5ZB-8

自動並列化コンパイラ MIRAI における最適化機能に関する一考察 －並列化のためのループ再構築手法について－

佐川 靖彰† 峰尾 昌明† 上原 哲太郎‡ 齋藤 彰一†† 國枝 義敏‡‡

和歌山大学大学院システム工学研究科† 京都大学大学院工学研究科‡
和歌山大学システム工学部†† 立命館大学情報理工学部‡‡

概要

ソフトウェア分散共有メモリシステム Fagus を実行時環境とする自動並列化コンパイラ MIRAI を開発している．この両者の組み合わせにより，分散メモリ型並列アーキテクチャにおけるデータの自動分割，すなわち初期分散配置の決定および実行時の再配置の自動化を目指す．今回，MIRAI コンパイラに Fagus を考慮したループ再構築部を実装する上で有効な手法の調査，検証を実施した．その結果，有効な手法を発見したので報告する．

1. はじめに

我々は，PC/WS クラスタをはじめとする分散メモリ型を含む幅広い並列アーキテクチャを対象とする自動並列化コンパイラ MIRAI[1](MultI-gRAIn automatic parallelizing and distributing compiler ; 以下，MIRAI コンパイラ)を開発している．同コンパイラの現版が生成するコードは，我々が開発したソフトウェア分散共有メモリ(Distributed Shared Memory : DSM)システム Fagus[2](以下，Fagus)に特化している．

MIRAI コンパイラはループレベルの並列化を実現しており，本研究では並列度を向上させるループ再構築手法について検討を行う．

2. MIRAI コンパイラと Fagus

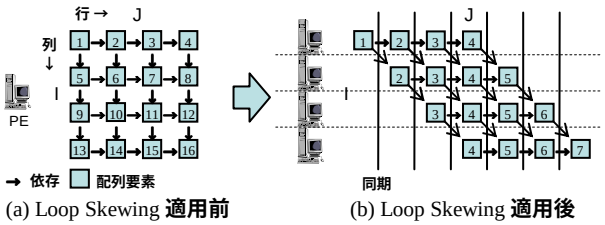
MIRAI コンパイラは，統一的中間表現により，依存解析や最適化を行い，最終的に Fagus のユーザ関数を挿入して並列化した C 言語の目的コードを出力する．Fagus は，メモリ空間を 4096 バイトの固定サイズ単位（以下ページと呼ぶ）に分割し，キャッシュの単位としている．配列を共有する場合，MIRAI コンパイラは行方向にデータを分割し，行ごとに 1 つの同期変数と関連付ける．このとき，1 行のサイズがページの整数倍となるように丸め込みを行う．共有変数へのアクセスのたびに必要となる排他制御は，当該共有変数に関連付けられた同期変数の獲得と解放のための Fagus の処理を，アクセスの前後に挿入することで行う．

3. ループ再構築手法の検討

本研究では，現 MIRAI コンパイラのループ再構築部[1]で，今まで並列化の対象にできない依存のあるループに注目し，並列実行可能となるように依存を解消する手法について検討する．

```
DO I=2,99
  DO J=2,99
    A(J,I) = ( A(J-1,I)+A(J+1,I)+
               A(J,I-1)+A(J,I+1) )/4
  ENDDO
ENDDO
```

図 1: プログラム例



(a) Loop Skewing **適用前** (b) Loop Skewing **適用後**

図 2: loop skewing による並列化

例として図 1 で考える．2 重ループの両方に，依存距離[3]が 1 のループ繰越し依存[3]が存在するため，このままでは並列化できない．しかし，ループ再構築手法の 1 つである Loop Skewing 手法[3]を適用すると，図 2(a)の構造から(b)の繰り返し構造に変えることができ，I のループでの並列化が可能となる．具体的には，I の各繰り返しを PE(PC クラスタのときは，単一 PC)に分担させる．各 PE は，図 2(b)の配列要素の番号順にアクセスしなければならない．同じ番号は並列実行可能部分であることを表す．よって，Loop Interchange 手法[1]を適用し，I と J のループの入れ替えも併せて行う．

Fagus での並列実行には以下の処理を施す必要がある．図 2 の依存を示す矢印は，矢印元の配列要素の計算結果を矢印先の配列要素の計算に使用する必要があることを表す．よって，配列要素 A(J,I)の計算の前に，隣接する PE が計算した最新の配列要素 A(J,I-1)を含むページと A(J,I+1)を含むページを取得する必要がある．これに対し，配列要素 A(J-1,I)と A(J+1,I)を含むページは，自 PE が有しているので，新たに取得する必要がない．前述の実行順序は J ループのイタレーションごとに同期を取ることで保証する．よって，同期の回

A Consideration of Loop Restructuring Techniques for an Automatic Parallelizing Compiler, MIRAI

† Yasuaki sagawa , Masaaki Mineo・Graduate School of Systems Engineering, Wakayama University

‡ Tetsutaro Uehara ・ Graduate School of Engineering, Kyoto University

†† Shoichi Saito ・ Faculty of Systems Engineering, Wakayama University

‡‡ Yoshitoshi Kunieda ・ College of Information Science and Engineering, Ritsumeikan University

数は行の長さ分必要であり，1 度の並列実行により計算できる量は 1 列分である．同期変数の獲得と解放，最新の計算結果を取得する回数は，全 PE 合わせて配列要素の計算の回数分必要である．また，他の PE が所有している最新の結果を取得する場合は同期変数の解放を待つ．これらの処理は計算時間と比べて非常に時間がかかるため，極力減らすことが並列実行時の速度向上につながる．そこで，あらかじめ Loop Tiling 手法[3)] を適用してから Loop Skewing を適用することを試みた．2 重ループにおいて Loop Tiling と Loop Skewing によって依存を解消し，並列化可能にできる条件は，依存ベクトル[3)] において，その成分の方向ベクトル[3)] がＩとＪともに負でないことである．

Loop Tiling を適用すると，図 2 の配列要素を小行列に分割できる．タイルサイズを大きくする(仮に 2 倍にする)ことにより，上記の並列実行のためのオーバーヘッドを低減する(理論上は約 1/2 とする)ことができる．しかし，タイルサイズを大きくすると，図 2(b)の配列要素番号 1，7 のように，全 PE で並列実行できない部分の割合の増加につながるため，並列実行による効果を十分に引き出すことができなくなる．したがって，これらを踏まえた上で適切なタイルサイズの決定が必要である．

4. 評価と考察

今回取り上げた手法の有効性を調査するため，4096×4096 の配列サイズのプログラムを用いて Fagus 上で速度計測実験を行った．まず，Loop Skewing のみ適用したプログラムの速度計測を試みた．しかし，2〜6 台の PE での並列実行では処理が終了しなかった．この原因は，前章で述べたループ J のイタレーションごとの同期，通信オーバーヘッドのためだと予想される．実際，隣接する PE からすべての配列要素の結果を取得するのにかかる時間は 4096×4096 の配列サイズで約 3.2 秒かかった．よって，1 行のページ転送時間は 3.2÷4096 秒である．手法適用後のデータ転送量は，1 つの配列要素の計算で 2 行分のページ(第 I-1 行と第 I+1 行)転送を行い，それが配列要素の計算回数分の 4094×4094 生じることになる．よって，総データ転送時間は約 3.2÷4096×2×4094×4094 秒(約 7 時間)と見積もれる．処理が終了しない一因としてデータ転送に多くの時間が費やされていることが明らかとなった．

そこで，正しいプログラムではなくなるが，隣接する PE の最新計算結果を取得せずに実行した結果を図 3 に示す．処理は終了するものの，それでも逐次実行より遅い．

次に，Loop Tiling を適用し，並列粒度を高めてから Loop Skewing を適用した結果を図 4 に示す．データ転送の時間を削減するため，6 台で実行できる最大のタイルサイズ(36 分割，1 タイルサイズ:683×683)にして計測した．図 4 からわかると

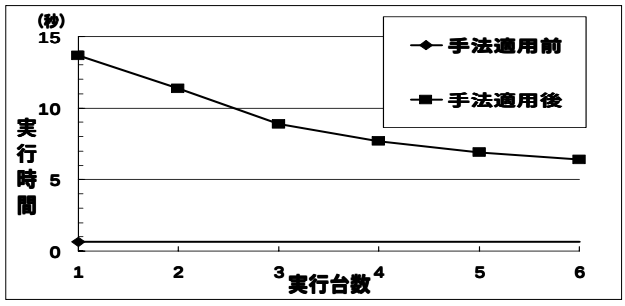


図 3: Loop Skewing 適用結果(データ転送なし)

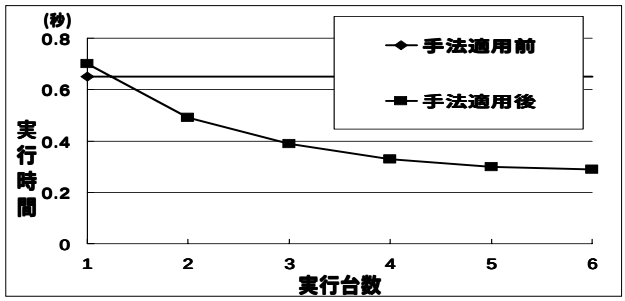


図 4: Loop Tiling と Loop Skewing 適用結果

おり，手法適用前より速度が向上した．データ転送は，各タイルの最初と最後の行の計算開始のみ隣接 PE から転送されるため，タイル数 36 個分生じる．よって，総データ転送時間は約 3.2÷4096×2×36 = 0.05625 秒程度と見積もれる．また，同期や同期変数の獲得と解放にかかる処理も 1/683 に削減される計算になる．このように並列実行にかかる処理を大きく削減したことで並列効果が得られたと思われる．したがって，両手法の組み合わせは並列度が向上し，速度の向上も期待できると結論づけることができる．

5. おわりに

今回，依存を解消して並列度を向上するためのループ再構築手法として Loop Skewing と適切なサイズでの Loop Tiling を組み合わせる方式を取り上げ，実験結果からその有効性を示した．よって，この両手法の組み合わせはループ再構築部の発展に向けて積極的に取り入れるべきである． しかし，それには言うまでもなく Loop Tiling において，最も効果的な分割を行えるタイルサイズの決定方法の確立が必要である．

参考文献

1) 信原裕文，峰尾昌明，上原哲太郎，齋藤彰一，國枝義敏，“PC クラスタを対象とするループレベル並列化機能を有する MIRAI コンパイラにおけるループ再構築部の実装”,情報処理学会研究報告，2004-EVA-8，pp.7-12 (2004.3).

2) 横手聡，林章仁，齋藤彰一，上原哲太郎，國枝義敏，“コンパイラによる制御が可能な DSM システム Fagus の実現，”情報研報，Vol.2000，No.75，2000-0S-85，pp.47--54 (2000.8).

3) 中田 育男,“コンパイラの構成と最適化，”朝倉書店(1999).