太郎，“遠隔メモリを利用する分散型大容量メモリシステム DLM の設計と 10Gb Ethernet における初期性能評価”，情報処理学会論文誌 コンピューティングシステム，Vol.1，No.3

pp.136-157, (2008,12).

2) 緑川博子，齋藤和広，佐藤三久，朴 泰祐："クラスタをメモリ資源として利用するための MPI による高速大容量メモリ"，情報処理学会論文誌，コンピューティングシステム，Vol.2, No.4, pp.15-36, (2009,12)
3) H. Midorikawa, K.Saito, M.Sato, T.Boku: "Using a Cluster as a Memory Resource: A Fast and Large Virtual Memory on MPI", Proc. of IEEE International Conference on Cluster Computing(Cluster2009), pp.1-10, (2009,9)
4) S. Yoshimura, H.Midorikawa; "A C Compiler for Large Data Sequential Processing using Remote Memory", proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.198-202 , (2011,8)
5) T2K-Tokyo HA8000 クラスタシステム[Online](2012), http://www.cc.u-tokyo.ac.jp/ha8000/
6) Himeno Benchmark web site [Online] (2012), http://accc.riken.jp/HPC/HimenoBMT.html
7) NPB2.3-omni-C web size [Online](2012) http://phase.hpcc.jp/Omni/benchmarks/NPB/index.html
8) 内山丞, 緑川博子,甲斐宗徳："遠隔メモリアクセスのためのページスワップページサイズ動的変更機構の検討"，第 10 回 FIT 論文集,B-050, pp365-367,（2011,9）
9) 内山 丞，緑川 博子: 遠隔メモリアクセスのためのスワップページサイズ自動調整機構の初期評価, HPCS2012, P5-3, (2012,1)