計算機アーキテクチャ　118－2
(1996. 5. 16)

並列オブジェクト指向システムにおけるオブジェクト管理方式

西野 秀昭　平田 博章　新實 治男　柴山 潔

京都工芸繊維大学 工芸学部 電子情報工学科

京都市左京区松ヶ崎御所海道町

あらまし　本稿では超並列計算機上での並列オブジェクト指向言語における，効率的なオブジェクトの管理方式を提案する．本方式では，各要素プロセッサ (以下 PE) の負荷とオブジェクト間のメッセージ通信遅延とを考慮して，仮想的なオブジェクト間の相対距離を導出し，動的にオブジェクトを最適な PE に配置する．本方式では，オブジェクトの動的再配置を効率よく行うため，関連性の高い複数のオブジェクを molecule と呼ばれる単位で管理する．また，molecule 同士の併合および分割を行い，molecule に属するオブジェクトの集合も動的に最適化する．

キーワード　並列オブジェクト指向，動的再配置

A Parallel Object Management Scheme for Massively Parallel Computers

Hideaki NISHINO, Hiroaki HIRATA, Haruo NIIMI, Kiyoshi SHIBAYAMA

Department of Electronics and Information Science, Faculty of Engineering and Design, Kyoto Institute of Technology

Gosyokaido-cyo,Matsugasaki,Sakyo-ku,Kyoto,606

Abstract　This paper proposes a parallel object management scheme for parallel object-oriented languages on massively parallel computers. In this scheme, we define virtual distance between objects by taking account of load distribution amang processor elements and communication delays between objects, which is to be utilized to allocate each object to optimal location dynamically. For efficient dynamic re-allocation of objects, we manage much relevant objects as an unit called "**molecule**". Molecules themselves can be merged or subdivided dynamically, which will lead to an optimal allocation of objects.

key words　parallel object-oriented language, dynamic re-allocation

1 はじめに

分散メモリ型並列計算機においては並列オブジェクト指向言語の各オブジェクトでのメッセージの処理時間はオブジェクトの配置に依存する．したがって，オブジェクトの配置の最適化が必要になる．特に実際の処理時間に加え，各要素プロセッサ (以降 PE) 間の通信遅延の影響も考慮する必要がある．そのため，通信遅延を含めた評価が必要になる．

オブジェクトを 1ヵ所に集中して配置するとメッセージの通信遅延は減少する．しかし，その PE の負荷が高くなり，スループットを低下させる．逆に，オブジェクトを分散して配置すると，各 PE の負荷は分散される．しかし，メッセージの通信遅延が増大する．このように，負荷の分散と通信遅延はトレードオフの関係にあり，全体としてのスループットの改善のためには，両者を考慮して最適配置を求める必要がある．本報告では，そのような配置を動的に最適化するアルゴリズムを提案する．[2]

2 オブジェクト管理方式

2.1 メッセージの分類

オブジェクトが他のオブジェクトへ通信を行い，処理を要求することを「メッセージを送信する」といい，メッセージを受信したオブジェクトはそれに対応する処理を行い，必要に応じてその結果を呼び出し元のオブジェクトへ返送する．ここで，呼び出し元のオブジェクトの処理と並列に行われるような処理を要求するメッセージを**非同期メッセージ**，その処理が完了し，結果の返送を待ち合わせるものを**同期メッセージ**と呼ぶ．これらの違いを図 1に示す．

非同期メッセージでは実行軌跡が複数になっていることがわかる．

異なる PE 上のオブジェクト間でメッセージを交信すると通信遅延のためスループットは低下する．しかし，非同期メッセージによる処理は並列に行うことによって全体的なスループットの向上が期待される．また，同一 PE 上のオブジェクト間でのメッセージの交信を行った場合，通信遅延は小さいが，非同期メッセージによる処理で並列処理が期待される場合でも並列処理を行うことができず，スループットの向上は見込めない．

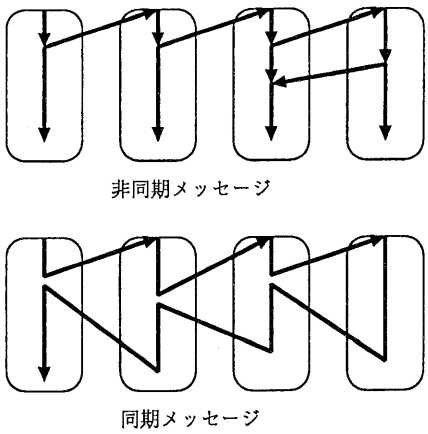

図 1: 非同期メッセージと同期メッセージの実行軌跡

この非同期メッセージによる動作を主体としたオブジェクトを**非同期オブジェクト**と呼び，また 同期メッセージによる動作を主体としたオブジェクトを**同期オブジェクト**と呼ぶ．

この分類は同期 / 非同期いずれかのメッセージしか受け取らない場合ににはオブジェクトの同期 / 非同期性も静的に決定され，非同期メッセージ，同期メッセージの両方の使用が考えられるオブジェクトの場合，非同期オブジェクトか同期オブジェクトかを静的に決定することができず，動的に決定せざるを得ない．

2.2 オブジェクトの配置

次に，オブジェクトの PE への配置を考える．まず同期メッセージを交信する場合，メッセージを送信したオブジェクトは，そのメッセージが送信先のオブジェクトに受信され，さらにそこでの処理が完了するまで，活動を休止することになる．このため，同期メッセージを交信する 2 つのオブジェクトは同じ PE に配置する方が，PE 間の通信が起らないため効率がよい．

非同期メッセージを交信する場合には, 通信遅延と, 処理時間で表される負荷量とを用いて, 以下の表1のように分類される.

したがって, 通信遅延と負荷量とを評価することにより, ある2つのオブジェクトを同じPEに配置すべきか, そうでないかを判断することができる.

表1: 非同期メッセージによる交信を行うオブジェクトの配置

	通信遅延 大	通信遅延 小
負荷 大	別途PEへ配置	別途PEへ配置
負荷 小	同一PEへ配置	——

そこで, この2つのオブジェクトの間に仮想距離という指標を導入する. オブジェクト間の距離はメッセージによる通信遅延に比例して小さくなり, 負荷に比例して大きくなると仮定する. ここで通信遅延を x, その比例定数を p, 負荷の量を y, その比例定数を qとすると, オブジェクト間の仮想距離 rは以下の式で表わされる.

$$r_1 = r_0 - pdx + qdy$$

こうして決定されたオブジェクト間の仮想距離をもとに, オブジェクトの配置を決定する. オブジェクトを同じPEに配置するか否かのしきい値を lとする. $r < l$ならば, 同じPEに配置する方が効率がよく, $l < r$ならば, オブジェクトを別々のPEに配置して並列処理を行う方が全体としての効率がよくなる.

また, 以下のことがいえる.

- 静的にオブジェクトが非同期か同期かが判明しない限り, この距離は静的に決定できない.
- 通信遅延と負荷を静的に決定することは困難である.
- ある処理段階で最適である配置が次の処理段階でも最適な配置であるとは限らない.

これらより, オブジェクト間の距離とそれをもとにしたオブジェクトの配置は動的に最適化されるべきであるといえる.

そこでまずオブジェクトの生成時には, 静的に決定したPEへ配置を行い, メッセージの交信状況を基に距離の再評価を行う. そして2つのオブジェクトの間の距離がしきい値 lを下回った時に, それらのオブジェクトを同一PE内に再配置する.

ここで再配置とは, あるオブジェクトが現在配置されているPEから別のPEに移されることである. このように, 動的にオブジェクトの再配置を行うことにより, オブジェクトの集合は常に最適に配置された状態に遷移する.

ただしこの方法では, 以下の図2のような状態のときに問題を起こす.

オブジェクトB, C, D, Eはすでに同一PEへ配置されているので, 同一PE上に存在している方がよいものとする.

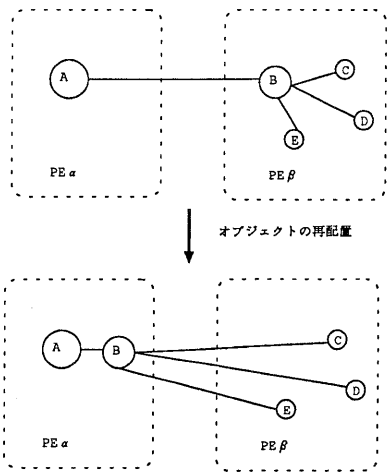

図2: オブジェクトのPE間の移動による問題

ここで, オブジェクトAとオブジェクトBの間で通信が増加し再配置がおこり, PE βのオブジェクトBはオブジェクトAのあるPE αへ再配置される.

これにより, オブジェクトBとオブジェクトC, D, E間にPEを隔てた通信が発生し, PE間の通信遅延が増加する. つまりここでは, オブジェクトBを再配置することによって, オブジェクトC, D, Eがオブジェクト Bと同一PEに配置されていたというメリットが失われている. またその後, オブジェクトBとオブジェクトC, D, E間の通信が続けれられるならば, そのたび

に仮想距離が小さくなり, C, D, E はいずれ B と同じ PE へ配置される. しかし, 最初にオブジェクト B, C, D, E は同一 PE に配置するのが最適であると分かっている. よって仮に, それらのオブジェクトを一度に移動するのに比べると, この場合はオブジェクトの移動が後回しになるため, 移動が行われるまで PE 間の通信量が増え, 全体としての通信遅延は大きくなる.

同時に再配置すべきオブジェクトを考慮してオブジェクト A とオブジェクト B のどちらを移動するかを決定することにより, この問題を軽減させることができる. しかし, オブジェクトの関連性まで考慮して処理を行うと, オブジェクトの再配置処理によるオーバーヘッドが無視できなくなる. このため, オブジェクトをその関連性にしたがって区分けする仕組みが必要である.

3 molecule

ここで, 再配置の起こる最小単位として **molecule** を導入する. これは, 関連性の深い複数のオブジェクトの集合である. オブジェクトを **molecule** に分割することによって, 複数のオブジェクトを 1 つの単位として見ることができ, 関連性の高いオブジェクトの移動を効率的に行うことができる. また, 距離計算処理をオブジェクト個々に行うよりも簡単化することができる.

molecule は最適配置を考えるための仮想的な単位である. 図 3 つまり, オブジェクト間のメッセージのやりとりは, **molecule** を導入しても何ら変わらない.

3.1 molecule の配置

次に, **molecule** の配置について考える. すべてのオブジェクトはいずれか 1 つの **molecule** に所属する. ある **molecule** に分類された各オブジェクトは, 同一 PE 上に配置される関連性の深いオブジェクトの集まりであるが, 各自がそれぞれ独自に他の **molecule** の任意のオブジェクトとメッセージの交信を行うことができる. つまりメッセージ処理は, **molecule** の導入によって何ら変化しない.

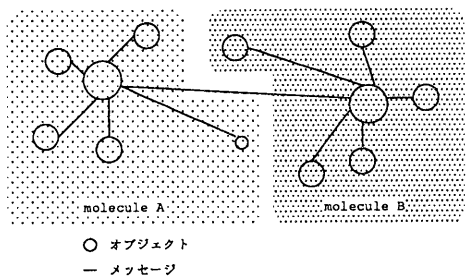

図 3: オブジェクトと molecule

ここで, オブジェクトに対して行ったように, **molecule** に対して仮想距離を導入する. **molecule** 内の各オブジェクトが他の **molecule** 内の各オブジェクトに対して行う通信遅延の合計を, その **molecule** 間の通信遅延とみなす, また他の **molecule** のオブジェクトからのメッセージによって生じた負荷の合計を, その **molecule** との負荷とみなす. ここで通信遅延を X, その比例定数を P, 負荷を Y, その比例定数を Q とすると, オブジェクト間の仮想距離の変化量ΔDは以下の式で表わされる.

$$\Delta D = -P\Delta X + Q\Delta Y$$

オブジェクトの場合と同じく, この **molecule** 間の仮想距離を用いて, **molecule** の配置を決定する. **molecule** を同じ PE に配置するかどうかのしきい値を L とおく. $D < L$ ならば, それは同じ PE に配置する方が効率がよく, また, $D < R$のときは, **molecule** を別々の PE に配置た方が全体としての効率がよくなる.

よって **molecule** 間の仮想距離が L を下回ったときに, その 2 つの **molecule** はどちらかの PE へ移動される. 関連性の深いオブジェクトがひとまとめになって移動するため, 先の **molecule** を導入しないモデルに比べ, いずれ再配置が起るであろう関連の深いオブジェクトを先にひとかたまりにして移動することができ, それまでの間の通信遅延の大きい PE 間の通信がなくなる. そのため, 全体的な効率がよくなる.

3.2 moleculeの併合

どのオブジェクトをどのmoleculeに配置するかについては，オブジェクト間の仮想距離と同様，静的に決定するのは困難である．

ここで，molecule自身も動的に最適化されるよう，「moleculeの併合」がおこるものとする．すなわち，molecule間の仮想距離Dがしきい値Fを下回った時，つまり$D < F$の時，2つのmoleculeに配置されているすべてのオブジェクトが1つのmoleculeに再配置される．

これにより，先述のオブジェクトの移動で問題になった，連鎖反応的なオブジェクトの移動は起らない．なぜなら，連鎖反応的な移動を起こす関連性の高いオブジェクトは，動的な併合によって1つのmoleculeに再配置されるからである．

しかし，併合によって関連性の高いmolecule同士が集まり，大量のオブジェクトを含むmoleculeに成長すると，その後の処理段階の変化などによって，molecule内に比較的関連性の低いオブジェクトが発生する可能性がある．この関連性の低いオブジェクトによって肥大したmoleculeは，逆に全体としての効率を下げる要因になる．

したがって，肥大したmoleculeの分割を適宜行う必要があり，そのためにはmolecule内部のオブジェクトの関連性を吟味するしくみが必要である．

3.3 moleculeの分割

再びオブジェクト間の仮想距離を考える．ただし，この仮想距離は同一molecule内のオブジェクト間だけを考え，他のmoleculeのオブジェクト間との仮想距離は考えない．前節と同様に，通信遅延をx，その比例定数をp，負荷をy，その比例定数をqとすると，オブジェクト間の仮想距離の変化量Δdは以下の式で表わされる．

$$\Delta d = -p\Delta x + q\Delta y$$

このmolecule内のオブジェクト間の距離がしきい値fを上回った時，つまり$f < d$の時，そのオブジェクトをmoleculeとして分割する．そしてさらに，そのオブジェクトとの距離がしきい値g以下のオブジェクトを，その新しいmoleculeに順次取り込む．

この処理を，「moleculeの分割」と呼ぶ．分割した後のmolecule同士は，相互のmolecule間距離によってその配置が決定される．

molecule内のオブジェクト間の仮想距離は，メッセージ交信の単位で適宜更新することができる．それによりオブジェクト間の関連性の深さを吟味し，必要に応じて分割を起こすことによってmoleculeへのオブジェクトの配置を最適化する．その結果，並列処理の効率を向上させ，全体的な効率を改善することができる．

molecule間の距離をしきい値と比較するだけのmoleculeの併合に対し，moleculeの分割は，まず内部に存在するオブジェクト間の仮想距離を全て比較し，新たに生成するmoleculeへ取り込まれるべきオブジェクトと既存のmoleculeへとどまるオブジェクトを分別するため，併合は分割に比べ処理が重いと言える．

3.4 静的配置

以上のような動的再配置を行うに当たって，moleculeおよびオブジェクトの初期配置が重要になってくる．

オブジェクトの生成されるとき，新たに生成されたオブジェクトは他とは独立したmoleculeに配置する．なぜなら，moleculeの併合に対し，moleculeの分割の方が処理が重いので，可能な限り分割を起こすことを避けるためである．

非同期オブジェクトの場合，さらにそれを生成したオブジェクトとは別のPEに配置されるように仮想距離の初期値を設定する．これは，先述のとおり非同期オブジェクトは独立したPEに配置した方が効率がいいからである．

3.5 定数の決定

先に述べた比例定数p, q, P, Qの決定方法はメッセージ交信時の通信遅延と負荷の平均を実験的に求め，それぞれの平均を代入したときに仮想距離の変化が起らない値に設定する．

4 おわりに

本稿で述べた方式では, 常に最適配置に遷移するため, 特殊な場合を除いて最適配置になる. また, 過去の履歴を利用して配置を行っているため, 最適配置になったときに処理の段階が進み最適配置が変化した場合などはかえって効率が低下する場合があるが, これは定数を調整することで解決することが出来る.

謝辞 本研究の一部は, 文部省科学研究費補助金・試験研究 (B)(1) No. 07558156 の補助による.

参考文献

[1] 石川裕, 所真理雄: オブジェクト指向並行プログラミング言語, 情報処理, Vol. 29, No. 4, pp. 325-333, 1988.

[2] 上原稔, 所真理雄: 計算場における分散プロセスの準最適配置, 情報処理学会論文誌, Vol. 36, No. 2, pp. 283-295, 1995.