計算機アーキテクチャ　130－4
（１９９８．　８．　５）

マルチグレイン並列化コンパイラにおける臨界投機実行の効果について

山名早人　小池汎平　児玉祐悦　坂根広史　山口喜教
電子技術総合研究所　情報アーキテクチャ部

本報告では、マルチグレイン並列化コンパイラに、我々が提案している臨界投機実行を適用する場合の適用部分とその効果について検討する。マルチグレイン並列処理は、従来のループ並列化に加え、粗粒度並列処理、近細粒度並列処理を階層的に組み合わせて多様な並列性を抽出する並列化手法である。マルチグレイン並列化コンパイラに臨界投機実行を適用することにより、メモリアンビゲーションなどによりデータ依存関係が静的に解析できず並列化できない部分や制御依存によって並列化できない部分に対する並列化が可能となる。本稿では、プログラム中のどの部分に対して臨界投機実行を適用すべきかについて、ループを９つのカテゴリに分類すると共に、効果的適用法について検討する。

A Study of Adopting the Unlimited Speculative Execution on Multigrain Parallelizing Compilers

Hayato YAMANA　Hanpei KOIKE　Yuetsu KODAMA　Hirohumi SAKANE　Yoshinori YAMAGUCHI
yamana@etl.go.jp
Computer Science Division, Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305-8568 Japan

This paper discusses the effectiveness of the unlimited speculative execution and how to adopt the scheme on multigrain parallelizing compilers. The multigrain parallelizing compilers exploit parallelism among coarse-grain tasks like loops, medium-grain tasks such as loop iterations, and near-fine-grain tasks such as statements. When we adopt the unlimited speculative execution scheme on multigrain parallelizing compilers, the codes, that are not parallelized because of memory ambiguation or control dependences, are able to be parallelized. In this paper, loops are classified into nine categories to make clear the applicable loops for the scheme. Moreover, the effective adopting schme is discussed.

1．まえがき

本報告では、マルチプロセッサシステムの実効性能を向上させる手法として笠原らが提案しているマルチグレイン並列処理[1]に、我々が提案している臨界投機実行[2]を適用する場合の適用部分の選択法、及び効果の予測法について検討する。

マルチグレイン並列処理は、従来のループ並列化（中粒度並列処理）に加え、粗粒度並列処理、近細粒度並列処理を階層的に組み合わせて多様な並列性を多くのプロセッサ上で効率よく利用するための並列化手法である。しかし、プラットフォームとなる並列計算機が十分な数のプロセッサを持つ場合（十分な計算機資源がある場合）、DOALLなどのループ並列性が抽出可能な部分を除いて、プロセッサの利用効率が悪くなるのは避けられない。このような場合、中粒度あるいは粗粒度レベルでの投機的実行を行うことにより、余剰なプロセッサを使って、計算時間を短縮することができる。具体的には、マルチグレイン並列化コンパイラに臨界投機実行を適用することにより、制御依存やメモリアンビゲーション(Memory Ambiguation)によりデータ依存関係が静的に解析できない部分に対する並列化が可能となる。

以下では、マルチグレイン並列処理及び臨界投機実行について述べた後、プログラム中のどの部分に対して臨界投機実行を適用すべきかについて、ループを９つのカテゴリに分類すると共に、効果的適用法について検討する。

2．マルチグレイン並列処理[1]

マルチグレイン並列処理は、粗粒度並列処理[3]、中粒度並列処理、近最粒度並列処理の3つから構成される[1][4]。

粗粒度並列処理では、プログラムを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種類の粗粒度タスク(マクロタスク(MT))に分割する[5]。マクロタスク分割後、マクロタスク間のデータ依存関係と制御依存関係から各マクロタスクの最早実行可能条件[3]を求める。次に、マクロタスクを実行時に計算機資源に割り当てるためのダイナミックスケジューリングルーチンが生成され、複数のプロセッサ（PC:プロセッサクラスタ）にマクロタスクを単位として割り当て、実行される。マクロタスクは階層構成をとっており、中粒度タスク、近細粒度タスク、あるいは、サブマクロタスクに階層的に分割できる。

中粒度並列処理は、繰り返し文のイタレーションレベルでの並列処理であり、RBが割り当てられたプロセッサクラスタ内で並列処理される。

近細粒度並列処理は、ループ並列化ができないRBのボディ部、あるいは、ループ外部のBPAでのステートメントレベルでの並列処理であり、これらのブロックが割り当てられたプロセッサクラスタ内で並列処理される。

このように、マルチグレイン並列処理では、多様なレベルの並列性を利用することによって、プログラムから最大限の並列性を抽出し利用している。

3．投機的実行の分類

投機的実行(Speculative Execution)は、大きく分けて、Control SpeculationとData Speculationに分類される(図１)[6]。Control Speculationは、制御依存を無視して先行的に実行開始する投機的実行であり、Data Speculation は、データ依存を無視して先行的にデータの値やメモリ上の位置を予測し計算を進める投機的実行である。

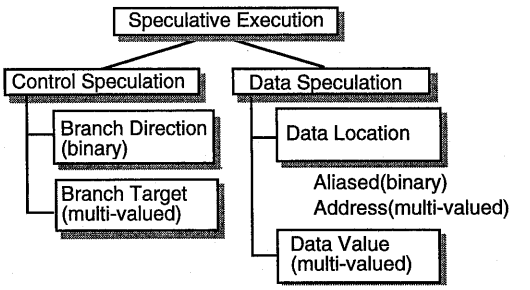


図１ 投機的実行の分類（文献6)より引用）

Control speculationは、さらに２つに分類できる。Branch Directionは、分岐先が２方向(例えば条件付ジャンプやDO WHILE型ループでの繰り返し条件など)である場合の投機的実行であり、Branch Targetは、最近の多くのマイクロプロセッサが内蔵するBranch Target Buffer(BTB)のように複数の分岐先を対象とした投機的実行である。

Data Speculation も、２つに分類できる。一つは、データアドレスを予想し先行的にプリフェッチするなど、Data Locationを対象とした投機的実行（さらに２値と多値で分類される）である。Data Location を対象とした投機的実行は、ポインタなどによりデータ依存の存在が実行時にしか決定されない場合(メモリアンビゲーションと呼ぶ)に用いられる投機的実行である。

Data Speculationのもう一方は、データを予測して先行的に実行を開始するData Value Speculation

である、例えば、データが外部ファイルや外部I/Oから渡される場合に、その値を予測して実行を開始する場合が該当する。あるいは、文献7)や文献8)が対象としているvalue predictorを使った投機的実行もData Value Speculationに分類される。

このように、Carnegie Mellon大学のM.H.Lipastiは投機的実行を分類しているが、Control SpeculationとData Speculationは表裏一体のものであり、Control Speculationを行った結果としてData Speculationが存在すると考えることができるため、Control SpeculationとData Speculationとを厳密に分類するのは難しい。

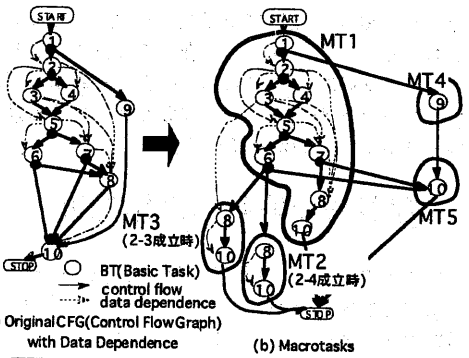
例えば、「条件分岐(制御依存関係)によってデータ依存関係が未定になる場合に、データ依存関係が存在しないことを前提にして投機的実行を行う場合」は、条件分岐の結果を仮定して投機的に実行を行うという意味から、Control Speculationであり、また、データのLocationを予測して投機的実行するという意味から、Data Location Speculationでもある。同様に、value predictor、すなわち個々の変数についてデータ予測を行う場合も、個々の変数に対するデータ依存関係と制御依存関係から、Control SpeculationやData Location Speculationと分類できる場合が存在する。

従って、以下では、ループ制御や条件分岐文などの制御依存に着目して投機的実行を行う場合をControl Speculation、メモリディスアンビゲーションやvalue predictionなど、データのLocationやvalueを予測して投機的実行を行う場合をData Speculationと分類することにする。

3. 臨界投機実行

臨界投機実行(Unlimited Speculative Execution) [2]は、中粒度(ループレベル)タスクレベルや粗粒度タスクレベルで投機的実行を行う並列化方式である。このように粒度の大きなレベルでの投機的実行により、プログラム全体に渡る投機的実行が可能となり、投機的実行の理想モデルであるOracle Model [9]適用時の理想的な速度向上率に近い速度向上を得ることができる。Oracle Modelとは、条件分岐の結果が実行前にすべて既知であり、計算機資源が無限であるという実行モデルである。

臨界投機実行は、❶投機的実行に適したマクロタスク生成法[2)10)]と❷マクロタスクの分散制御法[2)11)]の2つの方式から構成される。マクロタスクは、階層タスクグラフ(HTG)における同一階層のノード(基本タスクと呼ぶ)を単位として生成する。基本タスク間のデータ依存関係と制御依存関係から、投機的実行の効果のない部分、すなわち、データ依存と制御依存の両方を持つ基本タスク同士を融合する。また、データ依存関係が条件分岐により不定となる部分については、投機的実行の副作用を避けるため、



MT	臨界投機実行			マルチグレイン並列処理での最早実行開始条件
	実行開始条件	制御確定条件	実行停止条件	
1	True	True	False	True
2	True	2-4∧6-8	1-9∨2-3 ∨(2-4∧5-7) ∨(2-4∧5-10)	2-4∧6-8
3	3	2-3∧6-8	1-9∨2-4 ∨(2-3∧5-7) ∨(2-3∧6-10)	3∧6-8
4	True	1-9	1-2	1-9
5	True	1-9∨6-10 ∨7-10	6-8∨7-8	1-9∨6-10 ∨7-10

(c) マクロタスク制御条件

1はBT1の終了を、1-2はBT1でBT2側の分岐が選択されることを示す。∧はANDを∨はORを示す。

図2 投機的実行用マクロタスクの構成と制御条件

基本タスクの複製を行う。図2の(a)(b)にマクロタスクの構成例を示す。この例では、8と10の基本タスク(BT)が複製されている。例えばMT2とMT3は、2において、4の方向と3の方向に分岐した場合で3からのデータ依存関係が異なるため、複製されている。このような複製により、MT2は、他のマクロタスクからのデータ依存をもたなくなり、投機的実行が可能となる。このように、投機的実行の効果の小さい部分を一つのマクロブロックにまとめ、投機的実行の効果のある部分を分割点としてマクロタスクを生成する。

マクロタスクの分散制御法は、計算機資源に空きがある限り投機的にマクロタスクの実行を開始させる実行方式である。マクロタスクを集中的に管理せず分散管理することにより、制御オーバヘッドを改善している。図2(c)に、マルチグレイン並列処理で利用されている最早実行可能条件と、臨界投機実行での実行条件を示す。臨界投機実行では、図2(c)の実行開始条件が成立したマクロタスクから計算機資源に空きがある限り実行を開始させ、その後、実行停止条件が成立したマクロタスクについて、強制的に実行を停止させる。図2(c)からわかるように、投機的実行により、最早実行開始条件よりも早い段階

での実行が可能になる。

並列計算機EM-4(動作クロック12.5MHz)上でのこれまでの評価[2)10)11)]から、❶により生成されたマクロタスクの粒度は、最下位層で7～9μs、最下位層をさらに融合した階層では150～300μsとなることがわかっている。さらに、❷の実行方式により、マクロタスクの粒度が平均14.4μs以上あれば、制御のオーバヘッドがあっても、投機的実行による効果が得られることがわかっている。

4. 臨界投機実行の適用効果[12)]

マルチグレイン並列処理に臨界投機実行を適用する上で考慮すべき点は、マルチグレイン並列処理での臨界投機実行の適用部分の決定である。

ここでは、DOALLのように、RB内で大規模な並列性を持つマクロタスクを越えては、臨界投機的実行を適用しないと仮定し、科学技術計算で用いられる4本のサブルーチンを対象に臨界投機実行の効果を計測した結果を図3に示す。シミュレーションでは、すべての文を単項式に変換し、変換後の文単位の実行トレースから臨界投機実行を適用した場合と、適用しない場合(最早実行開始条件適用時)の各変数の定義参照時刻から、適用しない場合を基準とした速度向上比を計算した。なお、本シミュレーションは、マクロタスク化をする前の最下層の基本タスクを対象にシミュレーションを行った。その他の仮定としては、1)単項式の演算時間はすべて1単位時間、2)計算機資源は無限大、3)データ通信時間は0、4)DOALL型ループは並列実行、5)DOALL型ループを越える投機的実行は行わない、の5項目を設定した。

図3より、データ規模によって投機的実行の効果が異なることがわかる。LTとHBではデータ規模が増大するにつれ、投機的実行の効果が顕著にあらわれるのに対し、LXとBLでは投機的実行の効果が小さくなっている。この原因として考えられるのは、①LTとHBでは、投機的実行により逐次ループの各イタレーションを越えた並列性が得られた、②LXとBLでは、逐次ループ内に条件分岐が存在するものの、データ依存と制御依存の解決する時刻に差異がなく投機的実行の効果が得られなかった、③LXとBLでの投機的実行の効果はループ以外の部分によるものであるため、データ数増大によって投機的実行の効果が小さくなった、の3つである。

以上の結果から、臨界投機実行をマルチグレイン並列処理に適用する部分として最も適するのは、❶条件分岐を含む逐次ループで構成されるRBブロック内をサブマクロタスク化した階層で、かつ❷投機的実行によりイタレーション間で投機的実行の効果が得られる部分であると結論付けられる。この条件を満たす典型的なループはBoolean Recurrenceループ[13)]である。なお、②のような部分は、第3節で示した投機的実行に適したマクロタスク化により一つのマクロタスクとなるため、臨界投機実行では投機的実行の対象とならない。

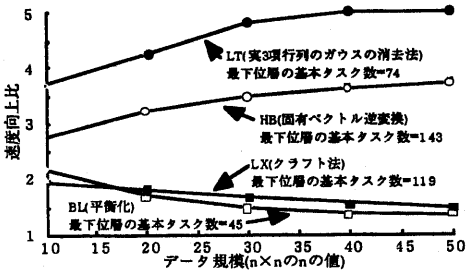


図3 データ規模による臨界投機実行の効果の違い

5. ループを対象とした投機的実行手法の分類

臨界投機実行の適用対象となるループを明確にするために、ループを分類すると共に、ループに関する主な投機的実行手法を分類整理する。

図4(a)に示すように、大きくループ制御部とループボディ部に関してループを分類する。ループ制御部とループボディ部についてそれぞれ3種類に分類することによって図4(b)に示すようにループを9つに分類することができる。

ループ制御	繰返回数固定	(L1)
	繰返回数不明 - ループボディに依存	(L2)
	繰返回数不明 - ループボディに非依存	(L3)
ループボディ	ループ伝搬依存無	(B1)
	ループ伝搬依存有 - 依存距離固定	(B2)
	ループ伝搬依存有 - 依存距離不明	(B3)

(a) ループの分類基準

	(L1)	(L2)	(L3)
(B1)	DOALL	投機的実行対象ループ	
(B2)	DOACROSS DOSERIAL		
(B3)		ループ例(1)	ループ例(2)

(b) ループの分類

(L1) イテレーション回数がそのループ実行直前までに決定しているループ
(L2) ループボディ内にLOOP EXITが存在したり、DO WHILE文の条件部でループボディで定義された値が参照されるループ
(L3) 繰り返し回数が外部入力や外部データによって決定されるループ
(B1) イテレーション間でデータ依存が存在しないループ
(B2) 依存距離が静的に解析可能なデータ依存が、イテレーション間に存在するループ。例えば、A(I)=A(I-3)+... など。
(B3) イテレーション間にデータ依存が存在するか否かが静的に解析できない、あるいは、ループ伝搬を伴うデータ依存の距離が不定であるループ。例えば、ポインタなどによるメモリアンビゲーションによるデータ依存を持つもの。

図4 ループの分類

```
while ( funct_units[i].class != ILLEGAL_CLASS ) {
    if (...) {
        if ( minclk > .... ) {
            minclk = .... ;
            ....
            if ( minclk == 0 ) break ;
        }
    }
    i++ ;
}
```

(a)ループ例(1)

124.m88ksimのsimtime.cのL.81-L.96を簡略記述(SPECint95)

```
while (( c = getchar() ) != EOF ) {
    ....
    .... = hash[hash_function(c)] ;
    ....
    hash[hash_function(...)] = .... ;
    ....
    if (free_entries < ...) { free_entries = ... }
    ....
}
```

(b)ループ例(2)

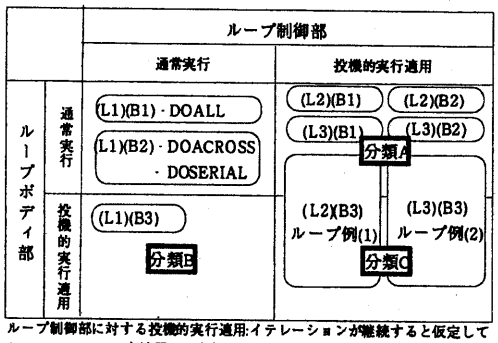
129.compressのL.475-L.546を簡略記述(SPECint95)

図5 ループ例

(L1)(B1)型のループは、イテレーション毎に並列化が可能なDOALL型ループであり、(L1)(B2)型のループはDOACROSS型(DOSERIAL型を含む)のループとなる。これら以外の7分類に分類されるループは制御依存やメデータ依存によりイテレーション間で並列実行ができないループである。

(L2)(B3)型と(L3)(B3)型のループ例を図5に示す。図5(a)では、break文により、ループ回数が不定となるため(L2)に分類される。また、minclkがループボディ内でループ伝搬依存する可能性があるが、依存距離は実行時にしかわからないため(B3)に分類される。一方、図5(b)は、繰り返し回数がループ以外の外部要因によって決定されるため、(L3)に分類される。また、hash[ ]及びfree_entriesについて、ループ伝搬依存する可能性があるが、依存距離は実行時にしかわからないため(B3)に分類される。

次にこれらのループに対する投機的実行を、ループ制御部、ループボディ部に対して投機的実行を適用するか否かで分類したものを図6に示す。例えば、J.G.SteffanらのTLDS(Thread-Level Data Speculation)[14]は、ループ制御部に対して投機的実行を行うと同時に、ループボディ部に対してもData Speculationを適用するので、図6の分類Cに分類される。なお、分類Aあるいは分類Bは、分類Cのサブセットであるので、TLDSは、分類A及び分類Cも対象とする。一方、Pen-Chung YewのSuperthreaded Processor[15]は、ループ制御部に対してのみControl Speculationを適用し、ループボディ部に対しては、投機的実行を適用しない。すなわち、ループボディ部におけるデータ依存関係は、実行時に動的に保証する実行形態を



ループ制御部に対する投機的実行適用:イテレーションが継続すると仮定してControl Speculationを適用して実行。
ループボディ部に対する投機的実行適用:イテレーション間のデータ依存関係が不定なものに対してData Location/Value Speculationを適用して実行。

図6 ループに対する投機的実行の分類

とる。このため、図6の分類Aに分類される。また、MUSCAT[16]は、TLDS同様、ループ制御部に対して投機的実行を行うと同時に、ループボディ部に対してもData Speculationを適用するので、分類C(分類A,Bを含む)に分類される。

一方、臨界投機実行の対象ループも、図6における分類C(分類A,Bを含む)に該当するループとなる。

6. 臨界投機実行の効果的適用法

図6の分類C(分類A,Bを含む)に該当するループにおいて、具体的にどのような部分に対して投機的実行を適用すれば、大きな効果が得られるかについて検討する。

仮定としてループ制御部に対しては、ループが継続する側に対してcontrol speculationを行うとする。このような仮定により、分類C(分類A,Bを含む)のループは、DOACROSS型ループ[17]の変形とみなすことができる。ループ運搬依存(loop carried dependence)をC1～Ckで表し、各々のデータ依存距離をΔで表すと図7のようになる。図7において、CiやΔが確定している場合が、分類A内の(L2)(B1)型,(L2)(B2)型,(L3)(B1)型,(L3)(B2)型のループであり、実行時にしかわからない場合が、分類B,分類Cの(L1)(B3)型,(L2)(B3)型,(L3)(B3)型ループに該当する。

CiやΔが確定している場合は、従来のDOACROSS型ループのスケジューリング手法[18]によりスケジューリングできると共に実行時間の予測も可能となる。一方、CiやΔが実行時にしか分からない場合には、Ci及びΔを実行トレースやvalue predictor[8]により予測することによって、Ci及びΔに対するData Speculationを行う。この時、臨界投機実行では、複数の予測対に対して同時にData Speculationを行うことができる。

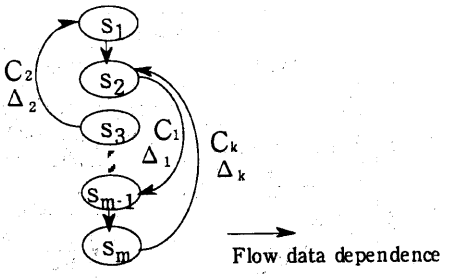


図7　文間のデータ依存関係モデル

投機的実行の効果を上げるためには、DOACROSS型ループのディレイd_0[17]（イテレーションの実行開始間隔)を小さくすることのできるData Speculation用の予測値(Data Value / Data Location)を用いた方がよい。すなわち、DOACROSS型ループのディレイd_0算出式[7]：

$$d_0 = \mathrm{MAX}_{i=1,k} | Ti / \Delta i |$$

Ti はi番目の LBD(Lexically-backward dependences) でもあるデータ依存のディスティネーション文とソース文間の実行時間

において、d_0を小さくでき、かつ、最終的にヒットする確率の高い予測値を用いたData Speculationを優先させるのがよいと考えられる。なお、具体的な優先順位決定法については、別途報告する。

7．まとめ

本報告では、マルチグレイン並列処理へ臨界投機実行を適用する場合の適用箇所及びその効果について予備的検討を行った。今後は、第5節で示した分類に従って、SPECint95等を対象にループの分類を行うと共に、第6節で示した効果的適用法に基づいて、投機的実行を適用すべき部分の抽出を行う。そして、実際にどのようなプログラム変換が可能であるかについて検討を行っていく。

謝辞

本研究を遂行するにあたりご指導、ご討論いただいた早稲田大学 笠原博徳氏、大蒔情報アーキテクチャ部長に感謝いたします。

参考文献

1) Kasahara,H., Honda,H.. and Narita, S.: A Multi-Grain Compilation Scheme for OSCAR, Proc. of 4th Workshop on Languages and Compilers for Parallel Computing, Springer-Verlag, Vol.589, pp.283-297 (1991)

2) Yamana,H.,Sato, M., Kodama, Y., Sakane, H., Sakai S., Yamaguchi, Y.: A Macrotask-level Unlimited Speculative Execution on Multiprocesors, Proc. of ICS95, pp.328-337(1995)

3) 本多,岩田, 笠原: Fortranプログラム粗粒度タスク間の並列性検出手法,信学論(D-I), Vol.J-73-D-I, No.12, pp.951-960 (1990)

4) 吉田,前田, 尾形, 笠原: Fortranマルチグレイン並列処理におけるデータローカライゼイション手法. 情報処理学会論文誌, Vol.36, No.7, pp.1551-1559 (1995)

5) 笠原, 合田, 吉田, 岡本, 本多: Fortranマクロデータフロー処理のマクロタスク生成手法, 信学論 (D-I), Vol.J75-D-I, No.8, pp.511-525(1992).

6) M.H.Lipasti,J.P.Shen: "Exceeding the Dataflow Limit via Value Prediction", IEEE Micro, Vol.29, pp.226-237 (1996.12).

7) Y.Sazeides, J.E.Smith: "The Predictability of Data Values", IEEE Micro, Vol.30, pp.248-258 (1997.12).

8) 小池,山名,山口: "投機的制御/データ依存グラフとJava Jog-time Analyzer - Java Virtual Accelerator 実現へ向けての予備評価 -", 情処研報, PRO-6, SWoPP98 (1998.8).

9) Nicolau,A., Fisher,J.:Measuring the Paralllelism available for Very Long Instruction Word Architecture, IEEE Trans. Comput., Vol.33, No.11, pp.968-976 (1984)

10) 山名,安江,石井,村岡: 並列処理システムにおけるマクロタスク間先行評価方式,信学論(D-I),Vol.J77-D-I,No.5,pp.343-353 (1994)

11) 山名, 佐藤,児玉,坂根, 坂井, 山口:並列計算機EM-4におけるマクロタスク間投機的実行の分散制御方式,情報処理学会論文誌,Vol.36, No.7,pp.1578-1588 (1995)

12) 山名,小池,児玉,坂根,山口:マルチグレイン並列化処理における臨界投機実行の適用, 情処第56回全大,2E-3 (1998.3)

13) Banerjee U.,Gajski,D.D: Fast Execution of Loop with IF statements, IEEE Trans. Comput., Vol.C-33, No.11, pp.1030-1033 (1984)

14) J.G.Steffan, T.C.Mowry: The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization, Proc. of HPCA-4, pp.2-13 (1998.2).

15) J.Y.Tsai, ZJiang, E.Ness, P.C.Yew: Performance Study of a Concurrent Multithread Processor, Proc. of HPCA-4, pp.24-35 (1998.2).

16) 酒井,鳥居, 近藤, 市川,小俣,西,枝廣: 制御並列アーキテクチャ向け自動並列化コンパイル手法, JSPP98, pp.383-390 (1998.6)

17) R.Cytron : Doacross : Beyond Vectrization for Multiprocessors, Proc. of Int. Conf. on Parallel Processing 86, pp.836-844 (1986).

18) 山名,安江,村岡,山口: 分散共有メモリ型並列計算機における1重Doacross型ループの実行時間算出法,信学論 D-I, Vol.J78-D-I,No.2,pp.170-178 (1995.2)