6E−4 実行開始条件による並列性検出手法—ループへの拡張

本多弘樹 (山梨大学), 笠原博徳 (早稲田大学)

1 はじめに

本稿では、Fortranプログラムのマクロタスク(粗粒度タスク)[1]レベルでの並列処理を自動的に行うシステムで必要となる、プログラム全域にわたるマクロタスク間並列性の自動検出手法について議論する。

プログラム全域にわたるマクロタスク集合の並列処理においては、プログラム中に条件分岐が存在するため、全てのマクロタスクが実行されるわけではなく、どのマクロタスクを実行すべきかが、条件分岐によってプログラム実行時に決定されるという特質がある。このため並列性検出手法は、全てのタスクが実行されるタスク集合を対象としたものとは異なったものとならざるをえない。

これに対して、マクロタスク間の並列性を実行開始条件として表現する並列性検出手法が既に開発されており[3, 1]、プログラム全域にわたっての並列性を検出・表現することが可能となっている。

しかしながら、これらの手法では、「並列性検出の対象とするマクロタスク集合のフローグラフが非循環であること」、という制約を前提としている。このため実際の並列処理において、プログラム中にループが存在する場合には、マクロタスクの構成や並列実行方式を工夫することにより、この前提を満たすようにする必要があった。例えば[2]では、1)ループ部分は最外側ループでひとつのマクロタスクとし、他の部分は基本ブロックごとにマクロタスクとすることによりプログラム全体のマクロタスク集合のフローグラフを非循環とする、2)最内側ループの各イタレーションのループボディ内部で複数のマクロタスクを構成し、ボディ部分のマクロタスク集合のフローグラフを非循環とする、3)このふたつの方式を階層的に組み合わせる、とういう方法で階層的に並列性検出を行ない、並列実行も階層的に行なうことにより、前述の前提を満たすようにしている。同様に[3]では、プログラムを階層的に分割しHierarchical Task Graph (HTG)として表現することにより、各階層でのフローグラフが非循環となるように工夫している。

一方、階層性を持たない任意のマクロタスク構成や、階層性を持たない並列実行方式による並列処理を実現するためには、前述の並列性検出手法を、マクロタスクのフローグラフにループが存在する場合にも対応させることが必要で、その実現が課題となっていた。

そこで本稿では、前述のマクロタスク間並列性自動検出手法で前提となっている制約、すなわちマクロタスクのフローグラフは非循環であることという制約、を除去するための方法を検討する。

以下では、まず、ループに対し階層性を導入しない並列実行モデルを検討した後、その並列実行モデルを実現するために必要となる実行開始条件を定義し、最後にまとめを述べる。

2 ループの並列実行モデル

以降の議論では、階層性の無いマクロタスク構成の一例として、プログラム中の全ての基本ブロックをマクロタスクとしたものを対象とする。

また、本稿ではループを以下のように分類する。

ループの種類:

1. 定回ループ:ループ実行開始時点で回転数がわかっているループ
   - 1-a. コンパイル時に回転数がわかっている定回ループ
   - 1-b. コンパイル時には回転数がわからない定回ループ
2. 不定回ループ:ループ実行開始時点で回転数がわかっていないループ

2.1 定回ループの並列実行モデル

定回ループではループが実行確定[1]となった時点で、回転数分の全イタレーションが実行確定となる。よって理論的にはループの全回アンローリングが可能で、並列性検出・並列実行において、ループの各イタレーション内マクロタスクをループ外マクロタスクと同等に取り扱った場合に得られる並列性が、ループ内マクロタスクに関する最大並列性と考えられる。実際にアンローリングせずにできるだけ最大並列性に近い並列性を得ることがループ並列処理の目標となる。これを実現するためには次の並列実行方式が可能でなければならない。

1. イタレーション間にまたがるマクロタスク間並列性を利用した並列実行。このためには、並列性検出に際しイタレーションとしての枠を取り払い、イタレーションにまたがるマクロタスク間の並列性を自動検出し、その並列性を実行開始条件として表現出来なければなければならない。

2. ループ内マクロタスクとループ外マクロタスクとの間の並列性を利用した並列実行。このためには、ループとしての枠も取り払い、ループ外マクロタスクとループ内マクロタスク間並列性を自動検出し、その並列性を実行開始条件として表現出来なければなければならない。

上記の方式を実現するに当たっては、従来の並列性検出手法およびそれを用いた並列実行方式と比較して、マクロタスク集合、実行開始条件の論理式の構成情報、実行開始条件の生成と登録の時期、に関して次のような違いが生じる。

マクロタスク集合: マクロタスク集合のフローグラフが非循環である場合には、マクロタスク集合に含まれる全要素はコンパイル時に決定している。しかし、フローグラフにループが存在し、そのループの回転数がプログラム実行時(以下、実行時)に決定される場合、マクロタスク集合を構成する要素も実行時に決定されることになる。

実行開始条件の構成情報: マクロタスクのフローグラフが非循環である場合には、他イタレーション内のマクロタスクや、ループ外のマクロタスクの実行情報が、イタレーション内マクロタスクの実行開始

Parallelism Detection Scheme with Execution Conditions for Loops
Hiroki HONDA (Yamanashi Univ.), Hironori KASAHARA (Waseda Univ.)

条件に含まれることはない。しかし、前述の二つの並列実行方式を実現するためには、他イタレーション内のマクロタスクや、さらにはループ外のマクロタスクの実行情報を実行開始条件に含めることができなければならない。

実行開始条件の生成と登録の時期：マクロタスク集合の全要素がコンパイル時に決定している場合には、全てのマクロタスクの実行開始条件をコンパイル時に生成することができる。また、実行開始条件成立の評価を行なうスケジューラへ、実行開始条件を評価対象として登録することもコンパイル時に行なうことができる。しかし、マクロタスク集合の構成要素が実行時に決定される場合には、実行開始条件を実行時に生成する必要が生じる。さらに実行開始条件のスケジューラへの登録を実行時に行なう機構も必要となる。

以上を踏まえ、定回ループの並列実行モデルとして、以下の3モデルを考える。

並列実行モデル1—ループとしての一括性は無し: ループをアンローリングした場合と同様に、ループの各イタレーション内のマクロタスク（以下ループ内マクロタスク）をループの外のマクロタスク（ループ外マクロタスク）と同等に取り扱う。ループ内マクロタスクの実行開始条件は、他イタレーション内のマクロタスクやループ外マクロタスクの実行情報を含むこととなる。

並列実行モデル2—ループ枠の実行開始条件のみ一括管理: ループ枠の実行開始条件のみを一括管理する。ループ内マクロタスクの実行開始条件は、他イタレーション内のマクロタスクやループ外マクロタスクの実行情報を含むが、ループ枠に関する条件を含むことはない。ここで、ループ枠の実行開始条件とは、ループの実行確定条件 [1] とループ制御変数設定関連のデータアクセス可能条件 [1] で構成される実行開始条件とする。

並列実行モデル3—ループ全体一括管理: ループ全体の実行開始条件を一括管理する。ここで、ループ全体の実行開始条件とは、ループの実行確定条件とループ制御変数設定関連のデータアクセス可能条件、および、ループボディ内変数のデータアクセス可能条件で構成される実行開始条件とする。ループ内マクロタスクの実行開始条件は、他イタレーション内のマクロタスクの実行情報を含むが、ループ外のマクロタスクの実行情報を含むことはない。

2.2 不定回ループ並列実行モデル

不定回ループでは、あるイタレーションを実行するか否かはそれ以前のイタレーションによって決定され、ループが実行確定となったとしても、その時点で各イタレーションは実行確定とはなるわけではない。このため、各イタレーションが実行確定となった時点で、そのイタレーション内マクロタスクの実行開始条件の生成・登録をしていくこととなる。

不定回ループの並列実行モデルは、定回ループの並列実行モデル2および3に対応して、次の2つのモデルが考えられる。

並列実行モデル1—ループ枠の実行開始条件のみ一括管理:
ループ枠の実行開始条件が成立した後に、順次各イタレーションが実行確定となった時点で、そのイタレーション内マクロタスクの実行開始条件の生成・登録をしていく。ループ内マクロタスクの実行開始条件は、他イタレーション内のマクロタスクやループ外マクロタスクの実行情報を含むことになる。

並列実行モデル2—ループ全体一括管理: ループ全体の実行開始条件が成立した後に、順次各イタレーションが実行確定となった時点で、そのイタレーション内マクロタスクの実行開始条件の生成・登録をしていく。ループ内マクロタスクの実行開始条件は、他イタレーション内のマクロタスクの実行情報を含むが、ループ外マクロタスクの実行情報を含むことはない。

3 実行開始条件の拡張

ループ内マクロタスク MTi の実行開始条件を、各並列実行モデルに応じて次のように定義する。

ループとしての一括性無しの場合:
同一イタレーション内、他イタレーション内およびループ外に存在する、MTi の全てのデータ依存先行マクロタスク [1] の集合を $DDMT[MTi]$ とし、マクロタスク MTi の実行開始条件を、つぎのように定義する。

MTi の実行開始条件 =
(ループの実行確定条件)
$\wedge$(イタレーション内での MTi の実行確定条件)
$\wedge(\wedge^{allMTk \in DDMT[MTi]}$($MTk$に対するデータアクセス可能条件))

ループ枠の実行開始条件のみ一括管理の場合:
同一イタレーション内、他イタレーション内およびループ外に存在する、MTi の全てのデータ依存先行マクロタスクの集合を $DDMT[MTi]$ とし、マクロタスク MTi の実行開始条件を、つぎのように定義する。

MTi の実行開始条件 =
(イタレーション内での MTi の実行確定条件)
$\wedge(\wedge^{allMTk \in DDMT[MTi]}$($MTk$に対するデータアクセス可能条件))

ループ全体一括管理の場合:
同一イタレーション内および他イタレーション内に存在する、MTi の全てのデータ依存先行マクロタスクの集合を $DDMT[MTi]$ とし、マクロタスク MTi の実行開始条件を、つぎのように定義する。

MTi の実行開始条件 =
(イタレーション内での MTi の実行確定条件)
$\wedge(\wedge^{allMTk \in DDMT[MTi]}$($MTk$に対するデータアクセス可能条件))

なお同一イタレーション内または他イタレーション内のマクロタスク MTk に対するデータアクセス可能条件は、マクロタスク終了条件と非実行確定条件 [1] とにより、次のように定義する。

MTkに対するデータアクセス可能条件 =
(MTkの終了条件)
$\vee$(MTkを含むイタレーション内での MTkの非実行確定条件)

4 おわりに

実際の並列処理において、前述のどの並列実行モデルを採用するかは、ハードウェア性能やスケジューラの実行開始条件評価の方式による。イタレーションやループとしての一括性を取り払い階層性を無くすということは、タスクの粒度を細粒度化しているととらえることができ、どのモデルを採用するかは、タスクサイズの大小による並列性とオーバーヘッドのトレードオフの問題に帰着する。

なお本研究の一部は、文部省科学研究費補助金奨励研究 (A) 04750316 による。

参考文献

[1] 本多弘樹, 岩田雅彦, 笠原博徳. Fortran プログラム粗粒度タスク間の並列性検出手法. 信学論, J73-D-1(12):951–960, 12 1990.

[2] 本多弘樹, 合田憲人, 岡本雅巳, 笠原博徳. Fortran プログラム粗粒度タスクの OSCAR における並列実行方式. 信学論, J75-D-1(8):526–535, 8 1992.

[3] M. Girkar and C. D. Polychronopoulos. Automatic extraction of functional parallelism from ordinary programs. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):166–178, Mar. 1992.