プログラミング 14－2
（1997. 8. 21）

異粒度系における静的評価に基づくタスク割付方式

青柳 洋一, 上原 稔, 森 秀樹
東洋大学工学部情報工学科

不特定のシステムに対する並列処理のモデル化を行う上で、異粒度の処理レベルを組み合わせたタスク割り付けが有効である。また、タスク分割のモデル化を行う上で、依存関係の明確となるグラフ表現が解析に適している。本研究では、プリミティブレベルの並行処理単位とストリームの組み合わせでモデル化されるパイプライン処理を対象に、タスクのプロセッサへの割り付けアルゴリズムを提案する。割り付けにあたり、システム依存と問題依存の項目をパラメータ化し、スループットの予測により分割を決定する。

Task Allocation Method Based on Static Evaluation for Heterogeneous Grained System

Yoichi Aoyagi, Minoru Uehara, Hideki Mori
Department of Information and Computer Sciences,
Toyo University

It is valuable to combine heterogeneous grained processes for modeling parallel processing on various systems. And graph representation which makes apparent dependency among tasks is suitable for task allocation. In this article, we propose task allocation algorithm for pipeline processing which is composed with primitive level concurrent processes and streams. On allocation, we parameterize factors which depend on system and problems, and decide the allocation with throughput expectation.

1 はじめに

並列処理を効果的に行うためには、問題毎に分割方式を検討する必要がある。また、処理の最適化を図る上で、プロセッサ数やメモリ構成などハードウェア構成を意識したアルゴリズムが必要な場合がある。しかし、そういった個々の条件に依存する並列化方式では、問題とシステムの検討項目の組み合わせが無数にあり、処理分割の方式の一般化が難しい。

タスク分割をモデル化する上で、依存の明確化の図れるグラフ表現が適しており、タスクグラフを用いた実行時間の算出法 [3] がいくつか提案されている。本研究では並行オブジェクト群をストリームで結んだグラフで構成されるパイプライン処理を対象に、タスク分割アルゴリズムを提案する。

2 節で問題の定義、3 節で分割アルゴリズムの提案、4 節でアルゴリズムの有効性の検討、5 節でまとめを述べる。

2 モデルの定義

タスク分割の対象とする処理の形態、問題の定義について述べる。

2.1 タスク分割対象の処理の形態

本論文では、並行オブジェクトをストリームで結んでパイプラインを構成する並列処理を対象とした分割方式を提案する。ここで、オブジェクトはループ処理を行うプリミティブな並行処理単位であり、ストリームはオブジェクト間の通信路である。ストリーム通信はオブジェクトの演算処理とオーバーラップして行れるものとする。

プリミティブの組み合わせで処理を実現することで、任意のグルーピングを可能にし、並列化方式のアルゴリズムレベルからの変更は行わないものとする。

2.2 問題の定義

本稿で扱うタスク分割問題は、m 台の均質なプロセッサ集合 $P = P_i(i = 1, 2, ..., m)$ に n 個の並行オブジェクト $Obj_i(i = 1, 2, .., n)$ を振り分け、高スループットの得られる配置を求める問題とする。オブジェクトは、各々がループ処理を行う論理上の並行処理単位であり、オブジェクト間のデータ交換はストリーム通信によって行われる。

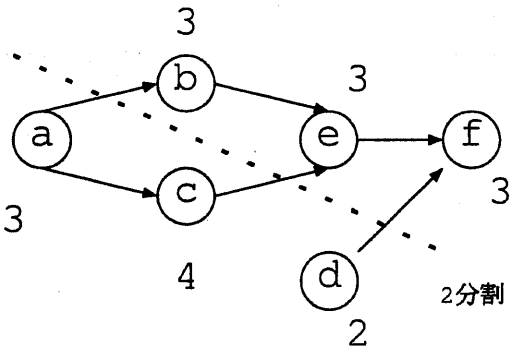

図 1: 並行オブジェクト群で構成される処理を 2 分割する例

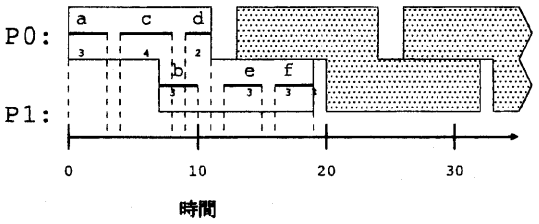

図 2: 2 分割した処理の実行のタイミング

図 1 は依存関係に循環の無いオブジェクト集合によるパイプライン処理の例を示している。ノードはオブジェクト、アークはストリームを表し、アークの向きはデータの流れる方向を示している。ノード i からノード j へのアークがある場合は、先行制約によりノード i の実行が行われるまでノード j の実行が待たされる。このとき、ノード i をノード j の直接先行ノードと呼ぶ。ノード内の記号はオブジェクト名であり、ノード脇の数値はオブジェクトの演算実行コスト (実行 1 回あたりの時間) を示している。

グルーピングされたオブジェクトは、グループで 1 つの大きなループ処理を行うタスクを形成する。繰り返し処理が対象となるため、1 回の実行時間の短縮よりも、繰り返し処理のスループットを上げることが重要となる。図 2 は、図 1 の処理を P0、P1 の 2 台のプロセッサへの分割したときの実行のタイミングを示している。囲み線は、ループ処理の 1 回の処理でプロセッサを占有する範囲を示す。このブロック状のタイミングで繰り返される処理の周期を短くするタスク分割を行う。

3 タスク分割

高スループットの得られるタスク分割を行うためには、プロセッサに振り分ける負荷に偏りが小さく、プロセッサ間の通信路の数が少ないことが重要となる。

オブジェクトのグルーピングは、グルーピングアルゴリズム [2] に従って行い、全体で行われる繰り返し処理の周期の最も短くなるグルーピングを選択し、プロセッサに配置する。

3.1 分割の決定に用いるパラメータ

多様に考えられる分割パターンの中から最もスループットの得られる分割を選択する。スループット予測に用いるパラメータは、システム依存のものと問題依存のものに分けられる。

システム依存のパラメータ

- プロセッサ台数 (>分割数)
- データサイズあたりの通信時間
- 各プリミティブオブジェクトの実行時間の平均

問題依存のパラメータ

- 各オブジェクトの種類
- 各オブジェクトの接続するストリーム
- 各ストリームの接続するオブジェクト

- 各ストリームの通信頻度、データサイズ

グルーピングパターンと実行順序のパターンについて実行の周期の予測を行い、最も周期の短くなる組み合わせを選択する。ここで、実行順序のパターンは、オブジェクトの先行制約に従って実行可能なオブジェクトの実行順序のリストである。

グルーピングの組み合わせは無数に考えられるが、その組み合わせの大部分は分割として不適当なものである。解を得るまでの試行回数を少なくするため、一定の法則に従って振り分けを行う。また、GA[1] の手法を取り入れて試行回数を少なくしている。

3.2 グルーピングの手順

1. パラメータを読みとり、プロセッサで受ける負荷の容量の上限 C をセットする

$$C = \sum(t_{exe}(i))/N_{pu} * a$$

$t_{exe}(i)$: オブジェクト i の演算実行時間
N_{pu} : プロセッサ数
a : 負荷容量に対する係数 (> 1)

2. 振り分け順序のリストをランダムに生成する
3. オブジェクトを通信の結び付きの強さに応じて振り分けてゆく
4. 各振り分け順序リストの中から、最も周期の小さくなるグルーピングの行われた振り分けリストを候補として選択
5. GA の規則に従い、実行の周期の予測値の値が変化しなくなるまで、交叉、変異、世代交代を行ってゆく

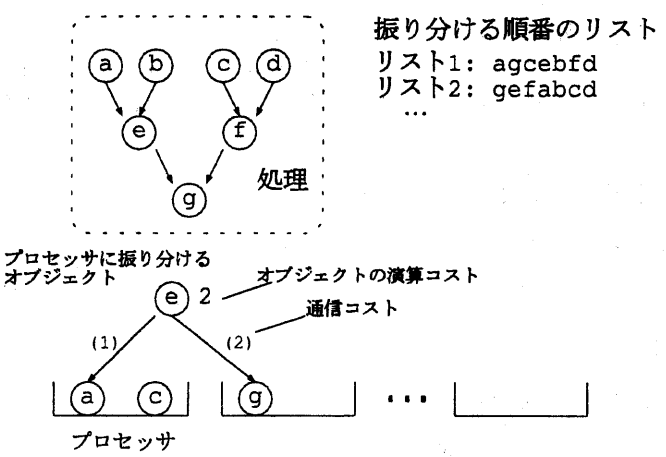

図 3: プロセッサへのオブジェクトの振り分けの過程

実行の周期を求める手順　上の手順でグルーピングされた処理についてプロセッサの占有時間を予測する。

1. 実行順序の組み合わせのリストと前述の手順で生成したグルーピングパターンを用意

2. 実行順リストに従ってオブジェクトの開始と終了のタイミングをセットしてゆく
オブジェクト i の実行開始の時刻 $t_{start}(i)$ は、j をオブジェクト $ い $ の直接先行オブジェクトとして

$$dmax = max_j(t_{end}(j) + t_{wait}(i,j))$$

$$t_{start}(i) = max(dmax, T_{end}(g) + t_{sw})$$

で求める。ここで、

$dmax$: (オブジェクト j の終了時刻 + i に対して生ずる遅延) の最も時間の遅いもの

$t_{wait}(i,j)$: データ転送遅延時間
オブジェクト i から j への通信コスト。異るプロセッサ上に配置された場合、遅延が生じるが、同一プロセッサ上の通信はメモリのコピー操作となるため、計算処理に対して無視できる (=0) とする

g : オブジェクト i の属するグループ

$T_{end}(g)$: グループ g で最後に実行されたオブジェクトの終了時刻

t_{sw} : オブジェクトの切替え時間
ローカル行われるオブジェクトの処理の切替えに生ずる時間

3. 実行順リストの中から実行時間の最小となる時間を周期の予測値とする

4 分割の有効性

分割アルゴリズムの有効性を示す。

木構造の処理の分割 15 個のオブジェクトで構成される木構造のグラフの 4 分割例を示す。オブジェクトの演算コストとして、平均 10 の指数乱数を与えた。オブジェクト間の通信コスト (遅延) は、プロセッサ間の場合 10、ローカルの通信は 0 とした。

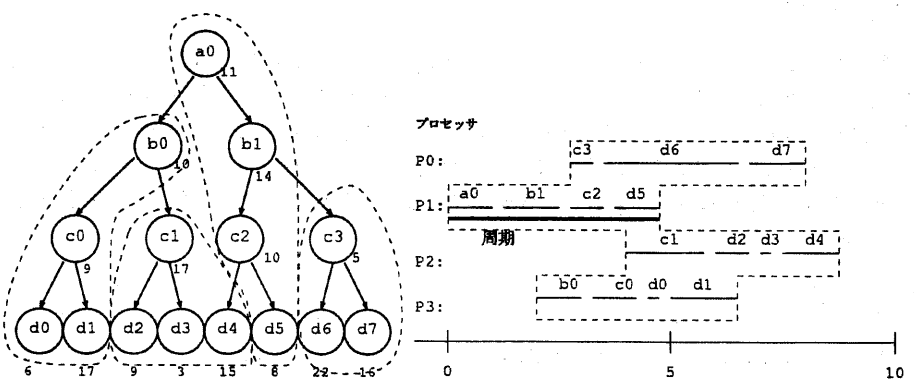

図 4: 木構造の処理を 4 分割する例と実行のタイミング

- 負荷がほぼ均等に分割され、プロセッサ内の処理に空きが少なくなるように分割された。

ランダムに生成したオブジェクトの分割　オブジェクトの総数を 50 とし、依存関係に循環が生じないようにランダムにストリームをセットした (ストリーム数=オブジェクト数*1.5)。オブジェクトの演算コストを平均 100 の指数乱数で与えた。パラメータとして、通信コストを 1 ～ 10000 まで変化させ、分割数を 1,2,5,10,20 としたときの実行周期の予測値のグラフを示す。

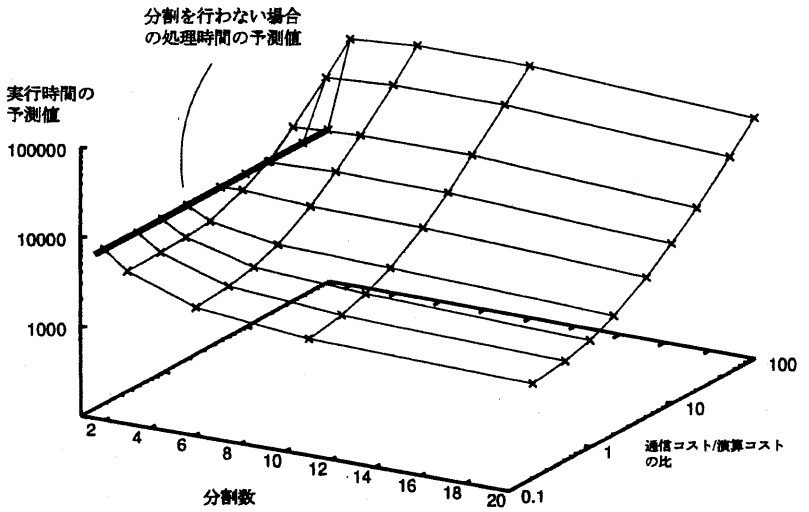

図 5: ランダムに生成した 50 個のオブジェクトの実行時間 (通信コスト、分割数をパラメータとした)

- 分割を行わない場合 (分割数=1) に対して並列化の効果が見られるが、通信コストの比を上げるにつれ、分割効果の低下が確認される。

5　まとめ

並行オブジェクトとストリームの組み合わせで実現されるパイプライン処理の分割アルゴリズムを提案した。多様に考えられる分割に対して、静的に得られるデータを用いた実行時間の予測により分割方法を選択した。アーキテクチャ依存のパラメータをあらかじめテスト実行により求めておくことで、並列システム一般に適用することが可能と考えられる。

今後、実際の多様なシステムにおける評価を収集し、パラメータの設定の補正などを行ってゆく。

参考文献

[1] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Progrtams.* Springer-Verlag, 1994.

[2] 青柳洋一, 上原稔, 森秀樹. ストリーム通信オブジェクトのプラットフォームに応じた分散. 情報処理学会研究報告, 1996. vol.96, No.82, pp.139-144.

[3] 李鼎超, 有田隆也, 石井直宏, 曽和将容. 非均質並列プロセッサ用プログラムの実行時間の下界. 情報処理学会論文誌, 1993. Vol.34,No.11,pp.2378-2385.