超流動OSのための適合型情報管理機構の基本設計

4F－1

田沼 均　平野 聡　須崎 有康　一杉 裕志

電子技術総合研究所

1 はじめに

現在、百万台規模のPE構成をとる汎用超並列システム用オペレーティングシステム「超流動OS」を開発中である[1]。「超流動OS」の目標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を混在して実行させる環境下で、変化する並列度や粒度などに合わせ情報(プログラム、データ等)を流動させることにより、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。

ここでは超流動OSにおいて管理情報を管理する機構(MetaShare)について、管理構造を状況に応じて適合的に変化させる方式の設計について述べる。

2 MetaShareの目的

計算機資源やアクティビティを管理するには、現在システムがどのような状態にあるか適切に把握している必要がある。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。

またOSは通常、実行制御、記憶管理などいくつかのサブシステムにより構成される。各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負荷やネットワーク負荷、メモリ負荷といった共通するものも多い。

管理情報共有機構(MetaShare)は超流動OSが超並列システムを管理する上で不可欠な情報を収集、蓄積し、システムの状況の把握が必要となっているサブシステムやPEに対し適切な情報を効率的に配布することが目的である。

ここでは超流動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]をモデルケースとしてMetaShareを設計する。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システムである。ここでのMetaShareの役割は、ページアウト要求が生じた時、全PE空間上からGVVMが効率的に動作するような(メモリ使用頻度が低く、距離的に近い)スワップアウト先となるPEを捜し出すことである。

Basic Design of An Administrative Information Management System Using Adaptive Structure

Hitoshi TANUMA,Satoshi HIRANO,Kuniyasu SUZAKI,Yuji ICHISUGI

Electrotechnical Laboratory

3 静的な構造を用いたMetaShare

MetaShareは広大なPE空間上にある膨大な情報を対象としているため、情報の収集、蓄積、配布にあたり構造化する必要がある。全体を領域に分割し階層を下がるほどその領域を細分化する階層型構造を採用した。

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root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rootに向かってメモリの使用頻度を送る。この い収集の過程で、領域の情報が集約されるPEにおいてメモリ使用頻度が最低のものから一定数のPEをスワップアウトの候補PEとして選別する。rootにおいては全システムの中で最小のメモリ使用頻度のPEが得られる。次に逆の経路をたどってrootの持っている最小メモリ使用頻度のPEの情報を各PEに伝える。これら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。各領域の平均メモリ使用頻度を求め、上位ノードではこの値を領域のメモリ使用頻度として管理する。各PEより一定時間毎にメモリ使用頻度を管理ノードに送り、領域の平均メモリ使用頻度を計算し、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非同期に独自に探索を行なう。まず全体の情報があるrootの管理ノードより始め、各領域の中の最も低いメモリ使用頻度の領域の管理ノードよりその管理する領域の中の最も低いメモリ使用頻度の領域を選ぶ。この動作を繰り返してスワップアウト候補PEを探索する。

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にはメモリの使用頻度と共に通信距離が重要となる[3]。1.の方式はシステム全体中の最小のメモリ使用頻度を持つPEが得られるが、これは距離については全く考慮されていない。2.の方式では大局的にメモリ使用頻度の低い領域が選択されるため、周囲にメモリ使用頻度の高いPEが存在して領域の平均メモリ使用頻度が高くなるとたとえメモリ使用頻度の低い領域が近くに存在してもその領域は選択されない。

また1.、2.の両方式ともにシステム中のどのPEからの要求に対しても同じ探索結果を得る。このことは、超

並列システムにおいてはスワップアウトの要求が同時に多数発生することが予想されるため、一部のPEまたは領域に対しスワップアウトの要求が集中してしまう可能性がある。

情報収集は基本的に全PEに対して行なうためネットワークなどシステムに多くの負荷をかける。そこで情報収集を行なう間隔を長くとることが望ましいが、MetaShareの情報と実際のシステムの状況のずれが大きくなり不都合が生じる。MetaShareと実際の状況とのずれを大きな影響を及ぼさない範囲に押えるには十分頻繁に情報収集を行なう必要が生じる。

4 状況への適合性の導入

前節の問題を解決を図るために、メモリ使用頻度及び通信距離に適合させた管理構造を採用する。

管理構造は階層型構造とする。一つの階層はいくつかの領域に分割される。一つの領域はその下の階層のいくつかの領域を集めて構成される。最上位階層はシステム全体を表現し、最下位階層では一領域一PEで構成される。同一階層においては領域間の重なりはない。また領域に直下の階層の領域の数は固定されている。各領域にはその領域を代表する管理PEがあり、領域の情報を収集、蓄積、管理し、領域への問い合わせはこの管理PEが対応する。管理PEを領域の所属関係にしたがって結ぶと木構造となる。

各階層において各領域に所属するPEが一定距離内に収まっている範囲内で各領域同士の平均メモリ使用頻度が均衡するよう各階層の領域を分割する。各領域分割がメモリ使用頻度について均衡していれば、メモリ使用頻度の大きいPEが存在すると、それに対応してメモリの使用頻度の低いPEがその領域内に存在するためスワップアウト先を探すのに容易となる。各領域には属するPE間距離の上限が定められている。

具体的な領域決定の動作は次の通り。

1. 各PEはGVVMよりメモリ使用頻度を収集する。収集したメモリ使用頻度を自分の所属している領域の管理PEに伝える。

2. 下位の階層から順に各PEがどの領域に属するか決定してゆく。各領域が直上のどの領域に所属するかは二階層上の領域管理PEが決定する。二階層上(n+2層)の領域管理PEは直下(n+1層)の各領域の平均メモリ使用頻度を比較して、できるだけ等しくなるように(n層)の領域がどの(n+1層)に所属するかを組替える。例えばn層の各領域の平均メモリ使用頻度が図1に示されたようになっていたとすると、n+2層の管理PEはn層の平均メモリ使用頻度が0の領域と24の領域を交換し、n+1層での各領域間の平均メモリ使用頻度の均衡を図る。この時あまり遠いPEが同一領域内に入らないように、n+2層の管理PEはn+1層の領域に属する全てのPEがその領域のPE間距離の上限を越えないように組替えをおこなう。

1.、2.の動作は一定時間毎に行ない、領域間で負荷が均衡するよう管理構造を構築し直す。

スワップアウト先PEを検索するには検索要求を出したPEは自分が属する各階層の領域の管理PEに対し、その領域のPE間距離の最大値と平均メモリ使用頻度を問い合わせる。この値により近くにスワップアウト先として適当な領域があるか、それとも広範囲に探した方が良いか、探索開始領域を決定する。探索開始領域を決定したら、その領域内で平均メモリ使用頻度の少ない領域をたどって階層を下がり、スワップアウト先PEまたは領域を得る。

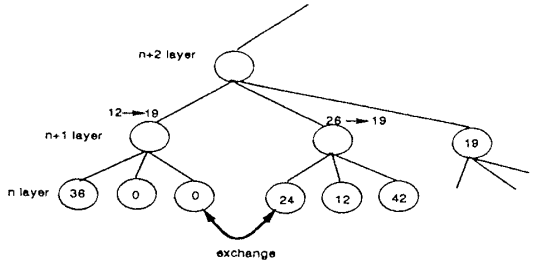

図 1: 領域の組替え

このように距離とメモリ使用頻度に適合させて階層型構造を構築すると次のような利点がある。

1. 探索を始める階層を選ぶことにより、スワップアウト先のPEの距離を考慮することができる。領域に属するPEはその距離以内のPEの全てではないが、領域に属していないPEは負荷分散の観点からスワップアウト先として不適切なPEである。またより近いところに適当なスワップアウト先があれば必ず見つけることができる。

2. 近傍、すなわち下位階層を優先して探索すれば、領域毎に個別のスワップアウト先を見つけるので分散してスワップアウト先を配置することができる。

3. 集中が起きないためMetaShareと実際の状況とのずれが生じても比較的影響が少なく、したがって情報収集の間隔を長くとれ、システムへの負荷を減らすことができる。

5 おわりに

状況に適合して蓄積構造を変化させるMetaShareの基本設計について述べた。モデルケースとしてGVVMで設計したが、本方式は負荷分散を行なうPEの再割つけなどの他のサブシステムにも応用可能と考える。現在シミュレータを作成して効果の検証を行なっている。

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。関係各位に感謝いたします。

参考文献

[1] 平野, 田沼, 須崎, 浜崎, 塚本. 超並列システム用オペレーティングシステム「超流動OS」の構想. 情報処理学会オペレーティング・システム研究会資料,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'93,1993

[3] 平野, 田沼, 須崎. 超流動OSのため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),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),1993.