九州大学学術情報リポジトリ
Kyushu University Institutional Repository

クリティカルパスを特定するためのヒューリスティックスの改善

千代延, 昭宏
九州工業大学大学院情報工学研究科

佐藤, 寿倫
九州大学システムLSI 研究センター

https://hdl.handle.net/2324/6321

出版情報：第4回先進的計算基盤システムシンポジウム, pp.264-265, 2006-05. 情報処理学会,電子情報通信学会,IEEE Computer Society Japan Chapter
バージョン：
権利関係：ここに掲載した著作物の利用に関する注意 本著作物の著作権は（社）情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。

クリティカルパスを特定するためのヒューリスティックスの改善

千代延 昭宏[†] 佐藤寿倫[††]

Improving Critical Path Identification Accuracy

Akihiro Chiyonobu[†] and Toshinori Sato[††]

1. はじめに

近年携帯情報端末や組み込みシステムにおいても高い処理性能が求められるようになっており，高性能なプロセッサが搭載されている．しかしこれらのシステムには高い処理性能が求められる一方で，その消費する電力も少量であることが求められている．

この問題に対して我々は，プログラム実行中のクリティカルパス (Critical Path: CP) に着目して研究を行っている[6]．これまでの検討では，各命令をクリティカルであると特定する方法が不十分であるため，性能と省電力の両立に必ずしも成功していない[6]．そこで，本稿では上記のアーキテクチャ実現に不可欠なクリティカルパス特定法について考察する．

2. クリティカルパス情報を利用した電力削減

CP とは命令間の依存関係を結んだ鎖のうち最長のものを結んだ実行パスであり，プログラムの実行時間を決定する命令列である[5]．我々は実行サイクル数を増やさずにプロセッサの低消費電力化を実現するため，プログラムの実行時間を決めるという CP の特性に着目した[6]．動作速度と消費電力の異なる演算器を用意し，実行時間に影響を与える CP 上の命令は高速かつ電力消費の大きな演算器で実行し，CP 上にない命令は低速かつ電力消費の小さな演算器で実行する．この結果，実行時間を増やすことなくプロセッサの消費電力を抑えることが可能となる．

2.1 クリティカルパスを特定する機構

CP を特定するためのモデルとして Fileds らのモデル[4] がある．彼らのモデルはリオーダバッファ容量などのハードウェア制限，データが作られた時間を考慮したデータ・フローグラフ (Data Flow Graph: DFG) を用いて CP を特定し，正確な CP を特定可能である．ただしトレース情報を用いるため，ハードウェアとして実装することは不可能である．

実行される命令がクリティカルか否かを判断する機構として，CP 予測器 (Critical Path Predictor: CPP)[3),4),6)] とパス情報テーブル (Path Information Table: PIT)[5)] がある．Tune らと筆者らが提案している CPP[3),6)] を用いたこれまでの検討では，CPP の予測精度が不十分であるため，必ずしも性能と省電力が両立できるとは限らないことが分かっている．一方，PIT は命令ウィンドウ内の命令の DFG を作成し，その最長パスを特定する．このため，上述の CPP よりも正確に CP を特定できる．Fields らの CPP[4)] と PIT はハードウェアが複雑で，機構自身が多くの電力を消費する．したがって，複雑度の小さな CPP を改良する必要がある．CPP は各命令の履歴に基づいて予測を行う．このため，実行された命令がクリティカルか否かを特定するヒューリスティックスの正確さが CP 予測精度を左右する．

2.2 クリティカルパス予測器の更新情報

これまでの研究で，ヒューリスティックスが特定する CP 情報は十分に正しくないことが分かっている[7)]．このため，我々は以下の新たなヒューリスティックスを考案した．

- 命令ウインドウ中で最古の命令，もしくはアクティブリスト中で最古の命令をクリティカルとする (QA)
- 予測ミスした分岐命令をクリティカルとする (BM)
- キャッシュミスしたロード命令をクリティカルとする (L1, L2)
- キャッシュミス中に実行された命令はノンクリティカルとする (E-DFUG)[2)]

各ヒューリスティックスは，それぞれ組み合わせて使うことができる．

3. 評価方法と結果

SimpleScalar ツールセット[1)] を利用して，各ヒューリスティックスと Fields らの CP モデルを組み込んだシュミレータを作成した．PISA 命令セットを用いた．ベンチマークは SPEC 2000 CINT である．どのプログラムも，先頭の 20 億命令でウォームアップし，続く 5 億命令をシミュレーションした．

† 九州工業大学大学院 情報工学研究科
Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
†† 九州大学 システム LSI 研究センター
System LSI Research Center, Kyushu University

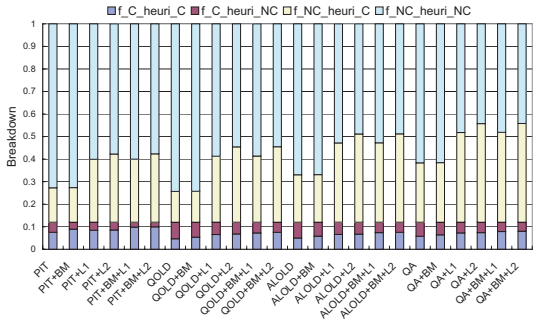

図 1 PIT と各ヒューリスティックスの結果

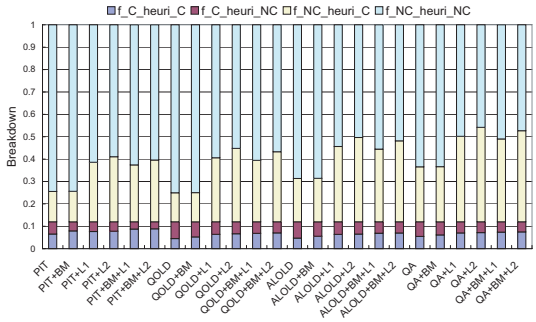

図 2 E-DFUG を適用した場合の結果

シミュレーション結果を図 1 と図 2 に示す．紙面の都合上それぞれの結果の平均のみを掲載する．横軸は PIT と Tune らによって提案されたヒューリスティックス (QOLD，ALOLD)[3]，今回提案するヒューリスティックスを表している．縦軸は Fields らの正しい CP モデルとの一致を示している．下から，正しい CP モデルとヒューリスティックスの両方がクリティカル (C) だった場合 (`f_C_heuri_C`)，正しい CP モデルは C だったが，ヒューリスティックスではノンクリティカル (NC) だった場合 (`f_C_heuri_NC`)，正しい CP モデルでは NC だったがヒューリスティックスでは C だった場合 (`f_NC_heuri_C`)，正しい CP モデルとヒューリスティックスの両方が NC だった場合 (`f_NC_heuri_NC`) をそれぞれ示している．図 2 は各ヒューリスティックスに加え，E-DFUG を適用した場合の結果である．

各ヒューリスティックスを見る．ヒューリスティックスを組み合わせるにつれ，正しく C を特定する頻度が増すが，NC の場合に C とする頻度も増えることが分かる．しかし BM を適用した場合は，間違った特定を増やすことなく C と特定する頻度が増加している．E-DFUG を適用した場合，`f_C_heuri_C` が減り，`f_NC_heuri_NC` が増加していることがわかる．このことから，E-DFUG を用いた場合に正しく NC を特定する頻度が増すことがわかる．正しい CP モデルが C とする部分については，若干 `f_C_heuri_C` が減り．`f_C_heuri_NC` が増える．しかしその差は最大で 1%，大部分は 0.05% 以下である．これは以下の理由による．一般にメモリ階層の上位へのアクセスはプロセッサから見ると非常に遅い．これによりメモリアクセス中に実行される命令は，メモリアクセスを引き起こした命令の実行終了後には NC になるためである．逆に当該メモリアクセス命令に依存する命令は C となる．

4. まとめと今後の課題

CPP を更新する情報を作るヒューリスティックスについて，その正確さを調査した．その結果，BM や E-DFUG という新たなヒューリスティックスを用いることで，その精度が向上することがわかった．今後はこれらのヒューリスティックスが作る情報で CPP を更新し，その予測精度を調査する予定である．

謝　　辞

本研究の一部は，文部科学省科学研究費補助金 (No. 16300019, No.176549) の援助によるものです．

参 考 文 献

1) D. Burger, T. M. Austin, “The SimpleScalar Tool Set, Version 2.0”, Technical Report CS-TR-97-1342, Computer Science Department, University of Wisconsin Madison, June 1997.
2) A. Chiyonobu, T. Sato, “Energy-Efficient Instruction Scheduling Exploiting Memory Access Slack”, Memory performance: Dealing with Applications, systems and architecture, September 2005.
3) E. S. Tune, D. Liang, D. M. Tullsen, B. Calder, “Dynamic Prediction of Critical Path Instructions”, the 7th International Symposium on High Perdormance Computer Arcitecture, January 2001
4) B. Fileds, S. Rubin, R. Blodik, “Foucusing Processor Policies via Critical-Path Prediction”, the 28th International Symposium on Computer Architecture, July 2001.
5) 小林良太郎, 安藤秀樹, 島田俊夫, “データフロー・グラフの最長パスに着目したクラスタ化スーパースカラ・プロセッサにおける命令発行機構”, 2001 年並列処理シンポジウム JSPP2001, 2001 年 6 月.
6) 千代延昭宏, 佐藤寿倫, 有田五次郎, “低消費電力プロセッサアーキテクチャ向けクリティカルパス予測器の提案”, 情処研報 2002-ARC-149, 2002 年 8 月.
7) 千代延昭宏, 佐藤寿倫, “プログラムの実行時における命令の重要度決定に関する検討”, 情処研報 2003-ARC-154, 2003 年 8 月.