計算機アーキテクチャ　107－14
（１９９４．　７．　２１）

投機的実行の現状と Unlimited Speculative Execution Schemeの提案

山名早人　佐藤三久　児玉祐悦　坂根広史　坂井修一†　山口喜教
電子技術総合研究所　　†新情報処理開発機構

本報告では，条件分岐の評価を待たずに実行を開始する投機的実行の現状をサーベイすると共に，並列計算機上で条件分岐を無制限に越えた実行を行い高速化を図る方式として，Unlimited Speculative Execution Schemeを提案する．従来の投機的実行に関する研究は，３つに分類される．(1)命令レベルの並列性，(2)Superscalar／VLIWを対象とした命令多重度充足のための投機的実行，(3)並列計算機上での投機的実行，に関する研究である．本報告では，この分類に基づいて従来の研究をまとめる．そして，12～630倍の速度向上が得られる投機的実行の理想モデル - Oracle Modelを並列計算機上で実現するための実行方式を提案する．

Survey of Today's Speculative Execution Schemes and a Proposal of Unlimited Speculative Execution Scheme

Hayato YAMANA　Mitsuhisa SATO　Yuetsu KODAMA　Hirohumi SAKANE
Shuichi SAKAI †　Yoshinori YAMAGUCHI
yamana@etl.go.jp
Electrotechnical Laboratory　　† Real World Computing Partnership
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan

Firstly, today's speculative execution schemes are surveyed. The schemes are categorized by three ; (1) researches for instruction level parallelism, (2) speculative execution schemes to fill the pipeline stages of VLIW/Superscalar processors, and (3) sepeulative execution schemes on parallel machines. Secondly, a new speculative execution scheme, called *unlimited speculative execution scheme*, is proposed on parallel machines to archive high performance which is extracted on *Oracle Model* such as 12～630 times faster than without speculation.

1．まえがき

条件分岐の評価結果を待たずに，演算を開始することを投機的実行(Speculative Execution)と呼ぶ．本報告では，投機的実行に関する最近の研究をサーベイすると共に，並列計算機上で投機的実行を行う新たな方式を提案する．

プログラムを並列化する際の問題点に，制御依存[FeOW87]に基づく実行順序関係（先行制約）のために十分な並列性をプログラムから引き出すことができないという点がある．例えば，以下の例では，S2の実行がS1の条件分岐で決定されるので，S1において制御が確定するまでS2を実行できない．

```
if (A) then                S1
    B=C×D                  S2
endif
```

この時，S2はS1に制御依存していると言う．すなわち，制御依存とは，条件分岐とその条件分岐によって実行されるかどうかが決まる文間の依存関係を表す．これに対して，S2とS1の間の制御依存に基づく先行制約を無視してS2の実行を開始し，後にS1の制御が確定した時点でS2の結果を有効とするか無効とするかを決定する方法を投機的実行と呼ぶ．条件分岐の結果が実行前に全て既知であるという仮定のもとでプログラムを実行すると，投機的実行を行わない場合に比較して12～630倍の速度向上が得られる[RiFo72] [NiFi84][LaWi92].

従来の投機的実行に関する研究は，大きく以下の3つに分類できる．

❶命令レベルの並列性に関する研究

❷Superscalar／VLIWを対象とした命令多重度充足のための投機的実行の研究

❸並列計算機上での投機的実行に関する研究

❶では，プログラムに内在する並列性がどの程度あり，どのような手法により並列性が抽出できるかをプログラムの実行トレースから解析する．投機的実行の理想モデルとして，全ての条件分岐の結果が実行前に既知であり，計算機資源が無限であることを仮定としたOracle Model[NiFi84]を用い，逐次実行からOracle Modelまでの間の各種モデルを想定することにより，命令レベルの並列度を解析している．

❷は，VLIW/Superscalar計算機が持つ命令多重度を満たすことを目的とする．VLIW/Superscalar計算機は，2つの命令レベル並列処理機構を持つ．時間並列処理(パイプライン処理)と空間並列処理(パイプラインの多重化)である．例えばパイプライン段数が4段，命令パイプラインが4本ある場合，16命令が同一時刻にパイプライン中に存在する．これら16個の命令中にデータ依存及び制御依存の依存関係が無ければ4本のパイプラインの全ステージを命令で埋めることができ，最大性能が出せる．しかし，実際には，これらの命令間にはデータ依存及び制御依存が存在し，全てのステージを命令で埋めることができない．そこで，パイプラインステージ中の命令の空きスロットを無くすために，投機的実行が行われる．

❸では，多数のプロセッサを用いて条件分岐を複数段に渡って投機的実行することによる速度向上を狙う．

以下では，投機的実行に関する従来の研究を上記で述べた3つに分類しサーベイを行う．そして，5節において，並列計算機上でOracle Modelを実現するための方式として新たに Unlimited Speculative Execution Schemeを提案する．

2．命令レベルの並列性と投機的実行

プログラムに内在する命令レベルの並列性に関して，従来の研究結果を元に，命令レベルの並列性を増加させる上での重要点を明らかにする．なお，「x段の条件ジャンプ」とは，常にx個の条件分岐を超えて投機的実行が行われることを示す．

投機的実行を行わずに，プログラム変換による命令レベルの並列性の向上は[KuMC72]に示される．Kuck,村岡,ChenらはFortranで記述された20本の数値計算系のプログラムを用い，**tree hight reduction**と**wavefront computation**を用いたプログラム変換により16台PEを使用時のPE稼働率が0.4-0.6となり，命令レベルの並列性抽出が可能であることを示した．

一方，同時期に，RisemanとFosterは，条件分岐の結果が実行前に全て既知であり，計算機資源が無限であることを仮定とした場合の速度向上率を求めている[RiFo72]．このような実行モデルを**Oracle Model**と呼ぶ[NiFi84]．Risemanらは，CDC3600用の7本の数値計算系及び非数値計算系のプログラムを用い，逐次に実行する場合に比較し平均51倍の速度向上が得られることを示した．0段の条件ジャンプ時，すなわち，計算機資源を無限とし，投機的実行を行わないとした場合の速度向上は1.72倍である．従って，無限段の条件ジャンプによって約30倍の速度向上が得られたことになる．一方，逐次に実行する場合に比較して10倍の速度向上を得るためには約16段の条件ジャンプが必要であり，その時の枝(Path)数は2^{16}になり，投機的実行は効果的な速度向上の手法としては適さないとしている．

Wallは，**branch prediction**，**resister renaming**，**alias analisys**，**window size**が命令レベル並列性に与える影響を調べ，特にbranch prediction(分岐予測)が並列度向上に最も重要である[Wall91]としている．しかし，[Wall91]では，シミュレータの制限上，1サイクル内の抽出可能最大並列度を64としており，投機的実行による性能向上の上限は不明である．

投機的実行の性能向上をよく表しているものに，[LaWi92]がある．[LaWi92]では，Oracle Model用いることにより，投機的実行を行わない場合を基準にして([LaWi92]のCD-MFモデルを1に換算)，3本のSPEC INTを含む非数値計算系の7本のプログラムで12倍(gcc)～630倍(eqntott)，Spice2g6で50倍の性能向上が得られることを示している．また，300×300の

matrix計算では，扱うデータ量に応じて十分な並列性が得られており，投機的実行を行っても，2.7倍の性能向上に留まるとしている．以上から，非数値計算系のプログラム，及び，数値計算系ではSpiceが投機的実行に向くことがわかる．

Theobaldらは，[ThGH92]で**smoothability**を定義し，投機的実行の効果が，どの程度，PE台数に比例するかを示している．つまり，Smoothabilityが100%であれば，PE台数に比例して，条件分岐のジャンプ段数が増加し，速度向上が得られる．数値計算系及び非数値計算系の9本のプログラムを評価した結果，いずれも50%以上であり，有限の計算機資源上で投機的実行の効果を十分に期待できるとしている．

また，山本らは，SPEC Benchmarkの一部(gcc, doduc,fppp,sipce2g6)を用い，並列性を検出するコード範囲，すなわち，**window size**が16の場合，16命令中の45%がすでに分岐確定しており，かつ，2サイクル以内に残り40%も分岐確定することを示している[YYUY92]．すなわち，window sizeが16の時，分岐確定以前に投機的実行を行うことが困難であることを示している．

また，[ThGH92]でも，window sizeを変えた場合の報告がされており，window sizeが64の場合，Oracle Modelの0.1～14%，2Kの場合でも0.3～52%の性能しか得られないことが示されている．さらに，扱うデータサイズが増大するとループ長が長くなるため，ループ間に渡る最適化が出来ず(window sizeが一定である場合)，Oracle Modelに対する性能が低下することを示している．

以上の文献から，以下のことがわかる．

❶非数値計算系(数値計算系のspiceを含む)のプログラムでは，投機的実行を行わない場合の平均命令間並列度は，1.5～2程度である[RiFo72][Wall91][ThGH92][KiHI93]．数値計算系では扱うデータサイズに依存する[KuMC72][ThGH92].

❷Oracle Modelを仮定すると，12～630倍の速度向上が可能である[RiFo72][NiFi84][LaWi92].

❸並列性を検出するWindow Sizeは十分に大きく(2K以上)なくてはならず[Wall91]，ループ間を跨った検出も重要である[ThGH92].

3．VLIW/Superscalar上での投機的実行

3.1 概要

VLIW/Superscalarを対象とした投機的実行の研究は，VLIW/Superscalar計算機の命令多重度を満たすことを目的とする．すなわち，パイプラインハザードによるパイプラインステージ中の空きスロットを無くすために，投機的実行が行われる．

パイプラインハザードとは，先行命令に対して後続の命令が依存関係(データ依存あるいは制御依存)を持つ場合，及び，必要とされるリソースが無いために，命令を連続して実行できないことを言う．特に，条件分岐命令による制御依存の場合，条件決定後に命令をフェッチするとbranch penaltyが生じ(図1でBr.がEXステージ実行後，T命令をフェッチ)に示すように，空きスロットが出来る．この時，(1)条件分岐後の片側のパス上の命令を予めプリフェッチし(分岐予測(**branch prediction**)と呼ぶ)，プリフェッチが失敗した場合(図1のi～i+1命令側が選択されなかった場合)にのみ，パイプラインをflash(無効化)し，新たな命令(図中のT命令)をフェッチしたり，(2)コンパイラによりi～i+1の部分に分岐に関係なく実行される命令を置くことにより(遅延分岐(**delayed branch**)[Lilj88][Carg91]と呼ぶ)，branch penaltyを無くす方法がとられる．なお，i～i+1の部分を**delay slot**と呼ぶ．しかし，(2)の方法により，delay slotを埋めるのは容易ではなく，delay slotが1つの場合は，その70%を埋めることができるが，delay slotが2つの場合には，2番目のdelay slotに対して25%しか埋めることができない[McHe86].

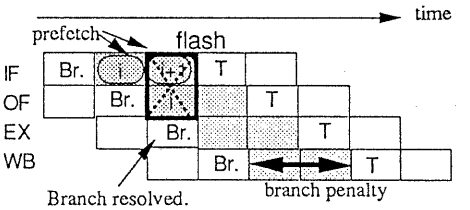

図1 パイプラインでのbranch penalty

以下では，空きスロットを解消するために行われる投機的実行を❶投機的実行の対象分岐先(片側／両側)，❷例外処理，❸コンパイラ支援，❹アーキテクチャ上の支援，❺言語，に着目して分類する．

3.2 分岐予測

分岐予測は，条件分岐後の片側のパスが選択されることを予測し，delay slotを埋める手法であり，動的分岐予測(**dynamic branch prediction**)と静的分岐予測(**static branch prediction**)に分類される．

動的分岐予測では，各条件分岐に対して1～2bitの**branch history table**[LeSm84][Carg91]をハードウェアで設け，過去の分岐履歴に基づいて次の分岐先を予測する．これに対して，静的分岐予測は，過去のトレース，あるいは，プログラムの特徴に基づいて分岐方向をコンパイル時に静的に決定する．

動的分岐予測は，静的分岐予測が不得意とするプログラムに対してもよい結果を出している．例えば，Spiceに対する静的分岐予測のヒット率は50～70%[FiFr92]であるのに対し，動的分岐予測による方法では，94%以上[PaSR92]である．

静的分岐予測による手法の内，プログラムの特徴に基づいて分岐先を決する方式は，(1)分岐命令の種類によって*Taken*(分岐)と*Not Taken*(非分岐)の確率が異なることに着目し，分岐命令の種類から予測(例えばbltzは*Not Taken*と予測)する手法[LeSm84]や，(2)分岐命令の使われ方を前後の文から判断し(例えばloopに使われているかを判断)ヒューリスティックに分岐

方向を決定する手法が提案されている[BaLa93].

分岐予測のヒット率は，動的分岐予測の場合，92～96%[LeSm84][PaSR92][YePa93]，静的分岐予測の場合，74～76%[FiFr92][BaLa93]である．しかし，動的分岐予測を行うためには2K～8Kbitのテーブルと付加ハードウェアが必要である．

3.3 ブースティング

分岐予測では，分岐命令を超えて命令を前方に移動できないのに対し，ブースティング(boosting)[SmLH90]は，投機的実行を示すフラグを命令に付加(コンパイル時)し，分岐命令を超えて前方に命令を移動する．これにより，3.1で述べた(1)(2)による空きスロットを解消する．フラグ付命令の実行結果は**shadow register**に書かれ，分岐が成立した段階でshadow resisterが通常レジスタとなり他の通常の命令からアクセス可能となる．分岐不成立の場合には，無効化される．図2では，i1,i2がブースティングされている．*.BR*は右方向への分岐，*.BRR*は右方向への2回の分岐を示す．i1での結果はshadow resisterに書かれ，i2で(分岐方向が同一であるので)shadow resisterを参照している．

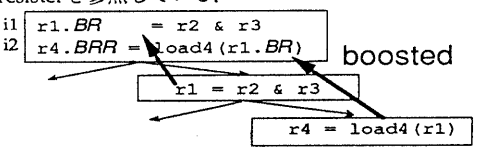

図2 ブースティング

ブースティングのレベル数(投機的実行のジャンプ段数)に応じてshadow resisterが必要となるので，[SmLH90]では1段の片側のみ，[SmHL92]では多段を許すが**likely pass**(選択確率の高いパス)のみを投機的実行の対象としている．ブースティングによる速度向上は，1.6倍程度である[SmLH90][SmHL92].

コード移動範囲を制限すると，ブーストされた結果がレジスタに書かれる前に無効化できる点に着目し，shadow resisterを不要化すると共に，分岐の両方向からのブースティングを行う方法[ANMH93a][ANMH93b][HANM93]も提案されている．また，[ChLS92]では，shadow resisterと通常レジスタを一体化しレジスタの効率的な利用方法を提案している．

3.4 例外処理

投機的実行を行う場合，投機的実行された命令において例外が発生する場合がある．ブースティングでは，投機的実行命令にフラグを付加すると，ハードウェアにより，投機的実行中の例外発生が抑制され，分岐確定時点で例外処理を行う[SmLH90].

一方，コンパイラレベルでの例外処理方法として**sential scheduling**[MCBH93]や**delayed exception**[ErKr94]があり，何れも，ユーザが例外処理を記述できるという特徴をもつ．これらの方式では，ただちに例外処理を行う命令と例外処理を抑制する命令の2種類をコンパイル段階で使い分け，例外処理を抑制する命令実行中に例外が生じた場合には，各レジスタに用意された**exception bit**を立てる．そして，プログラム中に配置した例外チェック命令によりexception bitを調べ例外処理を行う．

また，データ駆動計算機上で投機的実行を行う場合の例外処理方式としてエラートークン[HaYM91]が提案されている．

3.5 コンパイラ支援

投機的実行におけるコンパイラ支援は，コード最適化の対象範囲決定が重要なポイントである．

トレーススケジューリング(**trace scheduling**)[Fish81]は，最も実行確率の高いトレース(パス)を抽出しコード最適化を行う方法である．しかし，他のパスとのつじつま合わせのための**book keeping**が複雑[SSSH90]になるという問題を持つ．

複雑なbook keepingによる性能低下を無くす方法として，**superblock**[HMCC93]がある．superblockは(1)トレースの決定，(2)後部複製(**tail duprication**)により生成される．(1)ではbasic block単位で実行確率を収集し実行確率の高いトレースを抽出する．(2)はトレース中への制御の飛び込みを無くすためのコード複製であり，これにより，book keepingが簡単になる．図3の例では，BB6が複製されている．そして，生成されたsuperblock内でコード移動を行う．

superblockは1つのトレースを対象とするのに対し，実行確率の高い**複数の**トレースを1ブロックとする方法に**hyperblock**[MLCH92]がある(図3)．hyperblock-のコードスケジューリングは，**If-Conversion**(IC)を用いる．すなわち，*add r5,r5,2,if p1-F* のように*p1-F*という条件付き命令に変換する．そして，この条件をキーにしてスケジューリングする．この際，コードが分岐前に移動された場合，投機的実行となる．なお，対象プロセッサは，Cydra5[RYYT89]やTRACE[CNOP88]のように**predicated execution**(条件付実行)を備えるプロセッサである．

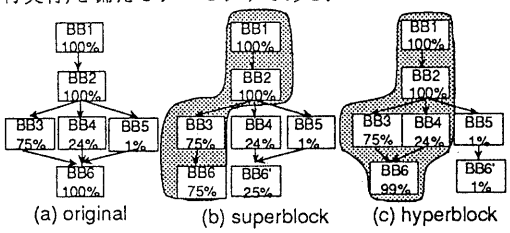

図3 Superblock / Hyperblock

さらに，[WMHR93]では，hyperblockとICを用いたスケジューリングを条件付実行の機能を持たないプロセッサでも使用できるように，条件付き命令を再び元のVLIW用コードに戻す変換手法，**Reverce If-Conversion**(RIC)を提案している．hyperblockとICを用いたスケジューリングにより17-19%の性能向上(modulo scheduling比)が得られる[WMHR93].

また，**percolation**(パーコレーション)[Nico85]や**modulo scheduling**(モジュロスケジューリング)[Char81]と投機的実行を組み合わせた評価も行われている．投機的実行を合わせて行うことにより，percolation比で20-30%[CMCW91]，exitを含むルー

プを対象としたmodulo scheduling比で3%～533%[TiLS90]の性能向上が得られることが報告されている．特に，exitを持つループでは，投機的実行が非常に有効であることがわかる．

3.6 アーキテクチャ上の支援

以下では，投機的実行を支援するハードウェアについて，(1)分岐命令の高速化，及び，(2)Superscalar独自の支援ハードウェアでついて述べる．

【分岐命令の高速化】非数値計算では，分岐命令の出現頻度が高く，分岐命令を超える命令移動を行っても十分な並列化が得られない．この点に着目し，VLIW計算機を対象に**multiway jump**が提案されている．[APBN91]では，2つのコンディションコードレジスタを調べ分岐を行う **BRTT A,B,offset** 命令を提案している．この場合コンディションコードレジスタA,Bが両方ともTRUEの時offsetに分岐する．複数のBRXX命令を並列実行し，multiway jumpを実現する．[MoEb92][MoEb93]では，target maskビット([MoEb93]では3bit)を設け，target maskビットが一致する命令を選択しmultiway branchを行う．条件判断と分岐結果に基づいた命令選択は同一サイクル中で行う．一方，[KKSF93]では，target maskビットに*don't care*を設け，複数のmultiway jumpを同時実行可能にした．また，[AMNH94]では，レジスタファイル上にブースティングで用いられるshadow resisterの他に分岐条件を用意し，複数の制御フローに対する投機的実行をサポートする．

【Superscalar Processor上での投機的実行支援】SuperScalar上での投機的実行支援には，(1)命令コードスケジューリングと，(2)分岐の保証処理がある．(1)は，in-order/out-of-orderに分類[Mura91]できる．in-orderでは，コンパイラが生成した命令順で命令発行し，特別な機構が必要ないので，out-of-orderでの命令発行機構のみを考える．out-of-orderで命令発行するには，実行時の動的なデータ依存関係検出が不可欠である．タグを用いて局所データフローを実現する**reservation station**[Toms67]，reservation stationを強化し命令の追い越しをフェッチwindow間で許した**dispatch stack**[AcKT86]，**fast dispatch stack**[DwTo92]，タグの連想マッチングを排除した**reservation station tag unit**[Shhi90]，複数のデータフローを認識し最尤フロー依存解消時に発行する**multiple dependency representation scheme**[MIKT89]，等が提案されている．一方，(2)分岐の保証処理はout-of-orderでの命令終了よって発生する不正確なマシン状態を解消する方法である．実行された結果(out-of-order)をスプールし，in-orderに並べ換えレジスタファイルに書き戻す**reorder buffer**[SmPl88]，古いデータを保持し例外発生時に復元する**history buffer**，**future file** [SmPl88]，**checkpoint repair**[HwPa87]，結果を一時的にスプールし制御確定時にレジスタファイルに書き戻す**resister update unit**[Shhi90]，実行命令列と結果をテーブルに持ち，制御確定時にレジスタファイルに結果を書き戻し，分岐予測失敗時には実行命令列へのポインタ変更処理により高速な復帰を行う**schedule table**[PiMe93]，分岐予測失敗時，パイプラインをflashせず，選択的な命令無効化を行う**selective instruction squashing**[MIKT89] [KIHM89]等が提案されている．

【その他】　投機的実行を命令レベルではなく基本ブロック単位で投機的実行するプロセッサが提案されている[MNON91][INOM93][AYKH94]．これらのプロセッサでは，基本ブロック単位で投機的実行を行い，基本ブロック単位で結果をスプールする．すなわち，shadow resisterを大きくした機構を持つ．

3.7 言語

投機的実行をプログラム上で明示する方法も考えられている．[JuHH93]では，***in future*** 文による投機的実行を提案しているが，構想の段階である．

4．並列計算機上での投機的実行

並列計算機上での投機的実行の手法を最初に提案したのは，BanerjeeとGajskiであり，並列計算機上で**boolean recurrence**[BaGa84]ループを投機的実行する専用ハードウェアを提案している[BaGa84]．

boolean recurrenceとは，条件分岐を決定する真理値(Boolean)が，以下に示すように循環参照関係にあることを言う．

```
DO I=1,N
    IF  F(U(I-1),U(I-2),...,U(I-M1),V(I-1),...,V(I-M2))
    THEN  U(I)=A(I) OP1 B(I)
    ELSE  V(I)=C(I) OP2 D(I)
ENDDO
```

boolean recurrenceを持つループは，条件分岐が*Taken*であるか*Not Taken*であるによって，データ依存関係が変化し，イテレーション間で並列実行できない．このため，このようなループの並列化においては投機的実行が有効であることが示されている[BaGa84]．しかし，[BaGa84]では，対象ループを，ループ内に1つの条件分岐を持つ場合に限定しており，一般的な投機的実行方式ではない．

また，投機的実行時の並列性爆発(計算機資源枯渇)の問題に対して，[Uht92]では，1cycle毎に最大1イテレーションの実行開始のみを許すという仮定の下で，1つの条件分岐を持つループを投機的実行する際に必要となるPE数算出法を示している．しかし，「1cycle毎に最大1イテレーションの実行開始」という条件は，「コードを十分に持ち上げることができない」という点から，2.で述べたwindow sizeが小さい場合の投機的実行に相当し，投機的実行の効果が小さくなる．従って，妥当な仮定とは言えない．しかし，データ依存を考慮することにより，プロセッサ数を抑えることができるという点は着目できる．

[ThGH93]では，並列計算機上で投機的実行を行う際の並列性の爆発を防ぐ現実的な解として，条件分岐では，分岐予測により，常に片方向のみを投機的実行を行う方法を示している．片方向に無限に投機的実行する場合の速度向上を基準とした場合，5段の

条件ジャンプで50%，10段で90%の性能達成ができることが示されている．しかし，片方向のみに投機的実行を行う場合，「投機的実行の段数の増加に伴い，分岐予測の失敗確率が増加し，無限に投機的実行を行った場合の性能自体がOracle Modelに比較し小さくなる」ことを考えると，10段の90%という値は，Oracle Modelの数%～十数%に相当する．

[Uht92][ThGH93]では，並列計算機上での実行制御について述べられていないのに対し，[YMHM88]では，並列処理システム-晴-と呼ぶデータ駆動型計算機を対象にタスクレベル(マクロタスクと呼ぶ)での投機的実行方式を提案している．

[YMHM89]では，投機的実行の効果を上げるためのタスク生成手法として複数トレースをマクロタスク群として生成し，生成された複数トレースを同時に実行するフローグラフ展開手法を提案している．さらに，現実的な投機的実行の解として，doall型ループのような並列度の高い部分を超える投機的実行は，本来の計算を遅延させるという意味から，投機的実行すべきでない点を指摘している．doall型ループを超える投機的実行を制限すると条件分岐5段程度で速度向上率が飽和することをシミュレーションにより示している[YYKM91]．また，[YIYM91]では，内部にループを含むプログラムでは，扱うデータ規模の増大に伴って，投機的実行の効果が増大するものと逆に減少するものがあることを示している．

一方，[YYIM94]では，投機的実行の効果を上げ，かつ，投機的実行時の副作用を排除するタスク生成手法を提案している．3.5で述べたsuperblockやhyperblockが制御依存のみからブロック生成を行っているのに対し，提案手法は，制御依存及びデータ依存の両方を考慮し，投機的実行の効果の低い部分を1つのタスクに融合している．これにより，タスク間の依存関係を小さくでき，タスクを単位とした投機的実行の効果を大きくすることができる．従って，3.6で述べた基本ブロック単位で投機的実行を行うアーキテクチャ上で，基本ブロックの代わりに用いることにより大きな効果が得られると期待できる．また，タスクの実行制御条件を示し，一般的な並列計算機上での実現方法を示している．

[YYIM94]では，提案されたマクロタスクを用いた実行方式を並列計算機EM-4[SYHK89]上に実装し，タスクサイズによる投機的実行の効果を調べている[YSKS93]．タスクサイズが10命令(1命令はA(i)=B(i)×C(i))の時，投機的実行の理論性能の50%程度，100命令の時に90%程度を出せることを確認している．[YSKS94]では，マクロタスクの集中制御を廃した分散制御手法を提案・実装し，タスクサイズが10命令以上の時，実行時オーバヘッドを[YSKS93]の方法に比較し小さく抑えることができることを示している．

このように，並列計算機上での投機的実行では，❶Oracle Modelを目指すが並列性の爆発を抑える手法，❷投機的実行の効果を発揮できるタスク生成手法，❸実行オーバヘッドを小さくする制御方式，さらに，上記では述べられていないが，❹データの配置やタスクのスケジューリング，が重要な問題である．また，並列計算機上のプロセッサ内では，SuperscalorやVLIWが行っている投機的実行を含めた命令並列処理が可能であり，今後，これらを組み合わせた投機的実行が重要となると考えられる．

5．Unlimited Speculative Execution Scheme

本節では，[YYIM94]で提案したマクロタスク(以下MT)生成手法と，[YSKS94]で示した制御手法，及び，以下で述べる実行モデルを用いることにより，前節で挙げた❶❷❸の問題を解決する方法を与える．なお，現時点で，❹は未解決の問題である．

まず，[YYIM94]で提案したMT生成手法を適用し❷の問題を解決する．[YYIM94]では，(1)基本ブロック間の制御依存とデータ依存に基づき，投機的実行の効果の小さい部分を融合すると共に，(2)制御依存に関わらずデータ依存が一意になるようにMTを生成する．(1)によりMTの粒度を大きくすることができ，(2)により投機的実行時の副作用問題[YYIM94]を回避できる．さらに，MT内では，VLIWやSuperscalarやが行っている投機的実行を行う．これにより，MT内及びMT間の階層化された投機的実行が可能となる．基本ブロック間でデータ依存及び制御依存の両方を持つものをMTとして融合しているため[YYIM94]，MT内では条件分岐を超えて命令を大きく持ち上げることができない．すなわち，小さなwindow sizeでも，MT内の並列性を十分抽出できる．一方，MT内に複数の条件分岐と命令が存在するため，MT間での投機的実行では，(1)window sizeを見かけ上大きくすることができる，(2)ループを1つのMTとすることでループ間の並列性検出が可能となり，2.の❸の要件を満たすことができる．このように投機的実行を階層化することによる利点は大きい．

次に，❸の問題を解決するために，[YSKS94]で提案する分散制御方式を用いる．分散制御方式は，条件分岐の結果を各PEに放送することにより，PEで実行中の各々のMTが，自分自身で，次にとるべき手順を決定するという制御手法である．これにより，MT制御に伴うオーバヘッドがMT数に依存しなくなる．

最後に，与えられた計算機資源の範囲内でOracle Modelを実現し❶の問題を解決する．Oracle Modelは，無限の条件分岐のジャンプを仮定しており，全てのMTが同時に実行できる印象を持つ．しかし，4.で述べた[Uht92]で指摘されているように，データ依存を考えると，ある時点で実行を開始できるMT数は限定される．つまり，データ依存が解決されたMTから実行を開始させることにより，条件ジャンプの段数にとらわれない投機的実行が可能となる．つまり，計算機資源の範囲内でOracle Modelが実現できると考える．具体的には，制御依存とデータ依存の競争を行わせ(competitive executionと呼ぶ)，データ依

存の解決と制御依存の解決において，データ依存の解決の方が早い場合に投機的実行を行う(図4).

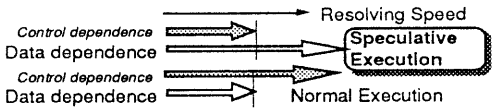

図4 competitive execution

これにより，対象とする並列処理計算機に応じた投機的実行が行われる．すなわち，通信コストやタスク起動コストが小さいほど，制御依存が解決する前に，より多くのMTを起動できることになり，広範囲に渡る投機的実行が可能となる．しかし，プロセッサ数が少ない場合には，資源の枯渇を引き起こす可能性がある．そこで，最大条件ジャンプ数NJを設定し，NJよりも先の投機的実行を制限する[YSKS94].

competitive executionに基づいた投機的実行方法を「MTの実行開始を明示することなく，条件ジャンプ段数を制限することなく投機的実行が行われる」という意味から**unlimited speculative execution scheme**と呼ぶ．図5に提案方式の概要を示す．

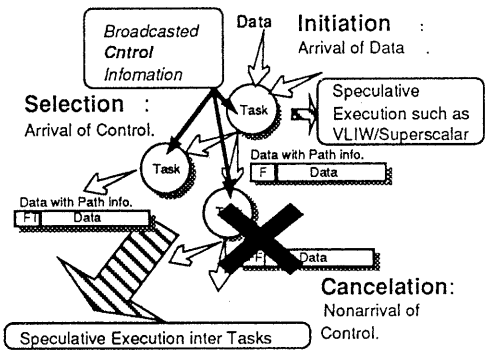

図5 unlimited speculative execution scheme

6．まとめ

本報告では，投機的実行の現状をサーベイすると共に，Unlimited Speculative Execution Schemeの構想を述べた．サーベイにあたっては，誤りのないように注意したが，誤り等がある場合には，ご指摘いただければ幸いである．今後は，5.で述べた方式について，(1)実行モデルの具体化，(2)シミュレーションによる性能評価，等を行っていきたい．

謝辞

本研究を遂行するにあたり御指導，御討論いただいた太田情報アーキテクチャ部長ならびに計算機方式研究室の同僚諸氏に感謝いたします．

参考文献

【まえがき】

[FeOW87] J.F.Ferrante, K.J.Ottenstein and J.D.Warren:"The Program Dependence Graph and Its Use in Optimization", ACM Trans. Prog. Lang. and Sys., 9, 3, pp.319-349 (1987)

【命令レベルの並列性と投機的実行】

[KiHI93] 木ノ内, 星子, 稲森:"並列演算と先行制御機能を持つプロセッサにおける命令間の論理的な依存特性と性能の関係", 信学論, 76-D-I, 8, pp.417-428 (1993)

[KuMC72] D.J.Kuck, Y.Muraoka and S.C.Chen:" On the Number of Operations Simultaneously Execute in Fortran-Like Programs and Their Resulting Speedup," IEEE Trans. Comput., 21, 12, pp.1293-1310 (1972)

[LaWi92] M.S.Lam and R.P.Wilson: "Limits of Control Flow Parallelism," Proc. of Ann. Symp. on Comput. Architecture, pp.46-57 (1992)

[NiFi82] A.Nicolau and J.A.Fisher:" Measuring the Parallelism available for Very Long Instruction Word Architecture",IEEE Trans. Comput., 33, 11, pp.968-976 (1984)

[RiFo72] E.M.Riseman and C.Foster: "The Inhibition of Potential Parallelism by Conditional Jumps,"IEEE Trans. Comput., 21,12, pp.1405-1411 (1972)

[ThGH92] K.B.Theobald, G.R.Gao, and L.J.Hendern:"On the Limits of Program Parallelism and its Smoothability," Proc. of 25th Ann. Int. Symp. on MicroArchitecture, pp.10-19 (1992)

[Wall91] D.W.Wall: "Limits of Instruction-Level Parallelism," Proc. Int. Conf on ASPLOS-IV, pp.176-188 (1991)

[YYUY92] 山本, 山口, 内海, 吉本, 長尾, 枝松:"マイクロプロセッサの構成のソフトウェアシミュレータによる性能解析", 情処研報, ARC-96-3, pp.17-24 (1992)

【VLIW / Superscalar上での投機的実行】

概要/分岐予測

[BaLa93] T.Ball and J.R.Larus: "Branch Prediction For Free," SIGPLAN93 Conf. on Prog. Lang. Desigin and Implementation, pp.300-313 (1993)

[Carg91] H.G.Cragon:"Branch Strategy Taxonomy and Performance Models," IEEE Comput. Soci. Press(1991)

[FiFr92] J.A.Fisher and S.M.Freudenberger: "Predicting Conditional Branch Directions From Previous Runs of a Program," Proc. Int. Conf on ASPLOS-V, pp.85-95 (1992)

[LeSm84] J.Lee and A.J.Smith:"Branch Prediction Strategies and Branch Target Buffer," IEEE Comut.,17,1,pp.6-22 (1984)

[Lilj88] D.J.Lilja : "Reducing the Branch Penalty in Pipelined Processors," IEEE Comput., 21, 7, pp.47-55 (1988)

[McHe86] S.McFarling and J.Hennessy:"Reducing Cost of Branches," Proc. of Ann. Symp. on Comput.Arch., pp.396-403 (1986)

[PaSR92] S.T.Pan, K.So and J.T.Rahmeh: "Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation," Proc. Int. Conf on ASPLOS-V, pp.76-84 (1992)

[YePa93] T.Y.Yeh and Y.N.Patt: "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History," Proc. of Ann. Symp. on Comput.Arch., pp.257-266 (1993)

ブースティング

[ANMH93a] H.Ando, C.Nakanishi, H.Machida, T.Hara and M.Nakaya: "Speculative Execution and Reducing Branch Penalty on a Superscalar Processor," IEICE Trans. Electron, E76-C, 7, pp.1080-1093 (1993)

[ANMH93b] H.Ando, C.Nakanishi, H.Machida, T.Hara, S.Kishida and M.Nakaya: "Speculative Execution and Reducing Branch Penalty in a Parallel Issue Machine," Proc. of IEEE Int. Conf. on Comput. Design, pp.106-113 (1993)

[ChLS92] M.C.Chang, F.Lai and R.J.Shang: "Exploiting Instruction-Level Parallelism with the Conjugate Register File Scheme," Proc. of 25th Ann. Int. Symp. on MicroArchitecture, pp.29-32 (1992)

[HANM93] 安藤, 町田, 中西, 原,中屋:"投機的実行のためのアーキテクチャ上の支援", 情処研報, ARC-105-5, pp.33-40 (1994)

[SmLH90] M.D.Smith, M.S.Lam and M.A.Horowitz: "Boosting Beyond Static Scheduling in a Superscalar Processor," Proc. of Ann Symp. on Comput. Arch., pp.344-344 (1990)

[SmLH92] M.D.Smith, M.S.Lam and M.A.Horowitz: "Efficient Superscalar Performance Through Boosting," Proc. Int. of Conf on ASPLOS-V, pp.248-259 (1992)

例外処理

[ErKr94] M.A.Ertl and A.Krall: "Delayed Exceptions - Speculative Execution of Trapping Instructions," Proc. of 5th Int. Conf. on Compiler Construction, LN.,786, pp.158-171 (1993)

[HaYM91] 萩本,山名,村岡:"並列処理システム－晴－の演算実

行機構の構成,"第39回情処全大,4Q-3,pp.6-71-72 (1991).

[MCBH93] S.A.Mahlke, W.Y.Chen, R.A.Bringmann, R.E.Hank and W.W.Hwu: "Sentinel Scheduling: A Model for Compiler-Controlled Speculative Execution," ACM Trans. on Comput. Syst., 11, 4 pp.376-408 (1993)

コンパイラ支援

[Char81] A.E.Charlesworth:"An Approach to Scientific Array Processing:The Architectual Design of the AP-120B/FPS-164 Family," IEEE Comput.,14,9,pp.18-7 (1981)

[CMCW91]P.P.Chang, S.A.Mahlke, W.Y.Chen, N.J.Warter and W.W.Hwu:"IMPACT:An Architectual Framework for Multiple-Instruction-Issue Processors," Proc. of Int. Symp. on Comput. Architecture,pp.266-275 (1991)

[CNOP88] R.P.Colwell, R.P.Nix, J.J.O'donnel, D.B.Papworth and P.K.Rodman: "A VLIW Architecture for a Trace Scheduling Compiler,"IEEE Trans. Comput., C-37, 8, pp.967-979 (1988)

[Fish81] J.A.Fisher:"Trace Scheduling: Technique for Global Microcode Compaction," IEEE Trans. Comput. C-30,7,pp.478-490(1981)

[HMCC93] W.W.Hwu,S.A.Mahlke,W.Y.Chen.P.P.Chang, N.J.Water, R.A.Bringman,R.G.Ouellette,R.E.Hank,T.Kiyohara, G.E.Haab,J.G.Holm and D.M.Lavery:"The Superblock:An Effective Structure for VLIW and Superscalar Compilation," J. of Supercomput.,7,1,pp.229-248 (1993)

[MLCH92] S.A.Mahlke, D.C.Lin, W.Y.Chen, R.E.Hank and R.A.Bringmann: "Effective Compiler Support for Predicated Execution Using the Hyperblock," Proc. of 25th Ann. Int. Symp. on MicroArchitecture, pp.45-54 (1992)

[Nico85] A.Nicolau:"Uniform Parallelism Exploitation in Ordinary Programs," Proc. of Int. Conf. on Parallel Processing, pp.614-618(1985)

[RYYT89] B.R.Rau,D.W.L.Yen,W.Yen and R.A.Towle:"The Cydra 5 Departmental Supercomputer," IEEE Comput.,22,1, pp.12-35 (1989)

[SSSH90] 斉藤, 新實, 柴山, 萩原: "低レベル並列処理計算機のためのマイクロプログラム最適化方式", 情処論, 31, 2, pp.307-315 (1990)

[TiLS90] P.Tirumalai, M.Lee and M.Schlansker: "Parallelization of Loops with Exits on Pipelined Architectures," Proc. of Supercomputing'90,pp.200-212 (1990)

[WMHR93] N.J.Warter, S.A.Mahlke, W.W.Hwu and B.R.Rau: "Reverse If-Conversion," SIGPLAN93 Conf. on Prog. Lang.Desigin and Implementation, pp.290-299 (1993)

アーキテクチャ上の支援

[AcKT86] R.D.Acosta, J.Kjelstrup and H.C.Torng:"An Instruction Issing Approach to Enhancing Performance in Multiple Functional Unit Processors," IEEE Trans. Comput.,35, 9,pp.815-828 (1986)

[AMNH94] 原, 安藤, 中西, 町田, 中屋: "スーパースカラ・プロセッサにおける分岐命令の並列実行", 情処研報, ARC-101-9 (1993)

[APBN91] A.Abnous, R.Potasman, N.Bagherzadeh and A.Nicolau: "A Percolation Based VILW Architecture," Proc. of 1991 Int. Conf. on Parallel Processing, 1, pp.144-149 (1991)

[AYKH94] 朝生, 柳瀬, 桐山, 林: "スーパースカラパイプラインによるブロック並列実行方式",情処研報, ARC-106-1 (1994)

[DwTo92] H.Dwyer and H.C.Torng : "An Out-of-Order Superscalar Processor with Speculative Execution and Fast, Precise Interrupts," Proc. of 25th Ann. Int. Symp. on MicroArchitecture, pp.272-281 (1992)

[HwPa87] W.W.Hwu and Y.N.Patt:"Checkpoint Repair for High-Performance Out-of-Order Execution Machines," IEEE Trans. Comput. 36, 12,pp.1496-1514 (1987)

[INOM93] 石井,野上,小野寺,三浦,村岡: "Restructured Instruction and Switched Context RISC Processor (RISC)^{2}の提案",情処研報, ARC-101-19 (1993)

[KIHM89] 久我, 入江, 弘中, 村上, 富田:"SIMP(単一命令流/多重命令パイプライン)方式に基づく「新風」プロセッサの低レベル並列処理アルゴリズム", 情処論, 30, 12, pp.1603-1611 (1989)

[KKSF93] 小松, 古関, 鈴木, 深沢: "拡張VLIWプロセッサGIFTにおける命令レベル並列処理機構", 情処論, 34, 12, pp.2599-2611 (1993)

[MIKT89] K.Murakami, N.Irie, M.Kuga and S.Tomita: "SIMP (Single Instruction stream/Multiple Instruction Pipelining): A Novel High-Speed Single-Processor Architecture," Proc. of Ann. Symp. on Comput. Architecture, pp.78-85 (1989)

[MNON91] 丸島,西,大沢,中崎:"パイプライン計算機における基本ブロック間並列処理アーキテクチャ"JSPP'91,pp.117-124 (1991)

[MoEb92] S.M.Moon and K.Ebcioglu: "Efficient Resource-Constrained Global Scheduling Technique for Superscalor and VLIW Processors," Proc. of 25th Ann. Int. Symp. on MicroArchitecture, pp.55-71 (1992)

[MoEb93] S.M.Moon and K.Ebcioglu: "On Performance and Efficiency of VLIW and Superscalar," Proc. of 1993 Int. Conf. on Parallel Processing, 2, pp.283-287 (1993)

[Mura91] 村上:"スーパースカラプロセッサの性能を最大限に引き出すコンパイラ技術",日経エレクトロニクス,251,pp.165-185 (1991.3.4)

[PiMe93] J.K.Pickett and D.G.Meyer: "Enhanced Superscalar Hardware: The Schedule Table," Proc. of Supercomputing'93 ,pp.636-644 (1993)

[SmPl88] J.E.Smith and A.R.Pleszkun:"Implementing Precise Interrupts in Pipelined Processors," IEEE Trans. Comput.,37,5, pp.562-573 (1988)

[Sohi90] G.S.Sohi:"Instruction Issue Logic for High-Performance Interruptible Multiple Functional Unit Pipelined Computers," IEEE Trans. Comput. 39,3,pp.349-359 (1990)

[Toma67] R.M.Tomasulo: "An Efficient Algorithm for Exploiting Multiple Arthmetic Units," IBM J. of Research and Development, pp.25-33 (1967)

言語

[JuHH93] D.S.Jusak, J,Hearne and H.Halliday: "Implementing Specuralive Parallelism in Possible Computational Worls," Proc. of Int. Conf. on Parallel Processing, 2, pp.292-296 (1993)

【並列計算機上での投機的実行】

[BaGa84] U.Banerjee and D.D.Gajski : "Fast Execution of Loops with IF Statements," IEEE Trans. Comput., C-33, 11, pp.1030-1033 (1984)

[SYHK89] S.Sakai,Y.Yamaguchi,K.Hiraki,Y.Kodama and T.Yuba:"An Architecture of a Dataflow Single Chip Processor," Proc. of Ann. Symp. om Comput. Arc.,pp.46-53 (1989)

[ThGH93] K.B.Theobald, G.R.Gao, and L.J.Hendern: "Speculative Execution and Branch Prediction on Parallel Machines," Proc. Int. Conf. of Supercomputing, pp.77-86 (1993)

[Uht92] A.K.Uht: "Requirements for Optimal Execution of Loops with Tests," IEEE Trans. Parallel and Distrib. Syst., 3, 5 pp.573-581 (1992)

[YHKM89]H.Yamana,T.Hagiwara,J.Kohdate,Y.Muraoka:"A Preceding Activation Scheme with Graph Unfolding for the Parallel Processing System -Harray-," Proc. of Supercomputing'89, pp.675-684 (1989).

[YIYM91] 山名,石崎,安江,村岡:"並列処理システムー晴ーにおける条件分岐の先行評価制御方式", 情処研報,ARC-89-19, pp.135-142 (1991).

[YMHM88]H.Yamana,T.Marushima,T.Hagiwara,Y.Muraoka:"System Architecture of Parallel Processing System -Harray-," Proc. of ACM Int. Conf. on Supercomputing, pp.76-89 (1988).

[YSKS93] 山名,佐藤,児玉,坂根,坂井,山口:"多段先行評価方式の並列計算機EM-4上での予備評価",情処学研報,HPC-48-14, pp.105-112(1993).

[YSKS94] 山名,佐藤,児玉,坂根,坂井,山口:"並列計算機EM-4における多段先行評価方式の分散制御方式",JSPP'94,pp.257-263(1994)

[YYIM94] 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,J77-D-I, 5, pp.343-353(1994)

[YYKM91] 山名,安江,神舘,村岡:"並列処理システムー晴ーにおける条件分岐の並列処理とその効果", 情処42回全大,4H-1(1991).