投機的実行研究の最新動向とタスク間投機的実行の有効性

1P－3

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

1．まえがき

投機的実行(Speculative Execution)に関して，94年～95年7月のサーベイを報告すると共に，我々が提案しているタスク間投機的実行[1]の有効性を示す．なお，94年までの調査については，文献[2]を参照していただきたい．

調査対象とした論文を**表1**に示し，近年の投機的実行に関する論文数の推移を**図1**に示す．図1に示すように，VLIWやSuperscalarが出始めた91年頃から投機的実行に関する論文が急増している．これらの研究は，(1)プログラムに内在する命令レベルの並列性調査，(2)Superscalar/VLIWでの投機的実行，(3)並列計算機での投機的実行に分類される．90年代前半は(1)に関する論文が多かったが，その後，(2)に関する論文が急増し，94-95年の論文はその中でも，分岐予測(**branch prediction**)と条件付実行(**predicated execution**)に関するものが全体の7割を占め，89-93年に多かったアーキテクチャ上の実現方法に関する論文が激減した．本報告では，現在最もホットな話題となっている分岐予測と条件付実行を中心に説明する．

表1 調査対象論文

・IEEE Trans. Computer	・Hawai Int.Conf.
・IEEE Trans. Parallel&Dist.	・ISCA
・ICS	・Supercomputing
・ASPLOS　・ICPP	・PACT
・IPPS	・JSPP
・IEICE Trans.	・IPSJ Trans.
・IEICE Tech. Rep.	・IPSJ Tech. Rep.

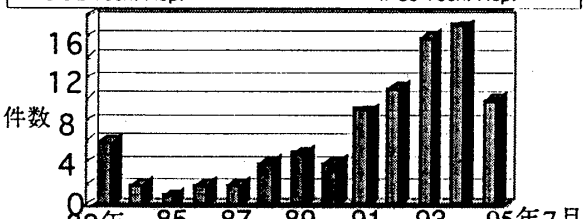

図1 投機的実行に関する論文数の推移

2．分岐予測と条件付実行

分岐予測及び条件付実行は，Superscalar/VLIW計算機の命令多重度を満たすことを目的とする投機的実行である．これらの手法によって，パイプラインハザードによるパイプラインステージの空スロットを埋める．

2.1 分岐予測(branch prediction)

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，空スロットを埋める手法であり，**静的**な方法と**動的**な方法に分類さる．現在、最も効率的であると言われているのは，**ミシガン大のYehとPattの方法**[3]と，**IBMのPanらの方法**[4]の動的な方法(96%～97%のヒット率)である．ミシガン大の方法は，**PAs**(Per-address Adaptive Branch Prediction using per-set histry table)，IBMの方法は**GAs**(Global Adaptive Branch Prediction using per-set history table)と呼ぶ[3]．PAsは，**図2**に示すように，直前のm回の分岐方向(ヒストリ)と分岐命令のアドレス(下位k-bit)及び分岐命令の種類(p種類)から1つの2bitカウンタ(**BHT:Branch History Table**)を用いて分岐方向を予測する．2bitのBHTは，4状態のオートマトンを構成する．これに対しGAsは，直前のm回の分岐方向と分岐命令の種類(p種類)からのみBHTを選択する．

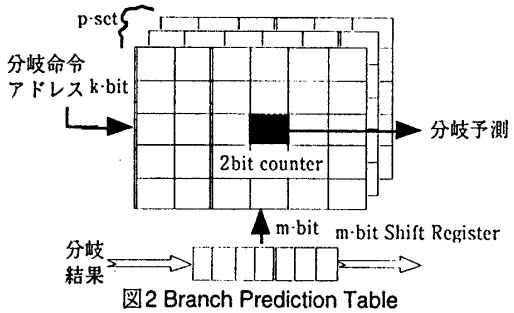

図2 Branch Prediction Table

PAsとGAsを比較した場合，cost-effectiveなのはPAsであり，8Kbitの資源を使用する場合，m=6，k=1K，p=16が最もよいと報告されている[3]．しかし，[5]では，misfetch(unconditional jump等)とmispredict(conditional jump等)では，リカバリのペナルティがmispredictの方が大きいことに着目し(misfetch:2cycle, mispredict:10cycle)，最も単純なGAg(GAsでp=0としたもの)で十分としている．また，予測した結果をヒストリとしてフィードバックすると，予測ヒット率が低下する[6]．さらに，kが十分でないと同一BHTが複数の分岐で共有され平均2.39%(k+m=10の条件の元，kを0～10に可変させた場合)予測ヒット率が低下[7]する．

動的な予測に対し，コンパイル時に静的に予測する方法がある．この場合，BHTが不要であり，ハードウエア量を小さく出来るという利点がある．元々は，[8]の分岐の種類と分岐命令の出現位置から予測していたが，近年では，動的手法と同様にヒストリを用いている．プロファイルから頻出するヒストリを取り出し，プログラムを再構成し，動的なヒストリと同様な効果を得る．これにより，約17%の性能向上が得られると**ハーバード大のYoungとSmith**[9]が報告している．また，[10]ではプロファイルを用いる方が正確なトレース(ブランチパタンではなくパスとして認識／BHTのaliasing問題)を利用でき，動的な方法に劣らないことを指摘している．一方，プログラムのコード量増加を抑える手法として，頻出パスをstate machineで構成する手法が提案されている[11]．

2.2 条件付実行(predicated execution)

命令に実行条件を示すオペランドを追加し，条件付で

Survey of Speculative Execution and the Effect of Task-level Speculation

Hayato YAMANA, Mitsuhisa SATO, Yuetsu KODAMA, Hirohumi SAKANE, Shuichi SAKAI†, Yoshinori YAMAGUCHI
Electrotechnical Laboratory　†Real World Computing Partnership

(ガード付きで)実行するのを条件付実行と呼ぶ．条件付実行では，条件決定後にハードウェアにより有効・無効化を行う．**安藤ら**は条件付で投機的実行結果をバッファリングする機構により，条件付実行を制限なく，例外処理を含めて実現するハードウェアを提案している[12].

条件付実行では，従来の命令セットを条件付命令に変更しなければならず，命令セットの変更コストが大きい．この為，[13]では，condが成立した場合にsrc→destへデータを転送する**cmov dest,src,cond**と呼ぶ命令を提案し，cmov命令のみの追加で，条件付実行を利用しない場合の33%の速度向上(8命令同時発行のマシン)が得られるとしている(ただしコード量は46%増加)．また，4命令同時発行のマシンでは，本手法で十分としている．同様に，[14]では，連続するm命令を1命令でガードする**GUARD cond,m**を提案し，通常の命令への変更を無くしている．

一方，[15]では，条件付実行は，シャドウレジスタ等多くのハードウェアを必要とするため，ハードウェアがno-trapping命令をサポートすれば，Superblock Schedulingにより条件付実行と同等な性能が得られるとしている．

2.3 分岐予測と条件付実行の融合

PAsを提案した**ミシガン大のPatt**らは，PAsを用いた場合でも，96%程度の予測ヒット率であるため，予測が困難な分岐命令を抽出し，それらの命令に対して条件付実行を行うという手法を提案している[16]．動的予測率が75%以下である分岐命令を抽出し，条件付命令で実行すると仮定したところ，8命令同時発行のマシンでcompress,eqntottが20〜23%性能向上が得られると報告している．しかし，ハードウェア量増大の問題が残る．

2.4 分岐予測のキャッシュへの影響

分岐予測を用いた場合，mispredictした命令によってキャッシュのヒット率が影響を受ける．[17]では，90%の予測ヒット率の時，キャッシュミスの増加は10%程度であると報告している．また，mispredict時にキャッシュされたデータの50%は無駄ではなく，後に使用される．

3．タスクレベルの投機的実行

前節で述べた命令レベルではなく，[1][2]で述べているようなタスクレベルで投機的実行を行うことにより，従来扱うことの出来なかったループ間・イテレーション間・タスク間の投機的実行が可能となる．[1]では，投機的実行に適したタスク生成手法及び，タスクの制御オーバヘッドを隠滅するタスク制御方法を提案している．本タスク制御手法により，並列計算機が持つ通信性能とタスクサイズから，実行時に動的に投機的実行の段数(Speculation Depth)が決定される．また，[18]ではマルチスレッド間で投機的実行を行わずに最早の実行開始タイミングで実行を開始する手法を提案([18]では投機的と呼んでいる)している．このように，並列計算機上での投機的実行の研究が，近年，重要となってきている．

例えば，ループ内の条件分岐で，一方の分岐が選択されるとloop carry dependenceが存在するが，もう片方の分岐が選択された場合には，loop carry dependenceが存在しないといったループを考える．このようなループは頻繁に現れる[18]．タスクレベルの投機的実行では，loop carry dependenceのない側の各イテレーションを複数同時に実行開始でき，全てのイテレーションでloop carry dependenceがない側が最終的に選択されたとすると，最大，イテレーション回数倍の速度向上が得られる．このような複数のイテレーション間に渡った投機的実行は，従来のSuperscalarやVLIWでは得らず，タスク間投機的実行が得意とする分野である．

4．まとめ

本報告では投機的実行に関する最新のサーベイを行い，近年の傾向として，branch predictionとpredicated executionが多くの研究機関で研究されていることを示した．また，タスクレベルの投機的実行の研究も不可欠であり，今後は，タスクレベルの投機的実行の効果を実際のアプリケーションで評価していく予定である．

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．

文献

[1] Hayato Yamana, Mitsuhisa Sato, Yuetsu Kodama, Hirofumi Sakane, Shuichi Sakai, and Yoshinori Yamaguchi: "A Macrotask-level Unlimited Speculative Execution on Multiprocessors", Proc. of ICS'95, Balcelona, Spain, pp.328-337 (1995.7)

[2] 山名,佐藤,児玉,坂根,坂井,山口:"投機的実行の現状とUnlimited Speculative Execution Schemeの提案",情処研報,ARC-107-14(SWoPP'94),pp.105-112 (1994.7)

[3] T.Yeh,Y.Patt:"A Comparison of Dynamic Predictions that use Two Levels of Branch History", Proc. of 20th ISCA, pp.257-266 (1993.5)

[4] S-T Pan, K.So, J.T.Rahmeh:" Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation", Proc. of 5th ASPLOS, pp.76-84 (1992.10)

[5] B.Calder,D.Grunwald:"Fast&Accurate Instruction Fetch and Branch Prediction",Proc. of 21th ISCA,pp.2-11 (1994)

[6] A.R.Talcott,W.Yamamoto,M.J.Serrano:"The Impact of Unresolved Branches on Branch Prediction Scheme Performance", Proc. of 21th ISCA,pp.12-21 (1994)

[7] A.R.Talcott,M.Nemirovsky,R.C.Wood:"The Influence of Branch Prediction Table Interference on Branch Prediction Scheme Performance", Proc. of PACT'95,pp.89-97 (1995.6)

[8] T.Ball, J.Larus:"Branch Prediction for Free", Proc. of 1993 PLDI, pp.300-313 (1993.6)

[9] C.Young, M.D.Smith:"Improving the Accuracy of Static Branch Prediction Using Branch Correlation", Proc. of 6th ASPLOS, pp.232-241 (1994)

[10] C.Young,N.Gloy,M.D.Smith:"A Comparative Analysis of Schemes for Correlated Branch Prediction", Proc. of 22th ISCA, pp.276-286 (1995.6)

[11] A.Krall:"Improving Semi-static Branch Prediction by Code Replication", Proc. of 1994 PLDI,pp.97-106 (1994)

[12] H.Ando,C.Nakanishi,T,Hara,M.Nakaya:"Unconstrained Speculative Execution with Predicated State Buffering", Proc. of 22th ISCA, pp.126-137 (1995.6)

[13] S.A.Mahlke et.al.:"A Comparison of Full and Partial Predicted Execution Support for ILP Processors", Proc. of 22th ISCA, pp.138-148 (1995.6)

[14] D.N.Pnevmatikatos, G.S.Sohi:"Guarded Execution and Branch Prediction in Dynamic ILP Processors", Proc. of 21th ISCA, pp.120-129 (1994)

[15] P.P.Chang et.al.:"Three Architectual Models for Compiler-Controlled Speculative Execution",IEEE Trans. Comput.,44,4,pp.481-494 (1995.4)

[16] P.Y.Chang,E.Hao,Y.N.Patt,P.P.Chang:"Using Predicated Execution to Improve the Performance of a Dynamically Scheduled Machine with Speculative Execution",Proc. of PACT'95,pp.99-108(1995.6)

[17] J.Pierce, T.Mudge:"The Effect of Speculative Execution on Cache Performance", Proc. of 21th ISCA,pp.172-179 (1994)

[18] P.K.Dubey et.al.:"Single-Program Speculative Multithreading(SPSM) Architecture: Compiler-assisted Fine-Grained Multithreading", Proc. of PACT'95, pp.109-121 (1995.6)