システムソフトウェアと
オペレーティング・システム 65−17
（1994．7．22）

管理情報共有機構 (MetaShare) を大域的仮想仮想記憶で利用した場合の基本性能評価

田沼 均　平野 聡　須崎 有康　一杉 裕志　塚本 享治

電子技術総合研究所

計算機システムを管理するには計算機を構成する各資源の状態の情報 (管理情報) を適切に把握することが重要である。特に超並列システムにおいては構成する PE 数が膨大な数にのぼるため、システム全体の状況を把握することは非常に困難である。本論文ではシステムの負荷状態に適合的に変化する管理情報共有機構 (MetaShare) の概略を述べ、並列マシンのシミュレータ上で大域的仮想仮想記憶で利用した場合の基本性能評価を行ない有効性を検証するとともに MetaShare の各パタメータの影響を測定したので、その結果について述べる。

Performancd Evaluation of the Administrative Information Management System for Massively Parallel Systems

TANUMA Hitoshi HIRANO Satoshi SUZAKI Kuniyasu
ICHISUGI Yuuji TSUKAMOTO Michiharu

Electrotechnical Laboratory

Operating systems must know all conditions about computing resources. But it is difficult to know them under massively parallel computer environment because the information is spread over millions of processing elements. We propose an administrative information management system(MetaShare). This system uses a hierarchical management structure, which dynamically reconstructs itself according to resouce condition changes. We also describe the result of performance evaluation of MetaShare which is used for searching a lowly loaded PE for the Global Virtual Virtual Memory(GVVM).

1 はじめに

現在、大規模な汎用超並列システムのためのオペレーティングシステム「超流動OS」を開発中である[1]。超並列システムではネットワークトポロジへの依存性が高く規則的な動作をするデータパラレルのプログラムや、規則性が低いコントロールパラレルのプログラムといったさまざまなパラダイムのプログラムが混在して実行される。「超流動OS」の目標はこの超並列システム環境下で、非常に多数のPEやプロセスを効率良く管理することである。

超並列システムを管理する上で以下の点が問題となる。

1. 計算機資源やアクティビティを管理するには、システムの状態を適切に把握することが必要である。超並列システムでは膨大な数のPEがネットワークにより接続された構成をとる。このためシステムの状況、例えば個々のPEのCPUの負荷、メモリの使用状況、ネットワークの負荷などを瞬時に正確に把握することは非常に困難な問題である。

2. OSは通常、実行制御、記憶管理などのサブシステムはそれぞれ目的に応じてシステムの状況の把握を行なうが、そのベースとなる情報は例えばCPU負荷やメモリ負荷、ネットワーク負荷といった共通するものが多い。このような状況は各々のサブシステムの機能の重複であると共に個別に同一情報を収集することにより例えばネットワークトラフィックなどの増大を招く。これが第二の問題である。

これらの問題を解決するために[4, 5]で管理情報共有機構(MetaShare)を提案した。MetaShareはOSのサブシステムで共通して使用されるメモリ負荷、CPU負荷、ネットワーク負荷といったシステムの管理情報を収集し、統合的に管理し、必要とする各サブシステムに配布するシステムである。本論文では、MetaShareを大域的仮想仮想記憶のスワップアウト先PEの探索に利用した場合の基本性能評価実験を行なったので、それについて述べる。

2 MetaShareで解決すべき問題 — 大域的仮想仮想記憶のスワップアウト先PEの探索

MetaShareは超流動OSの中で一般的な管理情報を管理する機構であるが、本論文ではそのモデルケースとして超流動OSの仮想記憶管理システムである大域的仮想仮想記憶(GVVM)[2, 3, 6]と組み合わせて使用する場合について述べる。

分散メモリアーキテクチャをとる多数のPEを有する超並列システムでは多数のプログラムが混在すると、PEごとに必要とするメモリ使用量に差が生ずる。逐次型システムではメモリ使用量が多くなると仮想記憶を利用して使用頻度の低いメモリ領域をディスクなどの二次記憶にスワップアウトする。しかし高速なネットワークを備えた超並列システムでは、メモリ使用量が多いPEはメモリの仮想空間のスワップアウト先としてメモリ使用量が少ないPEのメモリを利用することが考えられる。GVVMはPE空間全体でのメモリ頻度に基づくデマンドページング、及び他のPE上の使用頻度の低いメモリをスワップ領域として用いることによりメモリの自動的な負荷分散を行なうサブシステムである。

GVVMにおいてページアウト要求が生じた時、スワップアウト先となるPEを捜す必要がある。このスワップアウト先PEの探索にMetaShareを使用する。

3 MetaShareの概要

MetaShareはメモリ負荷といった管理情報を収集し、収集した情報を分散データ構造を用いて抽象化、集約化して統合的に管理し、OS内のサブシステムからの問い合わせに応じて探索し、必要とする情報を配布する機構である。以下、GVVMのスワップアウト先PEの探索に利用した場合のMetaShareの概要について述べる。

3.1 動作原理

MetaShareでは情報管理構造として階層型構造をとる(図1)。階層の最下位階層はPE空間の個々のPEに相当する。さらに上位の階層はPE空間内に分散データ構造として構築する。各階層でPEの集まりを領域と呼ぶ。図1においては四角が領域を表現する。階層型構造を採用した理由は以下の通りである。

1. 階層型構造ではルートに近付くと全体を集約した情報が得られ、リーフに近付くと個々のPEに近い情報が得られる。さらに検索等の操作も容易に効率的に実現が可能である。

2. 階層型構造はうまく構成すればシステム全体の情報収集及び配布のコストがPE数の対数オーダで可能となり、膨大な数のPEを有する超並列システムにおいても効率的な情報の収集、配布が期待できる。

GVVMのスワップアウト先PEを捜し出す目的でMetaShareの階層型構造を構築する際、次の二点が重要である。

第一は平均メモリ負荷の均衡化を図ることである。GVVMはメモリ負荷の均衡化を図るためにスワップアウト先PEを探索する。したがって探索の際に使用す

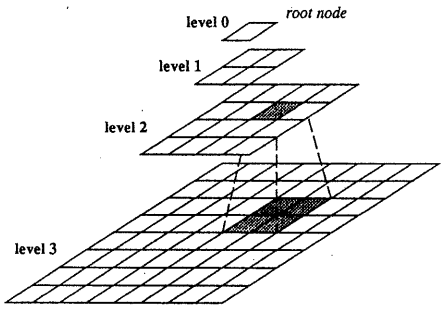

図 1: MetaShare の階層構造

る MetaShare の階層構造にはメモリ負荷の均衡化が反映されていなければならない。そこで情報構造の構築に際しては構成要素である各領域の平均メモリ負荷の均衡化を図る。各領域の平均メモリ負荷について均衡化している状態であれば、領域内全体のメモリ負荷の平均はシステム全体の平均値と近い値にあるためメモリ負荷が多くスワップアウトの要求を出した PE はその PE の所属する領域内にメモリ負荷の少ない PE が必ず存在する。

第二はスワップアウト元 PE とスワップアウト先 PE との間の通信距離を短くすることである。通信距離が長いと転送コストの増大を招き、システムの性能を損ねる。そこで領域を構成する際にその領域に所属する全ての PE について任意の 2PE の距離が一定値以下であるように制限する。以降この一定値を領域直径と呼ぶ。領域直径の距離はネットワークトポロジーに従った通信距離を使用する。このことにより 2 つの PE が同一の領域内に存在すれば、PE 間の距離はその領域直径以内にあることが保証される。この領域直径は最上位階層ではシステム全体をカバーする値となり、下位の階層に行くほど小さくなり、最下位階層では構成 PE が 1 つであることより 0 となる。

メモリ負荷はシステムの動作にしたがって刻々と変化する値である。したがって階層構造において常に領域間の平均メモリ負荷の均衡化を図るため、収集してきた情報に応じて動的に階層型構造を再構築する必要がある。状況に適合して再構築し均衡化を図ることにより、静的構造を用いた MetaShare の際に達成できなかった、メモリ負荷の均衡及び通信コストが低いスワップアウト先の探索の実現が可能となる。

3.2 動作の概要

MetaShare は階層型構造の各領域を代表する管理ノードにより構成される。各管理ノードはネットワーク上のパケットを用いて相互に情報の交換を行ないながら動作する。用いるパケットの種類は以下の 5 種類である。

MI packet 領域の平均メモリ負荷を上位階層の管理ノードに通知するパケット。

YCI packet 領域再構成後に下位領域の管理ノードに対し、下位領域の PE 構成を通知するパケット。

IYP packet 下位領域の管理ノードに対し、その下位領域は自分の領域に属していることを通知すると共に、下位領域の構成情報を要求するパケット。

MCI packet IYP packet の要求に応じ、領域に属する下位領域の情報を上位階層の管理ノードに通知するパケット。

NC packet 下位領域の管理ノードに対し、領域構成に変更がないことを通知するパケット。YCI packet の特殊形である。

動作は 2 つのフェーズより成る。

情報集約フェーズ

第一のフェーズは情報集約フェーズである。各 PE の情報収集部が収集してきたメモリ負荷値は、まず最下位階層 (level N とする) のノードに格納される。次にその情報をその PE の属する N-1 level の領域の管理ノードに通知する。受けとった管理ノードは領域の平均メモリ負荷値を計算し、上位の管理ノードに MI packet を使用して通知する。

以上の動作を各管理ノードで行ない、各階層の各領域の平均メモリ使用頻度を計算しつつ、その値を上位階層に通知し、最上位階層 (level 0) の管理ノードまで伝搬させる。level 0 の管理ノードに下位領域からの MI packet が全て揃った時点で、情報集約フェーズが終了し、第二のフェーズの領域再構成フェーズに入る。

領域再構成フェーズ

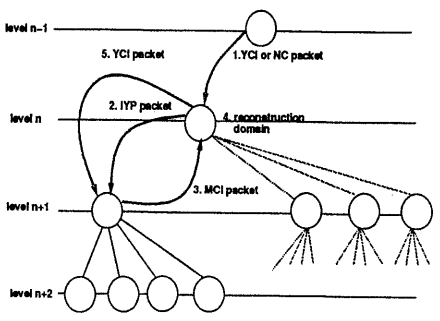

図 2: 領域再構成フェーズの packet の流れ (図中の数字は動作順序)

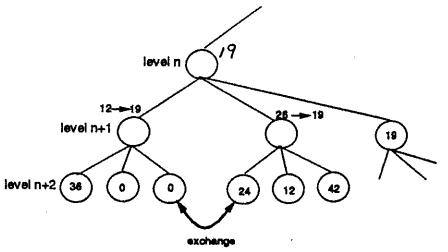

図 3: 均衡化の例

まず下位領域 (level n+1) の平均メモリ負荷を比較する。各領域の平均メモリ負荷の偏差が一定値を越えているとき、領域再構成動作を行なう。偏差が一定値以下であれば領域再構成が起こらなかったことを示す NC packet を、各下位の管理ノードに対して送る。

領域再構成動作は具体的には二段階下位 (level n+2) の領域の一段階下位 (level n+1) の領域への所属を組替えることにより実現する。まず二段階下 (level n+2) の領域の情報を得るために下位 (level n+1) の各管理ノードに対し IYP packet を送る。IYP packet を受けとった管理ノードは、自分の管理している下位 (level n+2) の領域情報を MCI packet を用い上位 (level n) の管理ノードに送る。送られてきた二段階下位 (level n+2) の領域の情報を元に、二段階下位 (level n+2) の領域を一段階下位 (level n+1) の領域に割り付ける (図 2)。領域の再構成は以下の制約条件の元で行なう。

1. 一段階下位の領域の平均メモリ負荷を均衡化させること。例えば図 3の状況で level n の管理ノードが領域再構成動作を行なうとする。level n での領域の平均値は 19 であるから、最初に平均値より大きい 42、36、24 を均衡化するように 36 と 24 の組、及び 42 に配分する。次に平均値より軽い 0、0、12 を均衡化するように配分すると、各領域構成は 36、24、0 の組と 42、12、0 の組となり、0 と 24 を交換した状態となる。

2. 各領域に所属する PE は相互の通信距離を一定数以内に収める必要がある。近似として管理ノードからの距離を指標に使用した。二段下位の領域は、二段下位の領域の管理ノードと一段下位の領域の管理ノードとの距離が一段下位の領域の定めた値 (領域直径の指標) 以下であれば、その一段下位の領域に所属することができる。

領域再構成動作終了後、各一段階下位の領域の管理ノードに対し結果を YCI packet により通知する。上位階層の管理ノードより NC packet または YCI packet を受けとった管理ノードは上述と同様の動作を行なう。ただし YCI packet を受けとった場合は下位の領域の所属が変更になっているため、下位領域に関する情報を再構成動作に先立ち更新する。

MetaShare は情報収集フェーズと領域再構成フェーズを一定時間間隔で同期して繰り返し実行する。

3.3 探索の方法

MetaShare がスワップアウト先 PE の探索要求を受けると以下の手順にしたがって探索を行なう。ただし最下層を level N とし、スワップアウト先 PE として適当であるかはその PE のメモリ負荷がスレシホルド値より小さいことにより決める。

まず探索要求を出した PE の所属する level n-1 の領域に対し探索する。level n-1 は各 PE で構成するためこの領域を管理する管理ノードに問い合わせれば良い。この領域の中で要求された個数の PE が得られたらそこで探索を終了し、得られた PE を結果として返す。

十分な PE 数を得られなければ、探索領域を拡大するために level n-2 の領域を探索する。まず探索要求を出した PE の所属する level n-2 の領域において、最も平均メモリ負荷の小さい level N-1 の領域を選ぶ。選んだ level n-1 の領域の管理ノードに問い合わせ、スワップアウト先として適当な PE を得る。ここで十分な個数の PE が得られなければ、さらに上位の階層を同様な方法で探索する。

領域再構成による均衡化の効果により低い階層において、つまり近い距離でメモリ負荷の小さい PE を見つけることが期待できる。

4 基本性能評価実験

4.1 実験の条件

並列マシンのシミュレータ上に MetaShare 及び GVVM を作成し、MetaShare の基本性能の評価実験を行なった。実験の目的は以下の通りである。

1. GVVM のスワップアウト PE 探索に利用した場合の MetaShare の有効性の検証。

2. MetaShare の挙動の解析。特に情報収集時間間隔、領域再構成頻度、PE 数の MetaShare に及ぼす影響の解析。

作成した並列マシンシミュレータは論文 [6] で使用したマルチプロセス環境シミュレータのスワップアウト先 PE 探査部分に本論文で述べた MetaShare を組み込んだものである。具体的な条件は以下の通りである。

- ネットワークは 2 次元トーラス ($N \times N$) で、ブロッキングのあるバーチャルカットスルー転送を行なう。PE 数は 35 から 324。距離はネットワークのホップ数とする。

- 最終的なスワップアウト先であるディスクは 9PE(3×3) 単位で配置する。ネットワークとディスクの転送速度比は 6.25 : 1。

PE 数	level				
	0	1	2	3	4
36	6	2	2		
81	8	4	2	2	
144	12	6	2	2	
225	14	8	4	2	
324	18	8	4	2	2

表 1: level ごとの領域直径の指標の大きさ

- PE 当たりの実記憶は 40 ページ、仮想記憶空間は 80 ページ、PE 上のプロセスは仮想空間を正規分布に従ってアクセスする。
- ページの書き込率は 30%。
- 計測した時間はシミュレータの仮想時間である。1 単位時間はネットワークで隣接する PE に 1 つのパケットの転送に要する時間に相当し、以下 1 step と呼ぶ。
- 投入するプロセスは長方形とし大きさは乱数で決定する。大きさの分布は縦、横の PE 数ともにシステムの辺長の 2 割以上 8 割以下の一様分布。
- プロセスの割り付けは文献 [7] によるファーストフィット・アルゴリズムを使用する。
- あらかじめ 40 個のプロセスを作成しておき、割り付け可能となった時点で直ちに投入する。全プロセスが実行を終了するまでの時間を終了時間と呼び、評価の尺度とする。
- ネットワークの転送時間及びディスクの転送時間は実行時間の中に含まれるが CPU 使用時間は含まれない。
- MetaShare は全体で同期して動作する。すなわち情報収集フェースに始まり領域再構成フェーズが終るまでのサイクルは、全 PE で同期して行なう。MetaShare の実行時間が情報収集時間間隔を越えた場合、情報収集フェーズの開始は前の MetaShare の実行が終了するまで遅らせる。
- MetaShare の階層構造は正方形の辺をそれぞれ 2 等分して作る 4 分木である。管理ノードはあらかじめシステム中に均一に分布するように割り当てる。
- 初期状態として各管理ノードは最も近い上位階層管理ノードの領域に所属するように階層構造を作成する。MetaShare の領域直径の指標は、この初期状態においてその領域内でその領域の管理ノードから最も遠い下位領域の管理ノードまでの距離とする。実際の値は表 1のようになる。

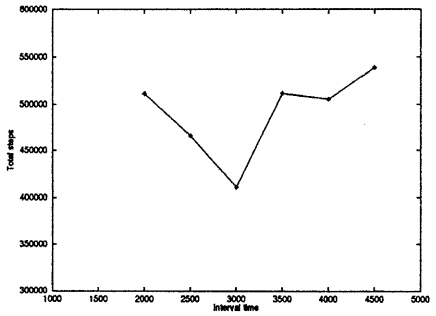

図 4: 収集時間間隔と終了時間

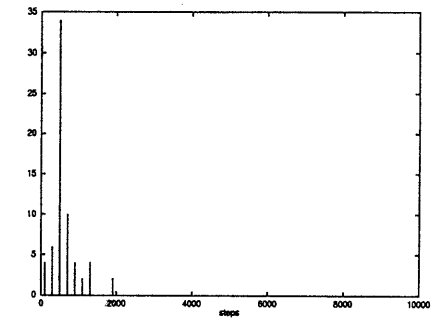

図 5: MetaShare の動作時間

- MetaShare ではメモリ負荷を 0 から 1 までの値で正規化して扱い、領域再構成はその領域に属する一段下位の各領域の平均メモリ負荷の偏差がある値 (領域再構成スレシホルド) を越えた時点で行なう。

4.2 収集時間間隔の影響

図 4に PE 数 324 においての情報収集時間間隔と GVVM シミュレーションの終了時間の関係を示す。

情報収集時間間隔を広げていくと 3000(steps) までは終了時間は低下し、3000 を越えると上昇する。

MetaShare の動作はシステム全体で同期して行なう。このため情報収集の時間間隔を情報収集フェーズに始まり領域再構成フェーズが終了するまでの時間 (MetaShare の動作時間) より短くとっても、情報収集の開始が待たされるため意味を持たない。

そこで MetaShare の動作時間 (情報収集フェーズ及び領域再構成フェーズに要する時間) の測定を行なった。

図 5は PE 数が 324、領域再構成スレシホルドを 0.05 とした場合の MetaShare の動作時間のヒストグラムである。動作時間は 200 間隔で区切って集計した。情報収集時間間隔は 1000 以下にしたのでは多くの場合 MetaShare の実行が終っていないため意味はなく、2000 程度ではまだ MetaShare の前の動作が終了せず、待た

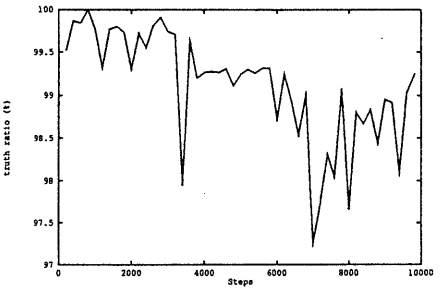

図 6: 正解率の時間的変動

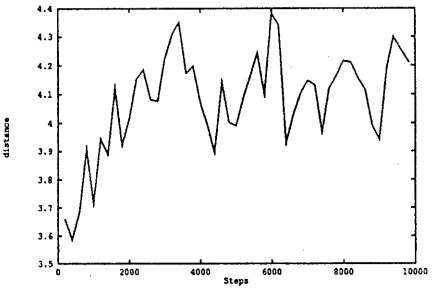

図 7: スワップアウト先 PE までの平均距離の時間的変動

される場合があることを示している。

情報収集時間間隔は以下のような影響を持つと考えられる。

- 情報収集時間間隔を長くすると MetaShare の蓄えている情報が古くなり信頼性が低下する。
- 情報収集間隔を短くすると MetaShare が頻繁に動作し、ネットワークに対し負荷をかけ、システム中の他の動作 (ページ転送など) を妨害する。

MetaShare の蓄えている情報の信頼性を測定した。情報収集間隔を 10000(steps) にとり、MetaShare の動作開始から情報収集間隔内での時間変動の平均を測定した。信頼性の基準として MetaShare にスワップアウト候補 PE を二つ選ばせ、実際にスワップアウトに適当だった場合の割合 (正解率) と、スワップアウト先 PE までの距離の平均を測定した。PE 数は 324、領域再構成スレッシホルドは 0.05 である。まず正解率の時間変動の結果を図 6に示す。

まず全体を通して正解率は 97% 以上である。MetaShare は高い確率でメモリ負荷値の十分低い PE を探索できる能力があるといえる。

正解率は変動が大きいがおおよそ 3000 程度までは 99.5% 程度の高率であり、その後徐々に低下する傾向にある。これは時間経過とともに MetaShare の蓄えている情報が古くなり、実際とずれてくるためと考える。

次にスワップアウト先 PE までの距離の時間変動の結果を図 7に示す。

まず候補 PE の内、最も遠くの PE を採用する必要がある場合においても距離が 4.4 程度である。ネットワークの大きさが 18 × 18PE であり、投入されるプロセスの大きさがこの場合 4 × 4PE から 14 × 14PE であることを考えると、MetaShare は十分近い PE を捜し出す能力があるといえる。

また距離については 3000 程度まで大きくなりその後、ほぼ一定の水準となる。3000 までの間は、情報収集時点から間がない間は MetaShare の捜し出したスワップアウト候補 PE の内最も近いものがスワップアウト先 PE として使用できる割合が多く、時間がたつに従って候補の内より遠くの PE を使用する必要が出てくるためと考えられる。図 6とも考え合わせると、3000 以降は候補 PE の内で最も遠くの PE が使用される割合が高くなるため、ほぼ一定の値を示すものと考えられる。

以上よりこのケースでは情報収集間隔として 3000 程度が適当な値といえる。PE 数がより大きくなると MetaShare の実行時間が大きくなり、情報収集時間間隔をより大きな値とする必要が生じる。使えるネットワークのバンド幅が小さい場合、例えばネットワークの転送速度が遅い場合やネットワークの形状が異なる場合、アプリケーションがネットワークを頻繁に使用する場合などは情報収集間隔をより大きくしてネットワークへの負荷を減らす必要が生じる。

4.3 領域再編成の頻度の影響

領域再構成は各領域において属するその一段下位の領域の平均メモリ負荷の偏差が領域再構成スレシホルドを上回る時点で行なう。従って領域再構成スレシホルドを大きくとれば領域再構成が起き難くなり、また小さくとれば領域再構成が頻繁に起こるようになる。

そこでまず、領域再構成スレシホルドと再構成の起こりやすさの関係を調べた。図 8は領域再構成スレシホルドを変化させ、MetaShare の情報収集時間間隔を単位としてその間に起こる領域再構成の平均回数を示すグラフである。PE 数は 225、情報収集時間間隔は 2000 とした。

領域再構成スレシホルドが 0.05 までは値を下げるに従って領域再構成が頻繁に起こるようになる。しかし 0.05 以下では飽和してしまい、それ以上は増えなくなる。この変化は各階層において同一ではない。例えば同様なグラフを level 2 のみの再構成平均回数について示したグラフが図 9である。

実験を行なった PE 数が 225 の場合、階層数は 5 で、

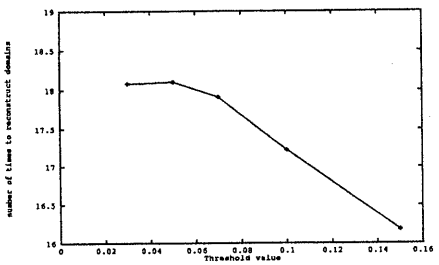

図 8: 領域再構成スレシホルドと全 level における平均領域再構成回数

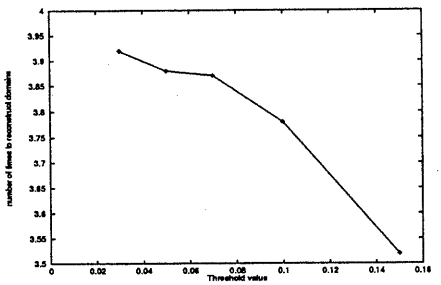

図 9: 領域再構成スレシホルドと level 2 の平均領域再構成回数

level0(ルート) と level4 (最下層、PE1 個で構成される) では再構成が起こらない。全体の平均領域再構成回数と異なり、0.05 より小さくしても再構成の回数は増加する。この傾向は level 1 においても同様である。つまり上位階層に行くに従って領域再構成回数の飽和点は下がる。しかし、管理ノードの数は圧倒的に下位階層の方が多いため全体の領域再構成回数には反映されない。

領域再構成回数が及ぼす影響は以下のことが考えられる。

- 領域再構成を行なうことにより、MetaShare はスワップアウト先 PE の探索でよりメモリ負荷の均衡が図れる PE を求めることができる。
- 上位階層はシステム中のより広範囲な領域の PE を構成対象とするので、上位階層で頻繁に再構成が起きると含まれる PE 間の距離が大きくなり易くなる。
- 再構成が起こった場合と起こらなかった場合と比較してパケットの総量が 4 分木の場合、14 倍にも達する。従って頻繁な領域再構成はネットワークに負荷をかけることとなる。

領域再構成スレシホルドを変化させた場合の終了時間を図 10に示す。

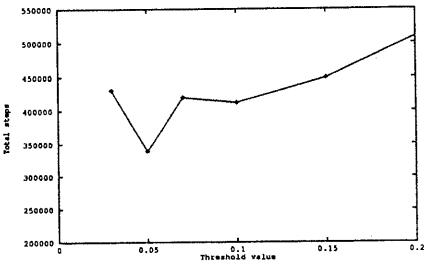

図 10: 領域再構成スレシホルドと終了時間

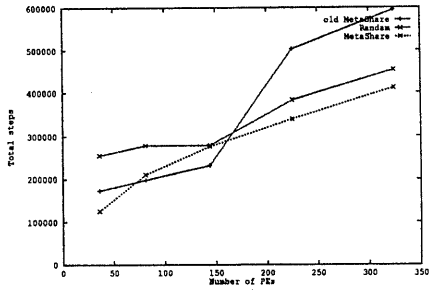

図 11: PE 数と終了時間

領域再構成スレシホルドが 0.05 まで減少傾向にあり、0.05 以上では増加傾向となる。領域再構成スレシホルドを 0.05 まで減少させる場合は領域再構成がより多く起こり、MetaShare の捜し出した PE がメモリ負荷分散に寄与し、全体の効率を上げるものと考える。0.05 よりさらに減少させると下位階層の再構成回数は飽和し、上位階層においてのみ増加し、不要な再構成が増加するため実行効率が低下するものと考えられる。

4.4 PE 数と終了時間

PE 数を変化させた場合の終了時間の影響を図 11に示す。参考に論文 [6] において評価した領域再構成を行なわない MetaShare(old MetaShare) とランダム戦略 (Randam) を用いた場合を合わせて載せた。領域再構成を行なわない MetaShare は 2 進木によるトーナメントにより全 PE より最もメモリ負荷の小さい PE を一定数個選びだし、その情報を再び 2 進木を用いて全 PE に知らせる。またランダム戦略は乱数を用いて全 PE 中からスワップアウト先 PE 候補を抽出する。

ここで述べた MetaShare は全体にわたって少ない終了時間を示し、PE 数が増加しても終了時間の増加は緩やかである。また他の方式と比較しても PE 数が 80 から 150 近辺では領域再構成を行なわない MetaShare の方が終了時間は小さいが他の PE 数、特に PE 数が増大した場合、本方式の MetaShare が最も良好な結果

を示した。

5 おわりに

本論文ではGVVMのスワップアウト先PEの探索に利用する場合のMetaShareの概要について述べ、並列マシン上のシミュレータ上での基本性能評価実験を行ない、MetaShareの有効性を示し、MetaShareのパラメータ(情報収集時間間隔、領域再編スレシホルド)やシステムのPE数の影響について述べた。

本論文で述べたMetaShareはGVVMのスワップアウト探索戦略として有効であり、他方式と比較しても効率が良い。また各パラメータは最適値を有し、MetaShareはチューニングを行なうことにより、効率良く動作することを示した。

現在、以下の点を重要な問題と考え、検討を行なっている。

1. 今回の基本評価実験はネットワークは2次元トーラス、PE数が36から324、変化させたMetaShareのパラメータとして情報収集時間間隔及び領域再構成スレシホルドといった条件の元で行なった。様々な条件、例えばより多数のPE、異なるネットワークなどの条件の元で、MetaShareのより詳細な動作解析を行なう必要がある。
2. MetaShareは全体で同期して動作し、このことにより情報収集間隔が制限される。MetaShareの動作を非同期化し、この制限を解消する必要がある。
3. 現在MetaShareは各PEのメモリ負荷の値のみを使用する。例えばプロセスの割り付け状況など他の情報を活用することにより効率的な実行を図る必要がある。
4. MetaShareは動作環境により各パラメータのチューニングを行なう必要があるが、パラメータの調整の自動化を図りたい。
5. 現在、MetaShareの応用としてGVVMのみであるが他のOSサブシシテム、例えばプロセス割付け等においての利用も検討している。

謝辞

本研究の一部はRWC計画の一環として「超並列システムアーキテクチャに関する研究」で行なわれたものである。関係各位に感謝する。

参考文献

[1] 平野, 田沼, 須崎, 濱崎, 塚本. 超並列用システム用オペレーティングシステム「超流動OS」の構想. 情報処理学会研究会報告93-OS-58,Vol. 93, No.27, pp17-24, 1993.

[2] 平野, 田沼, 須崎. 超並列システム用OS「超流動OS」における大域的仮想仮想記憶. JSPP'93, pp237-244, 1993.

[3] 平野, 田沼, 須崎. 超流動OSの大域的仮想仮想記憶におけるページ探索法の比較. 情報処理学会研究会報告93-OS-61(SWoPP'93),pp65-72, 1993.

[4] 田沼, 平野, 須崎, 浜崎, 塚本. 超流動OSのための管理情報共有機構(MetaShare)の設計. 情報処理学会研究会報告93-OS-61(SWoPP'93),pp73-80,1993.

[5] 田沼, 平野, 須崎, 一杉, 塚本. 適合型マップを用いた超並列システム用管理情報共有機構の提案. 情報処理学会研究会報告94-OS-63,Vol.94, No.32,pp33-39, 1994.

[6] 平野, 田沼, 須崎, 一杉, 塚本. 大域的仮想仮想記憶(GVVM)のマルチプロセス環境での評価.JSPP'94, pp365-362, 1994

[7] Y. Zhu. Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers. J. of Parallel and Distributed Computers,pp328-337,16, 1992.