計算機アーキテクチャ 137－29
ハイパフォーマンス 80－29
コンピューティング
（2000．3．3）

マルチグレイン自動並列化のための解析時インライニング

吉井謙一郎† 松井巌徹†† 小幡元樹†
熊澤慎也† 笠原博徳†

本論文では，マルチグレイン自動並列化コンパイラにおけるインタープロシージャ解析の高度化を目指し，解析時のインライン展開，並列性解析，フレキシブルクローニングを組み合わせたプロシージャ間並列性解析手法を提案する．本手法は，まず従来のインタープロシージャデータ依存解析手法で粗粒度並列性及びループ並列性が解析できなかったサブルーチンコールに対し，解析を目的としたインライン展開を施し並列性解析を行う．そしてサブルーチン内外にわたるグローバルな並列性が存在すればそのままインライン展開したコードを生成し，部分的に並列性が存在すればその部分はインライン展開したままにして存在しない部分を“フレキシブルクローニング”すなわち元のあるいは別な形のサブルーチンに変換する．この並列性解析手法により，生成コードの過度な増加を抑えつつグローバルな粗粒度及びループ並列性を有効に引き出すことができる．マルチプロセッサシステムOSCARのシミュレータ上でベンチマークプログラムARC2Dを用いて行った性能評価の結果，提案手法により，従来のアトムイメージ法による解析のみの場合に比べて速度向上率を約3倍以上改善するという結果が得られ，またコードサイズでも3%の増加にとどまっていることが確認された．

An Analysis-time Procedure Inlining Scheme for Multi-grain Automatic Parallelizing Compilation

KEN'ICHIROU YOSHII,† GANTETSU MATSUI,†† MOTOKI OBATA,†
SHINYA KUMAZAWA† and HIRONORI KASAHARA †

This paper proposes an interprocedural parallelism analysis scheme which combines analysis-time inline expansion and flexible cloning for multi-grain parallelization. The analysis-time inlining is applied to subroutines to which the previous interprocedural analysis scheme like “Atom Image” cannot analyze parallelism to exploit global parallelism. After the analysis of global parallelism over procedures, compiler generates inlined code for program part having global parallelism or applies “flexible cloning” to program parts without global parallelism into the original shape or different shape of subroutine. With this scheme, the compiler can exploit global coarse-grain and loop-level parallelism with minimum increase in the code size.

Performance evaluation using benchmark program ARC2D on a simulator of multiprocessor system OSCAR shows the proposed scheme gives us three times larger speedup than the popular Atom Image method with only 3% increase in the code.

1. はじめに

マルチプロセッサシステム上のFortran自動並列化コンパイラにおいては，従来よりループ並列化手法[1)2)]が広く用いられている．しかし，ループ並列化手法では単一ループのイタレーション間並列性しか利用できず，今後のプロセッサ数の増加と共にスケーラブルな実効性能向上が困難であると考えられる．この問題に対処するために，基本ブロック，ループ，およびサブルーチンコール等の粗粒度タスクレベルの並列性を利用するためのアプローチとして粗粒度タスク並列処理（マクロデータフロー処理）[3)]，ループイタレーション間の中粒度並列処理，ステートメント間の近細粒度並列処理を階層的に組み合わせてプログラム全域の並列性を利用するマルチグレイン並列処理が提案されている[4)]．この粗粒度タスク並列処理においては，サブルーチン内外の粗粒度タスク間並列性を抽出するためにインタープロシージャ解析が必須となる．インタープロシージャ解析は，ループ並列性を抽出するためにも従来から多くの手法が提案されており，配

† 早稲田大学 理工学部 電気電子情報工学科
Department of Electrical, Electronics and Computer Engineering, School of Science and Engineering, Waseda University
†† 松下電器産業（株）
Matsushita Electric Industrial Co., Ltd.

列添字の詳細とループ範囲をコールサイトへ伝搬するアトムイメージ法[5)6)]や，ループ境界とループストライドからサブルーチン内での参照配列部分を決定しコールサイトへ伝搬する *Bounded Regular Section* 解析法[7)]，コールサイトコンテキストの相違による解析精度の低下を防ぐためにサブルーチンを選択的にクローニングする手法[8)]などが提案されている．しかし，アトムイメージ法では実引数及び仮引数整合配列の次元が異なる場合サブルーチン側での情報をコールサイト側での情報に変換できず解析できない．また，*Bounded Regular Section* 解析法では大きくコールサイトコンテキストの異なるものが存在する場合に解析精度が低下し，選択的クローニング手法ではコードサイズが増大するなどの問題がある．これらの問題を解決するためにアトムイメージ法と配列の一次元化を組み合わせた手法[9)]も提案されているが，粗粒度タスク間並列性の抽出は考慮されていない．そこで本論文では，アトムイメージ法のような既存の解析手法を用いても粗粒度タスク間並列性の解析が行えないプロシージャコールに対して，コンパイラにおける解析時にのみインライン展開し並列性抽出を行った後，並列性及びコードサイズを考慮し元のサブルーチンあるいは他の形のサブルーチンにフレキシブルに戻し，解析精度を向上させつつコードサイズの過度な増大を抑える手法を提案する．

2. マルチグレイン並列処理

OSCAR マルチグレイン自動並列化コンパイラ[4)]では，プログラムを次の 3 種類のマクロタスク (MT) に分割する．

BPA　基本ブロック及びこれを複数融合したブロック
RB　最外側ナチュラルループ
SB　サブルーチンコール

ここで粗粒度タスク並列処理とはこの MT 間の並列性を用いた処理手法である．また中粒度並列処理とはループイタレーション間の並列性を用いた処理であり，近細粒度並列処理とは BPA 内の各ステートメントをタスクと定義してそのタスク間の並列性を用いた並列処理である．そして，この 3 つの粒度の並列処理を組み合わせたものがマルチグレイン並列処理である．

各ネスト階層の MT は階層的に定義されたプロセッサクラスタ (PC) 間で並列処理される．また各階層で PC に割り当てられた MT は，さらに PC 内のプロセッサエレメント (PE) あるいは次階層の PC により階層的に並列処理される．例えば MT が RB で DOALL ならば中粒度並列処理を，ループボディが大規模で基本ブロックから成る逐次ループの場合は近細粒度並列処理を，また RB が大規模逐次ループである場合にはループボディ部に対し階層的に粗粒度タスク並列処理を適用する．

本論文では，サブルーチン内外で効果的な粗粒度タスク並列処理を行うためのインタープロシージャ解析法を取り扱う．

3. 解析時インライニングによるインタープロシージャ解析

本節では，提案する解析時インライニングを用いたインタープロシージャ解析法及びその結果を効果的に使用するためのフレキシブルクローニングについて述べる．

3.1 適用する SB の選定とインライニング

解析時インライニングは従来の代表的なインタープロシージャ解析手法であるアトムイメージ法で粗粒度並列性の解析ができなかった SB に対して適用される．解析時インライニングを適用する SB の選定は以下のように行う．

1) アトムイメージ法で解析できなかった SB の中で，コンパイラによる推定コスト（タスク処理時間）が最大のもの
2) 1) の SB とデータ依存が存在する SB

ただし，2) ではアトムイメージ法で解析できた SB も解析時インライニングの対象として選ぶこともある．これは，1) で選ばれた SB から呼び出されるサブルーチン内部の MT とデータ依存を持つ SB から呼ばれるサブルーチン内部の MT との間でグローバルな粗粒度並列性を抽出できる可能性があるためである．この条件を満たした SB は並列性解析のために中間語レベルで展開される．

3.2 並列性解析

この解析時インライニングの使用により，引数情報の明確化，コールサイトコンテキストを用いた定数，シンボリック値の伝搬，無用コードの削除などを簡単に行うことができ，インライン展開を用いない従来のインタープロシージャ解析手法に比べ，より多くの粗粒度タスク間並列性を抽出することが可能となる．また粗粒度並列性の解析時には，OSCAR マルチグレイン自動並列化コンパイラの最早実行可能条件解析[10)3)]も使用する．

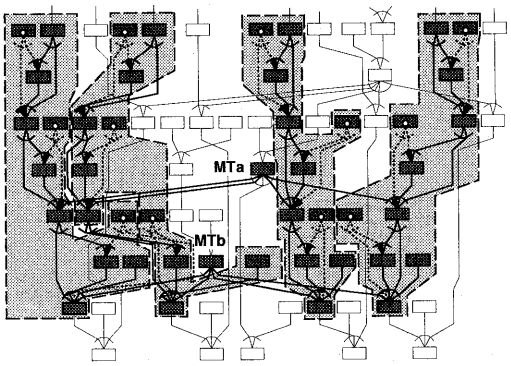
3.3 フレキシブルクローニング

並列性解析の終了後，このインライニングによってより多くの並列性を抽出できたかどうかを検査する．この検査においては，最早実行可能条件解析の結果から作成されたマクロタスクグラフ (MTG) に対してその MTG の並列度 Para[11)] を計算し，展開前の Para と比較することにより並列性が増加したかを判断する．並列度 Para は以下の式で表される．

$$\text{Para} = \frac{\text{1PC での逐次処理時間}}{\text{MTG のクリティカルパス長}}$$

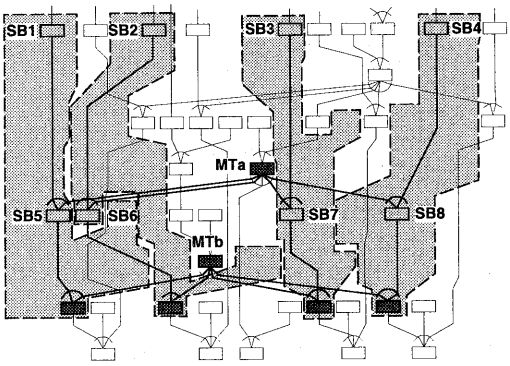
次に，展開によりできた各 MT について並列性の向上に寄与しているかどうかを検査する．これは，Para

が当該階層 MTG 全体における並列度を示すものであるため，全体として Para が増加していたとしても個々の SB の展開でできた MT の中には Para の増加に寄与していないものが存在する可能性があるからである．この検査によって，並列性増加に寄与していないと判断された MT 集合はフレキシブルクローニングにより元のサブルーチンあるいは異なる形のサブルーチンへ変換される．



内部のマクロタスクは解析時インライニングにより生成されたマクロタスク群

図 1　解析時インライニング直後の MTG の例



部はフレキシブルクローニングにより生成されたサブルーチンブロック

図 2　図 1に対しフレキシブルクローニングを適用した後の MTG

ここでは，フレキシブルクローニングのための部分並列性解析とクローニング例を図 1と図 2を用いて説明する．図 1は Perfect Club Benchmark 中のプログラム ARC2D におけるサブルーチン INTEGR の MTG の一部であり，サブルーチン FILERX を解析時インライニングした後 4 回転のループをアンローリングした場合を表している．図中点線で囲まれた 4ヶ所の部分は上記のアンローリングされた 1 イタレーション内の MT 群を表している．これらの MT 群は，1RB 中のループボディ部の MT であったが，解析時インライニングとアンローリングにより，以前はサブルーチン外にあった多くの MT との並列性がありまた MTa, MTb とのデータ依存が明確に解析できていることがわかる．

そこで解析時インライニング後に生成された MTG を出口ノードから辿り，解析時インライニングを施した結果生まれた MT 群の中で以前はサブルーチン外であった MT との間でデータ依存が生じているものを捜す．前述のように，図 1の例では各網掛け部分には外部の 2 つの MT（MTa と MTb）からの依存が生まれている．そこで新しく依存が生じた MT を入口ノードとした MT グループを生成し，そのグループの MTG（部分 MTG）に対して部分的な Para の値を計算し，この部分 MTG が全体の MTG の Para の増加に寄与しているかどうかを調べる．図 1の場合は，その部分が元々ほぼ一直線の部分 MTG であり Para が約 1.00 と並列性がないので，全体の MTG の Para の増加に寄与していないと判断される．従って，この部分 MTG はフレキシブルクローニングの対象になりサブルーチンへと変換される（SB1～SB8）．このフレキシブルクローニングを適用した結果は図 2の様になり，全体の並列性を維持したまま MTG が簡素化されたことがわかる．クローニングの際，クローニングする部分が同じコードを共有しているのであればクローニングの結果生まれるサブルーチンは一つだけとなり，インライン展開の副作用であるコード量の増大を最小限に抑えることができる．

このフレキシブルクローニングにより，従来のインタープロシージャ解析ではサブルーチン単位でのみしか粗粒度並列性の解析ができなかったのに対し，サブルーチン内部の MT を上位階層 MT へ変換することにより，従来まではサブルーチン内部に隠されていたグローバルな並列性を引き出すことができる．

4. 解析の適用例

この解析の適用例を Perfect Club Benchmark 中の ARC2D を用いて説明する．実際の解析においては，本手法を適用する SB を選定する前に解析をする場所を決定する必要がある．これにはコンパイラにより推定されたタスクコストが用いられる．すなわち，効果的な並列処理を実現するために推定コストが最も大きい部分を選びその部分に対して本手法を適用することになる．プログラム ARC2D の場合はサブルーチン INTEGR がコンパイラにより選ばれた．これは，実際実行においてもサブルーチン INTEGR は実行時間の約 95%を占めることから適切な選択であることがわかる．図 3にサブルーチン INTEGR の MTG を示す．

4.1 解析適用 SB の決定とインライニング

3.1 節で示した基準によりサブルーチン INTEGR 内

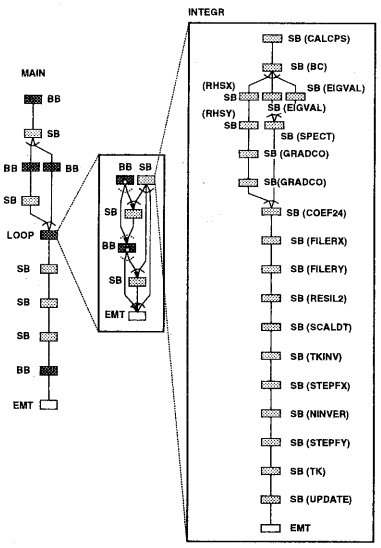


図3　サブルーチン INTEGR の MTG

でこの解析を適用する SB を選定する．この場合，サブルーチン STEPFY を呼び出す SB が 3.1 節の 1) に合致する．これは以下の検査結果によるものである．

- STEPFY 内から YPENTA を呼び出す部分が図 4 のプログラムのようになっており，STEPFY 側実引数 S が 3 次元配列であるのに対して対応する YPENTA 側仮引数整合配列 F が 2 次元配列である．このため，アトムイメージ法ではこの配列に関する情報を YPENTA 側から STEPFY 側へ伝搬することができず解析できない．そして下位階層で解析できなかったことがこの SB に伝搬されているため，この SB についてもアトムイメージ法では解析できない．
- 推定コストが INTEGR 全体の約 23%を占めており，アトムイメージ法で解析できなかった SB のうち最も大きい．このため，この SB に対して解析を適用するのが最も効果的である．

```
SUBROUTINE STEPFY(.....)
DIMENSION S(JDIM,KDIM,4)
DO N=2,4
 .....
 IF(N.EQ.2)THEN
  CALL YPENT2(..,S(1,1,N),..)
 ELSE
  CALL YPENTA(..,S(1,1,N),..)
 ENDIF
ENDDO

SUBROUTINE YPENTA(..,F,..)
DIMENSION F(JDIM,KDIM)
```

図4　サブルーチン STEPFY における YPENTA の呼び出し

さらに選定を進めると，INTEGR 内の残りの SB は全て 3.1 節の 2) に合致し解析時インライニングの対象となる．そして解析時インライニングの対象となった SB は中間語レベルでインライン展開される．さらに 1 回目の展開後もう一度選定を行うと，サブルーチン STEPFY を展開してできた SB である図 5中の MT10 と MT15 が 3.1 節の 1) に合致して選ばれ，また図 5中の MT5 も 3.1 節の 2) に合致し再度インライン展開が行われる．この 2 回目の展開後再度選定を行うと，既にこの階層にはサブルーチンコールは 1 つもなく選定及び解析時のインライン展開は終了する．

```
C -- MT5(SB) --
  CALL YPENT2(..,S(1,1,2),..)
.....
C -- MT10(SB) --
  CALL YPENTA(..,S(1,1,3),..)
.....
C -- MT15(SB) --
  CALL YPENTA(..,S(1,1,4),..)
```

図5　図 4のサブルーチン STEPFY をインライン展開することにより上位階層に現れた YPENTA 及び YPENT2 の呼び出し

4.2　並列性解析とフレキシブルクローニング

インライン展開後，コールサイトコンテキストを使ったシンボリック値（定数を含む）の伝搬や無用コードの削除など並列性増加のための各種最適化を行った後，データ依存解析及び並列性の解析を行う．

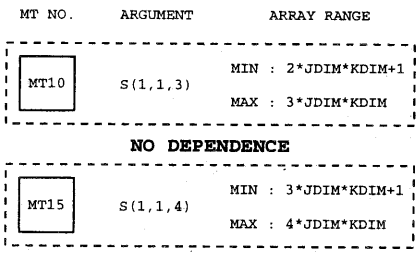


図6　図 5中の MT10 と MT15 の YPENTA の呼び出しに関して一次元化した添字を用いたデータ依存解析結果

サブルーチン INTEGR の場合，図 4及び図 5に示したアトムイメージ法では解析できない STEPFY から YPENTA の呼び出しが重要である．そこで，図 5中の MT10, MT15 について図 6のように一次元化した配列添字を検査すると，MT10 では配列 S の 2×JDIM×KDIM+1 から 3×JDIM×KDIM 番目の要素を参照し，MT15 では配列 S の 3×JDIM×KDIM+1 から 4×JDIM×KDIM までの要素を参照することがわかり依存がないことが判明する．しかしアトムイメージ法では 4.1 節で述べたようにサブルーチン YPENTA 内の S に関する参照情報をサブルーチン STEPFY 側に伝搬することができずこの解析結果を得ることができないのである．

これに対し解析時インライニングを用いる本手法で

は，繰り返し解析時インライニングを適用することによりサブルーチン STEPFY はインライン展開され，またサブルーチン YPENTA 及び YPENT2 もインライン展開されている．この結果，インライン展開により一次元化された配列添字を検査することができるため，図 6と同様の解析結果が得られ，図 5中の MT5, MT10, MT15 をインライン展開してできた MT 群同士には依存がないことがわかる．

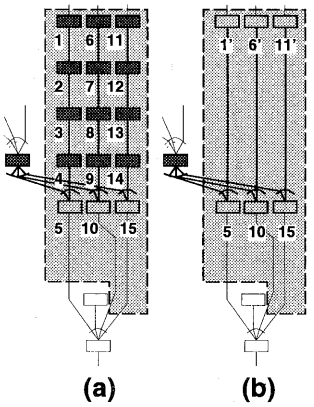


図 7　STEPFY 相当部分に対するフレキシブルクローニングの適用

さらに，3.3 節で述べたフレキシブルクローニングを適用すると，まず YPENTA 及び YPENT2 を展開した部分がフレキシブルクローニングされ図 7(a) 中の MT5, MT10, MT15 の様に元のサブルーチンを呼ぶ形になり，次に図 7(a) の MT1 から MT4，MT6 から MT9，そして MT11 から MT14 に適用され，図 7(b) のようになる．このクローニングの際，MT1 から MT4，MT6 から MT9，そして MT11 から MT14 は図 4の N のループをアンローリングしてできたものでありコード自体に違いがないため，クローニングによって新しくできた 3 つの SB(MT1', MT6', MT11') は同じサブルーチンを呼ぶようになり，コードサイズの増加が抑止できる．

5. 性能評価

本節では，4 節で本手法の説明に使用した ARC2D を使い，OSCAR シミュレータ上で本手法とアトムイメージ法の比較評価を行った．

ARC2D は解析手法としてオイラー法を用いた 2 次元流体解析プログラムであり，コード量は約 4500 行サブルーチン数 40 である．なお評価に際し，実行時間は ARC2D のメインループの 1 回転分の実行時間を表している．また，サブルーチン INTEGR の MTG に対する Para が約 3 であることから，評価の際の使用 PC 数は 3 が選ばれている．

図 8にそれぞれ PC 数を 1, 3 そして 3PC の時には 1PC 内 PE 数を 1, 2 と変化させた時の，アトムイメージ法のみを適用した場合と本手法を適用した場合のそれぞれの 1PE の時の実行時間に対する速度向上率のグラフを示す．

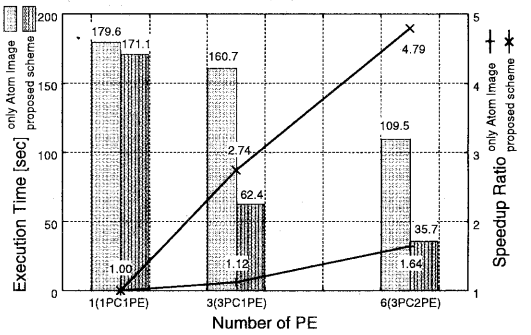


図 8　ARC2D によるアトムイメージ法のみ適用の場合と提案手法を適用した場合の比較評価結果

図 8から，インタープロシージャ解析にアトムイメージ法のみを適用した場合では，PE1 台の時の実行時間約 179.6 秒に対し PE を 6 台使用しても（構成は 3PC2PE）実行時間は約 109.5 秒までしか縮まらずに速度向上率が 1.64 倍にとどまっているのに対して，本手法を適用した場合では同じ 6 台の PE を使った場合（PC 構成は同じ）で 1PE の際の実行時間約 171.1 秒に対して約 35.7 秒と，4.79 倍の速度向上を得ている．また，アトムイメージ法のみによる解析の場合に比べ本手法を適用した場合の速度向上率の比は 6 台の PE を使用した場合で 3.07 倍である．

このような大きな性能差が出たのは，本手法の目的であるサブルーチンの境界を越えた粗粒度並列性の抽出が良く行われたからであり，これは，プログラム ARC2D 中で最も実行時間を費やすサブルーチン INTEGR 内 MT の実行状況を表す実行トレース図（図 9 と図 10）を比較することによって明らかになる．アトムイメージ法のみ適用した場合の実行状況を表す図 9 の実行トレース図と提案手法を適用した場合の実行状況を示す図 10の実行トレース図において，番号の振られている部分はその番号のマクロタスクがそのプロセッサクラスタで実行されていることを示し，逆に番号の振られていない灰色の部分はそのプロセッサクラスタ内の PE がアイドル状態であることを示している．図 9では，十分な粗粒度並列性が抽出できていないため灰色の部分つまりプロセッサがアイドル状態になっている時間が長いが，本手法を適用した図 10では粗粒度並列性のあるマクロタスクが効率よく各プロセッサクラスタに割り当てられていることがわかる．

また，フレキシブルインライニングにおいて，サブルーチンの境界を越えた粗粒度並列性抽出と並ぶも

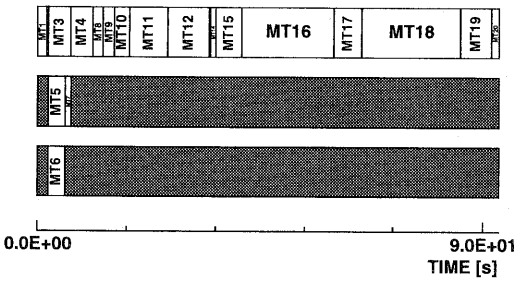


図 9　アトムイメージ法のみ適用した場合のサブルーチン INTEGR の実行トレース (3PC2PE)

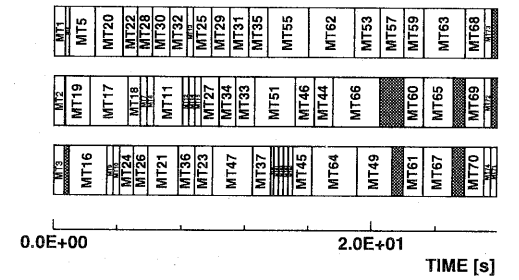


図 10　提案手法を適用した場合のサブルーチン INTEGR の実行トレース (3PC2PE)

う一つの大きな目標であるコードサイズ増加の抑制についてであるが，アトムイメージ法のみを適用した場合と本手法を適用した場合について，1PE 用のオブジェクトコードを出力させそのサイズを比較したところ，アトムイメージ法のみの場合が 158760byte に対し，本手法を適用した場合が 163580byte となり，約 3%の増加にとどまっていることが確認できた．

6. おわりに

本論文では，マルチグレイン自動並列化コンパイラにおける粗粒度タスク間の並列性抽出のためのインタープロシージャ解析手法として，従来のインタープロシージャ解析手法では解析できなかったサブルーチンコールに対して中間コードレベルでインライン展開を施した上で並列性の解析を行い，プロシージャ内外にわたるグローバルな並列性が存在すればインライン展開したコードをそのまま残し，グローバルな並列性が存在しない場合は並列性の存在しない部分をオリジナルと同一あるいは別形状のサブルーチンでフレキシブルにクローニングする，解析時インライニング及びフレキシブルクローニング手法を提案した．また，ベンチマークプログラム ARC2D による性能評価により従来のアトムイメージ法より約 3 倍のスピードアップを可能にすることが確認された．

本研究の一部は，通産省次世代情報処理基盤技術開発事業並列分散分野マルチプロセッサコンピューティング領域研究の一環として行なわれた．

参考文献

1) Banerjee, U.: *Loop Transformations for Restructuring Compilers*, Kluwer Academic Pub. (1993).
2) Wolfe, M.: *High Performance Compilers for Parallel Computing*, Addison-Wesley Publishing Company (1996).
3) 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, Vol. J73-D-I, No. 12, pp. 951–960 (1990).
4) Kasahara, H., Honda, H., Mogi, A., Ogura, A., Fujiwara, K. and Narita, S.: Multi-Grain Parallelizing Compilation Scheme for OSCAR, *Proceedings of 4th Workshop on Language and Compiler for Parallel Computing* (1991).
5) Li, Z. and Yew, P.-C.: Program Parallelization With Interprocedural Analysis, CSRD Technical Report 775, Center for Supercomputing Research & Development, University of Illinois (1988).
6) Li, Z. and Yew, P.-C.: Interprocedural Analysis For Parallel Computing, Csrd technical report, Center for Supercomputing Research & Development, University of Illinois (1988).
7) Havlak, P. and Kennedy, K.: An Implementation of Interprocedural Bounded Regular Section Analysis, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 3, pp. 350–360 (1991).
8) Hall, M., Amarasinghe, S., Murphy, B., Liao, S. and Lam, M.: Detecting Coarse-Grain Parallelism Using an Interprocedural Parallelizing Compiler, *Proceedings of Supercomputing '95* (1995).
9) Hind, M., Burke, M., Carini, P. and Midkiff, S.: An Empirical Study of Precise Interprocedural Array Analysis, *Scientific Programming*, Vol. 3, No. 3, pp. 255–271 (1994).
10) 笠原博徳: 並列処理技術, コロナ社 (1991).
11) 山本晃正, 稲石大祐, 宇治川泰史, 小幡元樹, 岡本雅巳, 笠原博徳: OSCAR マルチグレイン並列化コンパイラにおける階層的並列処理手法, 情報処理学会第 58 回全国大会 2D-04 (1999).