計算機アーキテクチャ　139－28
（２０００．８．５）

臨界投機実行のループへの適用

山名早人　　小池帆平
早稲田大学 理工学部 情報学科　電子技術総合研究所 情報アーキテクチャ部

本報告では、我々が提案している臨界投機実行をループに対してどのように適用させるかについて検討する。臨界投機実行は、タスクレベルでの投機的実行方式であり、これを、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ループや、制御依存によって並列化できないループに適用することによって、高速化を図る。ループを9つのカテゴリに分類すると共に、SPEC95int の compress に対して適用を検討した結果を示す。

Unlimited Speculative Execution for Loops

Hayato YAMANA
yamana@acm.org
Dep. of Information and Computer Science,
School of Science and Engineering
Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555 JAPAN

Hanpei KOIKE
hkoike@etl.go.jp
Computer Science Div.
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba,
305-8568 JAPAN

This paper discusses how to adopt the "Unlimited Speculative Execution" on loops. A task level speculative execution scheme, called the "Unlimited Speculative Execution", is adopted on the loops that are not able to be parallelized because of memory ambiguation or control dependences. In this paper, loops are classified into nine categories to make clear the applicable loops for the scheme. Moreover, we discusses the result after applying the scheme to SPEC95int compress program.

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) における中粒度タスクを対象に、我々が提案している臨界投機実行 2) を適用する手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、中粒度のループに対してどのように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。

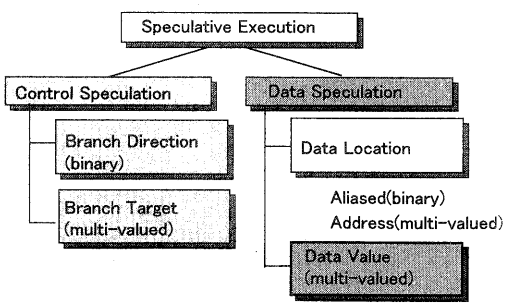

図1 投機的実行の分類(文献 6)より引用)

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。

4. 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、①投機的実行に適したマクロタスク生成法 2) 10) と②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、基本タスクの複製を行い、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。

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、ループを分類し、ループに関する主な投機的実行手法を分類整理する(図2)。図に示すように、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2に示すようにループを9つに分類することができる。

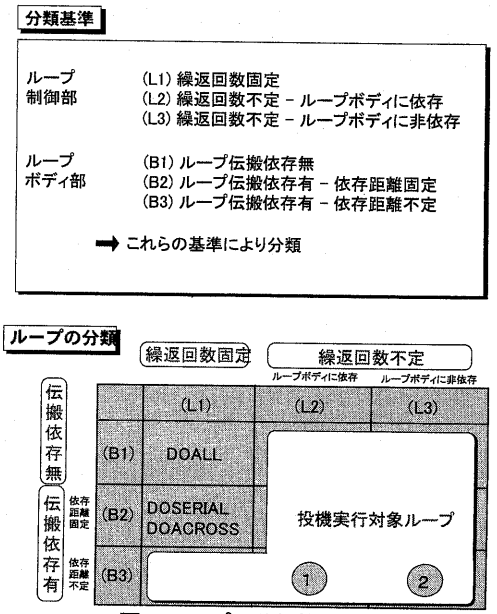

図2 ループの分類基準と分類

(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3に示す。ループ例(1)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、ループ例(2)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4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4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とる。このため、図4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4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。

ループ例(1)　(L2) 繰返回数不定 - ループボディに依存
(B3) ループ伝搬依存有 - 依存距離不定

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

SPEC95int 124.m88ksim (simtime.c)

ループ例(2)　(L3) 繰返回数不定 - ループボディに非依存
(B3) ループ伝搬依存有 - 依存距離不定

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

SPEC95int 129.compress - compress()

図3 ループ例

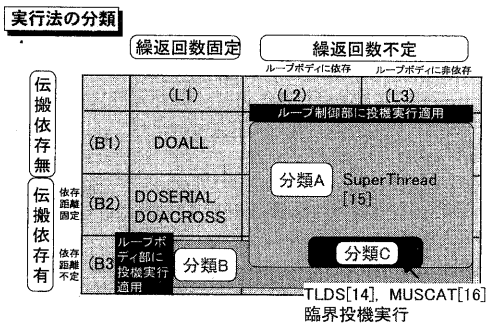

図4 ループに対する投機的実行の分類

5. 臨界投機実行の効果的適用法

図4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(図5)。

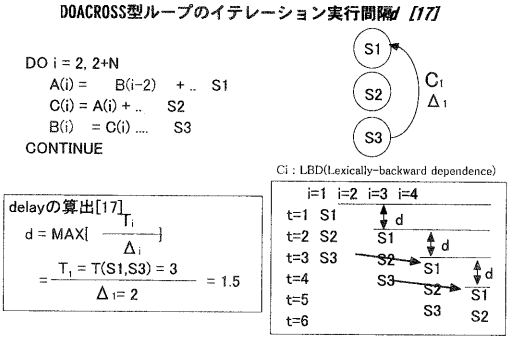

図5 DOACROSS ループ

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5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。

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 d0 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 d0 算出式 7) :

$$d0 = \mathrm{MAX}_{i=1,k}\{ Ti / \Delta i \}$$

Ti は i 番目の LBD(Lexically-backward dependences)で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、d0 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を優先させることにより効果的に投機的実行が適用できる。

以上をまとめると、投機的実行の適用方針は次のようになる。

① ループ制御部に対してループ継続側に Control Speculationを適用する。

② ループを DOACROSS 型ループと考え、ディレイd0を算出する。データ依存をCi∈LBDとし、データ依存 Ci のディスティネーション文とソース文間の実行時間をTi、データ依存 Ci が存在する時のイテレーション内の制御フローが選択される確率を Pi とすると、d0=MAX(Ti×Pi÷Δi)により求める。但しΔi は、データ依存距離である。

③ ディレイd0の決定要因となるデータ依存Ci∈LBDについて、Data Speculation が可能かどうかを実行時のヒストリ等から調べ、予測可能であればData Speculationを適用できると判断し、その予測成功率Qiをヒストリから求める。

④ ③において、Ci∈LBDにData Speculationを適用する度にData Speculationを適用されていないデータ依存 Ci∈LBD により d0 を再計算する。この時、d0 を決定しているデータ依存で、Data Speculationの適用可能なCiが存在しなくなるか、LBD が存在しなくなるまで③～④を繰り返す。

⑤ ③でData Speculationを適用すると判断された{Ci}の各々のデータ依存に対して、イテレーション内での制御フローFi を求める。さらに、{Ci}に全く関係ないイテレーション内の制御フロー群を一つにまとめた制御フローを求め、その制御フローを含めて{Fi}とする。

⑥ 制御フローFi の平均実行時間をヒストリ等から求め、Liとする。

⑦ 投機的実行を適用しない場合を基準とした、投機的実行を適用した場合の速度向上率は、

$$\frac{\Sigma(Li \times Pi)}{\Sigma(Li \times Pi \times (1-Qi))}$$

と表すことができる。

以上の手法をSPECint95 の 129.compressに適用してみる。129.compress では、実行時間の 99%を関数compress が占めるため、以下では、関数 compress のみを考える。

図6にその結果を示す。図6の例では、C1 についてData Speculation が可能であるが、C2, C3 についてはData Speculation が適用不可能(データ値が予測不可能)と判断された。このため、d0 は C2、C3 で決定され、C4 以降について Data Speculation を適用し100%成功したとしても、効果が得られない。従って、Data Speculation を C1 のみについて適用する。

本例の場合、F1 は C1 が存在するループ内の制御フロー、F2 は C1 が存在しないループ内の制御フローとなり、入力として、input:refを用いた場合、Q1 は30%、Q2 は70%、さらに、P1 は100%となった。

これらの値から、129.compress に対して関数compress にイテレーション間で投機的実行を適用した場合、約 2.18 倍高速化が可能であることが分かった。ただし、この値は、関数 compress の1イテレーション内での並列化や投機的実行を考えていないため、さらに高速化が期待できるものと考えられる。

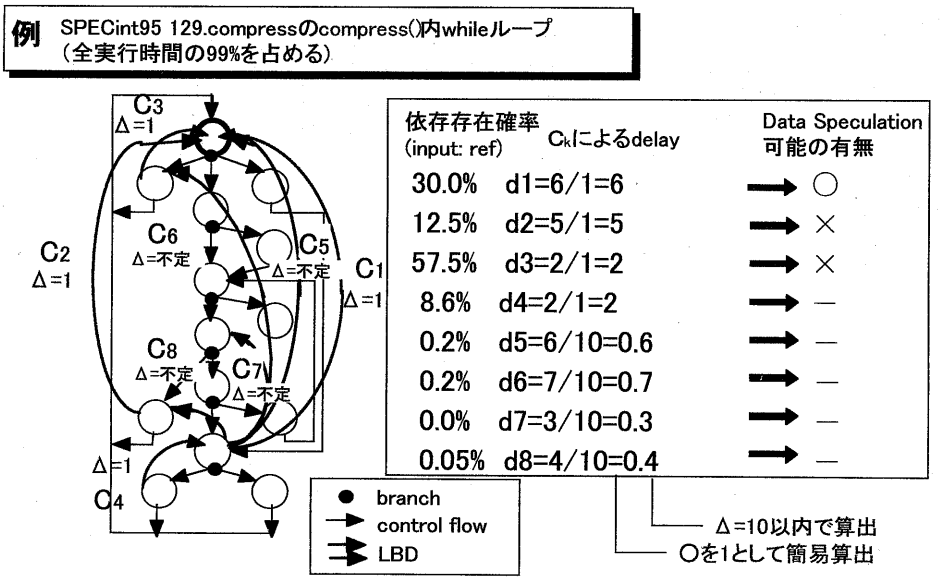

図6　129.compress に対する投機的実行の適用

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節で示した分類をさらに、詳細化すると共に、第6節で示した例について、実際にシミュレーションにより、その効果を確認する予定である。さらに、SPECint95 の他のループについても、分類及び効果の予想を行っていく。

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、電子技術総合研究所大蒔情報アーキテクチャ部長、情報アーキテクチャ部予測投機アーキテクチャラボ諸氏に感謝いたします。

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, Z.Jiang, 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)