システムソフトウェアと
オペレーティング・システム 63－5
（１９９４．３．２２）

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

田沼 均　　平野 聡　　須崎 有康　　一杉 裕志

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。本論文では適合的に変化する管理情報共有機構 (MetaShare) を提案する。本システムは管理情報を階層構造を用いて管理する。階層構造はシステムの状態により動的に再構成する。MetaShare のアプリケーションの一つとして超流動 OS の一部である大域的仮想仮想記憶と組み合わせた実験システムを作成し予備実験を行なったのであわせて報告する。

Proposal of an Administrative Information Management System Using Adaptive Information Structure for Massively Parallel Computers

TANUMA Hitoshi　HIRANO Satoshi　SUZAKI Kuniyasu　ICHISUGI Yuji

Electrotechnical Laboratory

Operating systems must know all conditions about conputing resources. But it is difficult to know about 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 the accumulating information.

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せる。「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 やアクティビティを効率良く管理することである (図 1)。

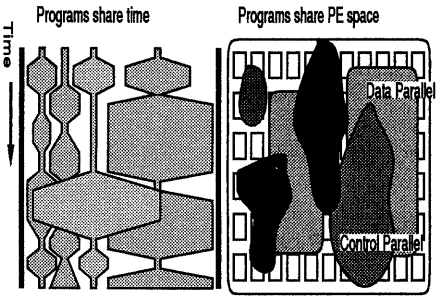

図 1: 時間と空間を分割共有する超流動 OS

超並列システムを管理する上で以下の点が問題となる。

第一に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がある。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。

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] で管理情報共有機構 (MetaShare) を提案した。本論文ではより効率的な管理を目指し、システムの状況に応じて動的に変化する適合型マップを用いる MetaShare を提案する。

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] 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。探索の対象は超並列システム内の全 PE であり、以下の条件を満たすものである。

1. 他の PE のページを受け入れる余裕があるほど十分にメモリ使用量の少ない PE であること。
2. PE 間距離が小さいこと。

   スワップアウトを行なう際、さらに将来スワップアウトされた内容が必要となった時にスワップインを行なう際、大量のメモリ転送が必要となる。ネットワークに対し負荷をかけないためにも相互の PE 間距離が小さいことが必要である。

このスワップアウト先 PE の探索に MetaShare を使用する。

3 従来の静的な構造を用いた MetaShare

MetaShare では情報管理構造として階層型構造をとる (図 2)。階層型構造を採用した理由は以下の通りである。

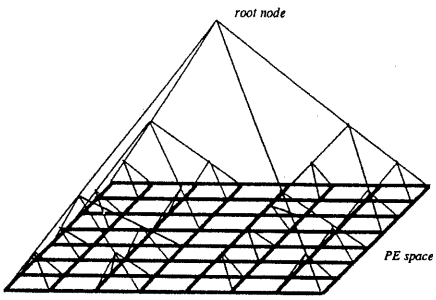

図 2: MetaShare の階層構造

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。

これまでに固定した階層型構造を用いたMetaShareとして以下の二つの方式を考案した。

1. 階層型構造を用い全てのPEから一つのルートに対し情報を集約し、そこから全てのPEに対し集約された情報を配布する方式 [3]。

   この方式ではまず全てのPEより階層型構造に沿ってルートに向かってPEアドレス、メモリ使用量の対を送る。この収集の過程で、全システム中でメモリ使用量の少ないPEを得るリダクションを行ない、メモリ使用量が少ないPEを幾つか選別する。こうしてルートにおいては全システムの中でメモリ使用量の少ないPEが得られる。次に逆の経路をたどってルートの持っているメモリ使用量の少ないPEの情報を各PEに伝える。これらの動作を一定時間毎に繰り返す。

2. 各領域個別にその領域の情報を集約し格納する方式 [4]。

   この方式ではまずシステム全体をネットワークのトポロジに沿って領域分割を行ない階層型構造を作る。各領域にはその領域の情報を管理する管理ノードをおき、領域の包含関係は管理ノードの木構造となる。上位管理ノードにおいては情報は下位領域毎に管理する。上管理ノードでは各領域の平均メモリ使用量を得て、この値を領域のメモリ使用量とする。各PEより一定時間毎にメモリ使用量を管理ノードに送り、領域の平均メモリ使用量を計算し、管理ノードの木構造にしたがって情報を伝搬させていく。

   探索は階層型構造の更新とは独立に行なう。各領域の中で最もメモリ使用量の少ない領域を代表する管理ノードを選別する。この動作をルートからリーフに向かって伝搬させる。行きついたリーフが求めるPEとなる。

これらの方式では以下の問題が生じる。

GVVMの効率を考えるとスワップアウト先PEとしてメモリの使用量が少なく、しかも通信距離が短いことが条件である [3]。1.の方式はシステム全体中のメモリ使用量の少ないPEが得られるが、これは通信距離に関しては全く考慮されていない。2.の方式ではメモリ使用量を管理ノードで平均するため周囲にメモリ使用量の少ないPEが存在してもそれが選択されるとは限らない。

また1.、2.の両方式において得られる結果は異なるが可能性があるが、各々の方式では探索は探索要求を出したPEによらず一意に定まる。スワップアウトの要求が同時に多数発生する場合には、同一PEに対しスワップアウトの要求が集中する可能性がある。

4 動的構造を用いたMetaShare

4.1 動作原理

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。

第一は平均メモリ使用量の均衡化を図ることである。GVVMはメモリ使用量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るMetaShareの階層構造にはメモリ使用量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使用量の均衡化を図る。各領域の平均メモリ使用量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使用量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使用量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使用量の少ない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。

メモリ使用量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使用量の均衡化を図るには、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、メモリ使用量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。

スワップアウト先のPEの探索は次のように行なう。

1. より小さい平均メモリ使用量を有する領域は、相対的にメモリ量の余裕があり他からのスワップアウトを受け入れられる可能性が高い。そこでスワップアウト要求を出したPEが属する全ての領域の平均メモリ使用量を各管理ノードより得、最小の平均メモリ使用量を有する領域を探査開始領域とする。

2. 探査開始領域から所属する最小のメモリ使用量を持つ領域をたどって最下位層まで下る。最下位層は1つのPEにより構成されるため、行きついた先がスワップアウト先として求めるPEである。

もちろんメモリ使用量の少ないPEは探索で得られたPE以外にもあり得る。しかしメモリ使用頻度を均衡化させるという意味で本方法により得られるPEがGVVMにおけるスワップアウト先PEとして妥当なPEである。

4.2 MetaShareの構成

本論文で提案するMetaShareは情報収集部、情報構造管理部、問い合わせ処理部の3つの独立して動作するモジュールにより構成される。

4.2.1 情報収集部

各PEのメモリ使用量を収集するモジュールである。各PEに1つづつ存在し、一定時間毎にページャよりメモリ使用量の情報を得て、情報構造管理部に伝える。

4.2.2 情報構造管理部

MetaShareの階層型構造の維持、管理を行なうモジュールである。

さらに情報構造管理部は、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。管理ノードの機能は3つある。

1. 領域の平均メモリ使用量、領域直径、領域に含まれるPEの数、属する下位層の領域などの領域情報を蓄積する。

2. 管理ノードの代表する領域に含まれ下位の領域の構成を管理する。具体的には下位層の各領域の平均メモリ使用量の均衡化を図るように、下位層の各領域を構成するPEを、収集された情報を元に決定する。その際領域に所属するPEはその領域の領域直径の範囲内にあることとする。

3. 問い合わせ処理部に対し蓄積している情報を提供する。

管理ノードはPE空間のあらかじめ定めたPEに固定的に分散配置する。

4.2.3 問い合わせ処理部

GVVMよりスワップアウト先PEの問い合わせを受け、探索を行ない、結果をGVVMに返すモジュールである。探索の基本動作は4.1の「動作原理」で述べた通りである。情報を得るためには領域の管理ノードに対し問い合わせパケットを送り、管理ノードはそのパケットに応じて中に蓄積している情報を送り返す。

4.3 情報構造管理部の動作

情報収集部及び問い合わせ処理部の動作は4.1の「動作原理」、及び4.2の「MetaShareの構成」において述べた通りである。ここでは情報構造管理部の詳細な動作について述べる。

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 領域の平均メモリ使用量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使用頻度は、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使用頻度を計算し、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

まず下位領域(level n+1)の平均メモリ使用量を比較する。各領域の平均メモリ使用量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送る。

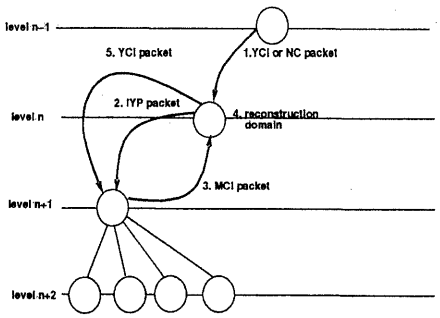

図3: 領域再構成フェーズのpacketの流れ(図中の数字は動作順序)

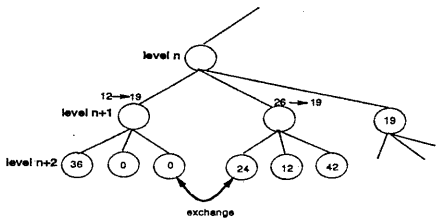

図4: 均衡化の例

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。送られきた二段階下位(level n+2)の領域の情報を元に情報を元に、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3)。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使用量を均衡化させること。例えば図4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。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. 4.1の「動作原理」で述べたように各領域に

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い、一段階下位の管理ノードからの距離と二段階下位の領域内に属するPEの内で二段階下位の管理ノードとの距離の最大値を加算し、その値が領域直径の1/2以内であればその二段階下位の領域は一段階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。

5 予備実験

5.1 予備実験の概要

並列マシンのシミュレータ上にMetaShareを作成して予備的な実験を行なった。実験の目的はGVVMを対象とするMetaShareが、収集してきた情報を元に構成する情報構造において負荷の均衡が図られていること、およびその情報構造を元に検索を行ないGVVMのスワップアウト先PEとして妥当なPE、即ちスワップアウト要求を出したPEに距離的に近くメモリ使用量の少ないPEを捜し出すことを確認することである。

実験はまず各PEのメモリ使用量の空間的パターンを与え、シミュレータ上のMetaShareを動作させた。得られた構造を用いて以下の測定を行なった。

1. 平均メモリ使用量の均衡化の様子を見るために各階層においての領域の平均メモリ使用量の分散を測定した。
2. 与えたパターンのPEの中からメモリ使用量の高いPE全てについてスワップアウト先の検索を行なった。検索の結果得られたPEのメモリ使用量、及び検索要求を出したPEとの距離を測定した。

作成したMetaShareは初期構造として3の「従来の静的な構造を用いたMetaShare」の2で述べた静的構造を用いたMetaShareと同一の構造を構築する。そこで初期構造の構築が完了した時点で比較のために同様な測定を行なった。

なおPE数は324、ネットワークは2次元トーラス(18x18)、メモリ使用量は正規化されているものとして0から1までの値をとるとした。領域直径はlevel 0とlevel 1の階層ではシステム全体をカバーする値(18)とし以下階層を下る毎に領域直径の大きさを1/2毎に小さくした。MetaShareの階層数は6とし、この時、初期構造は1つの領域が最大4つの下位領域により構成される。また、MetaShareは各領域において下位領域の平均メモリ使用量の偏差が0.05を越えた時点で、領域再構成を行なうように設定した。メモリ使用量が0.5を越えるとスワップアウトの要求が出るものとして2.の実験を行なった。

5.2 予備実験の結果および検討

実験にはPEの2つのメモリ使用頻度のパターンを使用した。

1. PE空間の左半分が1、右半分が0即ちシステムの右半分が非常に負荷が重く左半分は全く働いていない状態である。この状態での各階層における分散を表1に示す。

	level 1	level 2	level 3	level 4
初期構造	0.11	0.18	0.22	0.25
再構成後	0.00090	0.068	0.14	0.19

表 1: システムの半分で使用頻度が極端にわかれた時の動作時の分散

システムの両側で負荷が極端にわかれるため、初期構造(静的な構造体を用いた場合に相当する)は負荷の分かれる境界以外では各階層においてどの領域でも値が1か0に分かれる。従って領域間の均衡は全くなく分散が大きい。しかしMetaShareを動作させ負荷の均衡化を図ると各階層において特に上位の階層では極めて低い分散の値を得ることができ、十分な均衡化が行なわれていることが確認できた。領域が下がるにつれ一つの領域において均衡化の対象とするPEが少なくなるため均衡化は図りにくくなり、level 4においてはあまり均衡化の効果が出なくなった。

スワップアウト先PEの探索結果は初期構造、再構成後共にメモリ使用量が0のPEのみを探しだし、メモリ使用量の少ないPEを探索するという仕様は満たしている。スワップアウト先までの距離の平均は初期構造が9.9で再構成後が10.5となる。これはMetaShare

では負荷均衡が考慮されているので若干遠いPEもスワップアウト先として探索されるのに対し、初期構造では負荷の均衡が図られず、高いメモリ使用量を有するPEに近い一部のPEにスワップアウト先が集中するため結果的に平均距離が近くなることが原因である。

2. 図5のパターンを使用した。このパターンは現実のシステム動作の状況を模している。

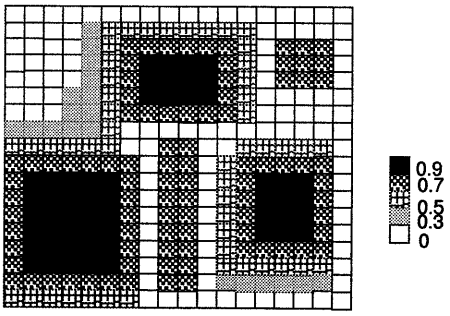

図 5: 実験に使用した各PEのメモリ使用量のパターン

各階層の分散の結果を表2に示す。

	level 1	level 2	level 3	level 4
初期構造	0.022	0.042	0.072	0.10
MetaShare	0.0020	0.013	0.051	0.072

表 2: 図5のパターンを使用した場合の分散

図5のパターンにおいても初期構造と比較してMetaShareの使用は各階層において領域の分散を低く押え、MetaShareによる均衡化の効果が生じていることが分かる。初期構造、再構成後共に検索により得られた全てのPEのメモリ使用量は0、即ち両者とも全ての検索結果はメモリ使用量の少ないPEを捜し出した。またスワップアウト要求元PEと検索結果PEとの距離の平均は初期構造の場合10.1、MetaShareの場合8.6となりMetaShareの方が距離的に近いPEを捜し出すことに成功した。これはMetaShareで動的な領域の再構成を行なうことにより、メモリ使用量の多いPEは各々そのPEの近くに存在するメモリ使用量の少ないPEを発見していることを表している。

6 おわりに

本論文では大域的仮想仮想記憶(GVVM)のスワップアウト先PEの探索を対象とし、システムの状況に適合して情報構造を動的に変化させる管理情報共有機構(MetaShare)を提案した。さらに並列計算機シミュレータ上においてプロトタイプを作成し、提案したMetaShareの有効性を予備実験により検証した。

現状では本論文で提案したMetaShareは開発中であるため動作効率は悪く、静的構造を用いたMetaShareと比較すると10倍程度の実行時間を要する。しかし、今後通信の効率化、種々のパラメータの調整、アルゴリズムの改良などを行なうことによりより効率化を図ってゆきたい。さらに並列計算機シミュレータ上で作成されたGVVMと接続し本格的な性能評価を行なっていく予定である。

謝辞

本研究はリアルワールドコンピューティング計画の一環として「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。熱心に討論していただいた塚本 享治 分散システム研究室室長に感謝する。

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, 17-24, 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.