4J-02　データ依存のみをもつ任意形状のマクロタスクグラフに対するデータローカライゼーション手法

成清 暁博[†], 八木 哲志[†] 松崎 秀則[††], 小幡 元樹[†] 吉田 明正[†††], 笠原 博徳[†]

[†] 早稲田大学理工学部電気電子情報工学科,[††] (株) 東芝,[†††] 東邦大学理学部情報科学科

1　はじめに

本論文では，ダイナミックスケジューリングを用いた粗粒度タスク並列処理において，複数の粗粒度タスク間での共有データをローカルメモリ（LM）経由で授受するデータローカライゼーション手法 [1][2][3] の拡張手法を提案する．本手法は，データ依存のみを持つ任意形状のマクロタスクグラフ全体または部分グラフに適用し，定数または変数上下限値ループに対して，ローカルメモリサイズやキャッシュサイズを考慮したデータの分割・配置を行うことにより，より多くの粗粒度タスク（MT）間データ授受を LM 経由で行えるようにし，プロセッサ間データ転送オーバーヘッドを軽減して速度向上を図る．

2　階層型粗粒度タスク並列処理

粗粒度タスク（MT）は，擬似代入文ブロック（BPA），繰り返しブロック（RB），サブルーチンブロック（SB）の 3 種類から構成される [4]．MT は，プロセッサクラスタ（PC; PE を論理的にグループ化したもの）に割り当てられて並列処理される．このとき，PC に割り当てられた MT 内部では，ループ並列化，近細粒度並列処理，あるいは，サブ MT が定義され階層的に粗粒度並列処理が適用される．コンパイラは，MT 生成後，MT 間，あるいは，サブ MT 間のコントロールフローとデータフローを解析し，階層型マクロフローグラフ [4] と，各 MT の最早実行可能条件を表す階層型マクロタスクグラフ（MTG）[4] を生成する．MTG をもとに，粗粒度タスクを実行時に PC に割り当てるダイナミックスケジューリングルーチンを，各階層ごとに生成する．

3　任意形状データ依存マクロタスクグラフにおけるデータローカライゼーション手法

各階層で粗粒度並列処理を行なう際，データ転送オーバヘッドを軽減し，効率良い並列処理を実現するためには，データローカライゼーションを行なうことが必要である．本手法では各階層において，以下の手順でデータローカライゼーションを行う．

1. 複数の MT 間データ転送をローカルメモリ経由で実現するために，着目階層のマクロタスクグラフ全体または部分グラフとして，ローカルメモリ経由で転送を行う MT 集合（ターゲットループグループ; TLG）を生成する．
2. コンパイラに，分散・配置すべきローカルメモリサイズまたはキャッシュサイズが指示された場合，TLG 内の配列変数使用量を解析し，適切なループ分割数を決定する．
3. 拡張されたループ整合分割法を適用し，定数，または，変数上下限値ループの処理とデータを，LM 経由のデータ授受が行なえるように分割する．
4. パーシャルスタティックタスク割当を用いたダイナミックスケジューリング [2][3] により，多量のデータ転送を必要とする MT 集合を実行時に同一プロセッサクラスタ（PC）にダイナミックスケジューリングする．
5. 同一 PC に割り当てられる MT 集合に対して，LM 経由のデータ授受を可能とするデータ転送コードを生成する．

3.1　変数インデックス上下限値をもつループの整合分割

ループ整合分割法は，データ依存があり，ループインデクス上下限値が定数または変数である複数ループ（RB）を，LM 経由データ授受を実現できるように，LM サイズや配列の参照添字範囲を考慮して最適な分割を行なうものである．

3.1.1　ターゲットループグループ（TLG）生成

ループ整合分割では，各階層のマクロタスクグラフ（MTG）から，ループ整合分割の対象となる部分 MTG を選ぶ．この部分 MTG のことを，ターゲットループグループ（TLG）と呼ぶ．TLG の生成は，データ依存グラフにおいてデータ転送量が最大のエッジで接続された 2 つの MT 集合から始め，その TLG から伸びるエッジのうち条件を満足する MT を TLG に繰り返し追加していくことで行なわれる．ひとつの TLG が生成されると，同一階層中で TLG に含まれなかった残りのマクロタスク集合について同様の処理を，グループが生成できなくなるまで繰り返す．図 1 (a) に TLG の例を示した．ループ上下限値が変数の場合，データ転送量も変数であるため，データ転送量はシンボリックに比較し，比較できるものを TLG に含める．

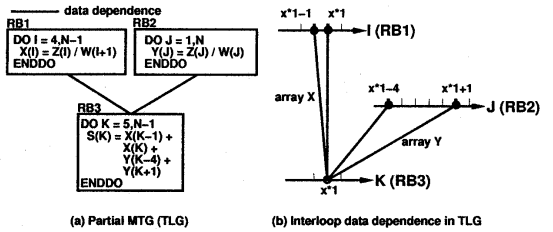

図 1: 変数上下限値ループ集合から構成されるターゲットループグループ（TLG）．

3.1.2　TLG 内ループ間データ依存解析

次に，各階層で生成された各 TLG（m 個の RB で構成されているものとする）内部において，ループ間データ依存を解析する．このループ間データ依存は，TLG の出口ノード（出口ノードが複数存在する場合は，そのうちの 1 つのノード）RB_m（TLG 内で m 番目の RB，標準ループと呼ぶ）のイタレーションから，TLG 内の各 RB_i（TLG 内で i 番目の RB，$1 \leq i \leq m-1$）のイタレーションへの直接的あるいは間接的（他 RB を経由した）データ依存を意味する．図 1 (a) の TLG における依存解析情報は図 1 (b) のように表される．

3.2　メモリサイズ，キャッシュサイズを考慮した分割数の決定

ユーザーはコンパイラに対して，ループ分割数の代わりに，分割メモリサイズまたはキャッシュサイズを指示することがで

---

* A Data Localization Scheme for Any Macrotasks with Data Dependencies
Akihiro Narikiyo[†],Satoshi Yagi[†], Hidenori Matsuzaki[††],Motoki Obata[†], Akimasa Yoshida[†††],Hironori Kasahara[†]
[†]Department of Electrical, Electronics and Computer Engineering, Waseda University [††]Toshiba Corporation [†††]Dept. of Information Science, Toho University

きる．コンパイラは分割数を決定するために，まず，TLG 内のすべてのループで使用されるデータ量を調査し，データが指定されたサイズに収まるループイタレーション数 t を解析する．ループ上下限値が定数の場合は，ループ回転数（総イタレーション数）を t_{total} とすれば，分割数 n は，$n = \lceil \frac{t_{total}}{t} \rceil$ で求めることができる．ループインデクス上下限値が変数の場合は，ループ回転数が不明であるため，コンパイル時に分割数を求めることが不可能である．そこで，ここでは考え得る最大の分割数 n_{max} を与えることにする．最大の分割数を考えるためには，考え得るループ回転数の最大を考えればよい．ループインデクス変数で配列を操作するループを対象としているため，ループ回転数の最大は，TLG 内で定義・参照されるすべての配列について，その各々の次元のうち，最大の定義サイズになると考えられる．その定義サイズを t_{max} とすると，最大の分割数は，$n_{max} = \lceil \frac{t_{max}}{t} \rceil$ で与えられる．

3.2.1 TLG 内変数インデックス範囲ループの整合分割

コンパイラは，まず標準ループ RB_m を均等に n 分割し，部分標準ループ（RB_m^p）を生成する．次に依存解析情報をみて，TLG 内の他のループ $RB_i (1 \leq i \leq m-1)$ を分割する．このとき，単一の部分標準ループがデータ依存する RB_i のイタレーション集合を Localizable-Region; LR（RB_i^p）と定義し，複数の部分標準ループが共通にデータ依存している RB_i のイタレーション集合を Commonly-Accessed-Region; CAR（$RB_i^{<p,p+1>}$）として生成する．

図 1 (a) のように，ループ上下限値が変数の場合，これを 3 分割すると，図 2 のように，分割後のループ上下限値は線形式で示される．コンパイラは，後述する LM 経由データ授受を適用する配列の検出のための解析情報としてこれらの線形式を保存し，ループの実行開始の直前で初期値と終値を求めるためのオブジェクトコードを埋め込む．また，分割数が実行時に決定される場合，分割された MT の直前に，分割数を決定するための MT を追加する．この MT の実行時に分割対象ループの回転数 t_{total} が得られるので，分割数 n は，$n = \lceil \frac{t_{total}}{t} \rceil$ として実行時に決定できる．ループはすでに，コンパイル時に最大の分割数 n_{max} で分割されているため，実行時には，$(n_{max} - n)$ 個の MT は不要である．そのため，実行時にダイナミックスケジューラに通知し，不要なマクロタスクの割り当てを行わないようにし，後続マクロタスクが待つべき MT の数を $(n_{max} - n)$ だけ減らすようコード生成を行う．

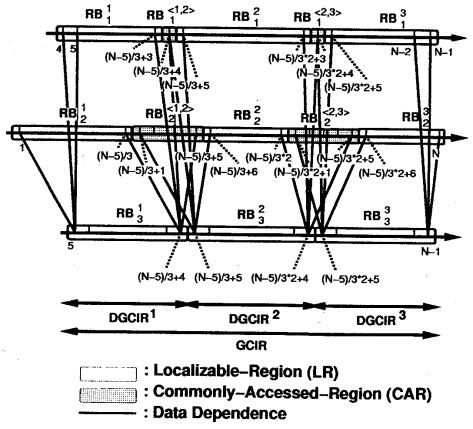

図 2: ループ整合分割によるループインデクスの分割．

3.3 パーシャルスタティックタスク割当を用いたダイナミックスケジューリング

整合分割後の MT において，スケジューリングオーバーヘッド削減のため，CAR は隣接する LR に融合する．次に，データ転送が多い MT 集合をデータローカライゼーショングループ（DLG）と定義し，DLG 内のある MT が最初に割り当てられる際に，DLG 内の他のマクロタスクが同じ PC に割り当てられるよう指定する．

3.4 データ転送コード生成

DLG 内部で，各配列変数ごとに，データローカライゼーション適用時に必要となるデータ転送を解析する．配列の定義・参照範囲解析は，シンボリックに行われる．シンボリックな解析は時として困難であるため，イタレーション空間において参照領域のエッジが不規則な形状をもつ場合，それらを内包する最小の矩形参照範囲に置き換えることにより，解析を単純化する．次に，解析された転送範囲をもとに，配列 X について，ローカルメモリ（LM）経由，集中共有メモリ（CSM）経由のそれぞれのデータ転送時間をシンボリックに求めて比較し，ローカルメモリ経由でのデータ転送が有利と判断された場合にデータローカライゼーションを適用する．データローカライゼーションが適用される配列については，DLG 内における当該配列への定義・参照範囲を考慮して，必要最小限のメモリ容量を LM 上に確保する．DLG 内では，データローカライゼーションの適用される配列変数に対して，各 PE 上の LM へのロード・ストア命令を生成する．また，データローカライゼーションの適用される配列要素が，DLG 外の MT と共有される場合には，CSM 経由でデータ授受を行なう転送コードを生成する．

4 性能評価

性能評価に用いる OSCAR[5] は，LM を持つ 32 ビット RISC プロセッサ（PE）を，集中共有メモリ（CSM）に 3 本のバスで接続した分散・集中共有メモリ型マルチプロセッサシステムである．OSCAR では，PE 上の LDM へのロードやストアに 1 クロックを要し，CSM へのロードやストアには 4 クロックを要する．本データローカライゼーション手法を伴う階層型粗粒度タスク並列処理を，変数上下限値ループをもつ SPEC95 ベンチマークの tomcatv プログラムを用いて評価した結果，データローカライゼーションを伴わない場合に比べて，2PE で 16.4%，4PE で 14.6%，6PE で 16.6%の速度向上を確認した．

5 おわりに

本論文では，階層型粗粒度タスク並列処理におけるデータローカライゼーション手法において，変数上下限値ループを含む，データ依存のみを持つ任意形状のマクロタスクグラフを対象とし，サイズを考慮した整合分割により多くのデータ転送を，LM 経由で実現するデータローカライゼーション手法を提案した．OSCAR シミュレータ上での性能評価により，提案したデータローカライゼーション手法は，階層的に粗粒度並列処理される粗粒度タスク間データ転送を軽減し，集中共有メモリ経由データ授受を行う階層型粗粒度タスク並列処理に比べて顕著な性能向上を得ることを確認した．

参考文献

[1] 吉田明正, 前田誠司, 尾形航, 笠原博徳: Fortran マクロデータフロー処理におけるデータローカライゼーション手法, 情報処理学会論文誌, Vol. 35, No. 9, pp. 1848–1860 (1994).

[2] Kasahara, H., Yoshida, A.: A Data-Localization Compilation Scheme Using Partial-Static Task Assignment, Journal of Parallel Computing, Vol. 24, No. 3, pp. 579–596, (1998).

[3] 吉田明正, 越塚健一, 岡本雅巳, 笠原博徳: 階層型粗粒度並列処理における同一階層内ループ間データローカライゼーション手法, 情報処理学会論文誌, Vol. 40, No. 5, pp. 2054–2063, (1999).

[4] 岡本雅巳, 合田憲人, 宮沢稔, 本多弘樹, 笠原博徳: OSCAR マルチグレインコンパイラにおける階層型マクロデータフロー処理手法, 情報処理学会論文誌, Vol. 35, No. 4, pp. 513–521 (1994).

[5] 笠原博徳, 成田誠之助, 橋本親: OSCAR のアーキテクチャ, 電子情報通信学会論文誌 (D), Vol. J71-D, No. 8 (1988).