様式 C-19

科学研究費補助金研究成果報告書

平成 23 年 5 月 9 日現在

機関番号： 31303
研究種目： 基盤研究(C)
研究期間： 2008 ～ 2010
課題番号： 20500046
研究課題名（和文） 静的データ依存関係に基づく命令ステアリング方式に関する研究

研究課題名（英文） Instruction Steering Based on Static Data Dependency

研究代表者
鈴木 健一 （SUZUKI KEN-ICHI）
東北工業大学・工学部・准教授
研究者番号： 50300520

研究成果の概要（和文）：現在のマイクロプロセッサは，複数の命令を並列実行することで，高速処理を実現している．本研究では，プログラムを機械語に翻訳する際に，実行速度に重要な影響を与える命令を予め抽出(静的データ依存の解析)しておき，実際にプログラムを実行するときには，簡単な処理(命令ステアリング)だけで済むようにすることで，従来方式よりも高効率な処理を実現する．研究期間内に行なった評価では，静的ステアリングでも従来方式と変わらない性能が得られることを示した．

研究成果の概要（英文）： Modern microprocessors achieve high performance by executing multiple instructions in parallel. In this research, we have introduced a new execution model where instructions in a local critical path are statically found at the compile time, and only the instruction steering is dynamically performed at execution time. From the performance evaluations, we have shown that the IPC of our execution model is comparable to that of existing models, even in the case of no dynamic steering.

交付決定額

（金額単位：円）

	直接経費	間接経費	合 計
2008 年度	1,300,000	390,000	1,690,000
2009 年度	1,000,000	300,000	1,300,000
2010 年度	1,000,000	300,000	1,300,000
年度			
年度			
総 計	3,300,000	990,000	4,290,000

研究分野： 計算機科学
科研費の分科・細目： 情報学・計算機システム・ネットワーク
キーワード：計算機アーキテクチャ，データフォワーディング，レジスタファイル

１．研究開始当初の背景

(1) スーパースカラプロセッサや VILW プロセッサは，命令レベル並列性(ILP; Instruction Level Parallelism)を利用して並列処理を行なうことで，高い IPC (Instructions Per Cycle) を実現する．さらに，近年のマイクロプロセッサでは，深い段数のパイプライン化により，動作周波数を向上させている．一方，SMT(Simultaneous Multi-Threading) などにより，ILP を越えた並列性が確保できるようになってきている．したがって，空間的並列性(多数の演算器の利用) と時間的並列性(深いパイプライン) を効果的に利用するマイクロアーキテクチャの確立が求められている．

(2) パイプラインが深くなると，データハザードを避けるための複雑なフォワーディングパスが必要になる．特に，スーパースカラプロセッサでは，複数の演算器(FU; Function Unit) を相互結合する必要から，さらに複雑なフォワーディングパス構成となる．今後，半導体プロセスの微細化により配線遅延が増大すると，フォワーディングパスへの要求は極めて厳しいものになっていく．
図 1 は，フォワーディングパスの制限を緩和するために，演算器毎にレジスタファイルを配置したクラスタ化プロセッサの例である．

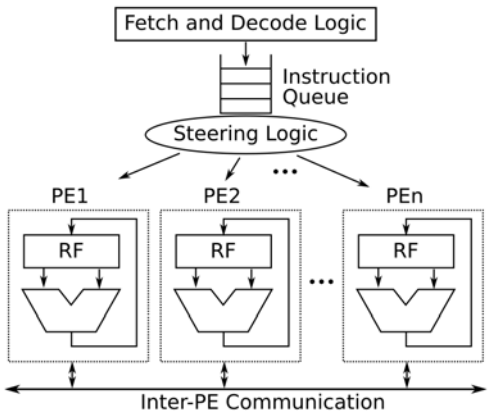

図 1: クラスタ化プロセッサ

２．研究の目的
(1) 高 ILP を生かし切るには，データ依存性を考慮し，メモリから FU へのデータ供給を低レイテンシで実現する必要がある．したがって，これからのマイクロアーキテクチャには，「簡潔なフォワーディングパス」と「低レイテンシのデータロード機構」による高 ILP の活用が必要不可欠である．

(2) 本研究では，コンパイル時の静的データ依存解析を利用し，この 2 項目の実現を目指す．スーパースカラアーキテクチャが利用する命令の動的依存関係に加えて，コンパイル時に把握する静的依存関係をも利用して，命令スケジューリングを行なう方式を確立することを目的とする．

３．研究の方法
(1) プログラム中に含まれる各命令を各演算器(図 1 での PE に相当する)に割り当てる処理のことを命令ステアリングと呼ぶ．従来のスーパースカラプロセッサでは，命令ステアリングは，動的に行なわれていた．すなわち，コンパイラが生成したコードは，実行時に命令キューに取り込まれた後で初めてステアリングを受ける．一方，本研究では，静的命令ステアリングを考慮する．すなわち，コンパイル時に得られる情報を予めコード中に埋め込んでおき，それをステアリングに利用する．

(2) 静的ステアリングのためのデータ依存解析を行なうには，コンパイラを新規生成あるいは既存のものを改造することが考えられる．しかし，これでは，長期の開発期間が必要となると予想されたことから，本研究では，既存コンパイラが生成したコードについて，データ依存解析を行なうツールを開発することにした．これにより，開発期間を短期に留めることが可能になる．このツールは，命令間のデータ依存関係を解析し，結果をシミュレータに提供する．

(3) 静的ステアリングを行なうための指標として，本研究では，基本ブロック内でのクリティカルパスを用いる．分岐命令の分岐条件と飛び先は実行時にならないと判明しないことから，静的ステアリングアルゴリズムでは基本ブロックを越えた解析が困難であるからである．各基本ブロック内の命令列を後方から逆に探索しつつ，各命令のオペランドの依存関係を調べることで，最も時間のかかるパスを見つけることができる．これが基本ブロック内のクリティカルパスである．上述のツールは，クリティカルパスに含まれる全ての命令にマークを付けることで，「重要な命令」として提示する．

(4) 最も簡単な静的ステアリングアルゴリズムとして，クリティカルパスに含まれる命令を一つの演算器に割り当て，それ以外のものは，ラウンドロビン式に割り当てるものが考えられる．クリティカルパスに含まれる命令の遅延は，全体の実行時間に与える影響が他の命令に比べて大きいと考えられるからである．本研究では，この方式を static と呼ぶ．

(5) 動的ステアリングアルゴリズムの代表的なものとして，Mod 方式がある．これは，各命令の依存関係に関係なく，ラウンドロビンで演算器を選ぶものである．また，依存関係を重視する方式として，DB (Dependency-Based) がある．一方，依存関係だけを考えてステアリングを行なうと，特定の演算器に負荷が集中してしまい，高い性能が得られなくなることから，負荷分散をパラメタとして用いる RMBS が知られている．

(6) これらの既存ステアリング手法と，静的ステアリング方式の有効性を確認するために，性能評価が必要である．実機を用意できない新しいマイクロプロセッサの性能評価には，通常，ソフトウェアシミュレーションが用いられる．本研究では，マルチコアのプ

ロセッサについて，命令レベルのシミュレーションを行なわないとその妥当性を示すことができない．そこで，マルチコアプロセッサについて，新規のシミュレーション環境を構築し，性能評価を行なう必要がある．シミュレーション環境としては，まず，SimpleScalar を基礎とし，クラスタ化されたプロセッサを評価できるように改良したものを用意した．

(7) 研究途中で，SimpleScalar による評価では計算時間が膨大になることが判明したことから，Valgrind および Simics を用いるシミュレーション環境を構築した．

(8) これらのシミュレーション環境で，各ステアリング手法の IPC や命令毎の遅延を評価し，静的ステアリング方式の有効性を示す．

４．研究成果

(1) 最も簡単な静的ステアリング方式として，上述した static と既存の動的ステアリング方式の比較を SimpleScalar を用いた性能評価により行なった．ここでは，パイプラインステージを IF, ID, Map, Issue, Reg-read, Ex, Commit の７段とし，Map ステージでレジスタリネーミングと命令ステアリングを行なうことにした．その他の主要なアーキテクチャパラメタを表１に示す．

表 1: 主要なアーキテクチャパラメタ

Branch Predictor	Tournament
Fetch Unit	16 insts/cycle
Inst. Window Size	256
Total Issue Width	8
Number of Int. PEs	8
Int-FU Latencies	ALU=1, MUL=7, DIV=12
Inter-PE Comm. Delay	4 cycles
Load/Store Queue	64-entry

PE 間通信とステアリング方式の関連を強調して見るために，他の演算器からのオペランド読み込みに 4 サイクルかかるとしている．また，メモリ遅延の影響を受けないように，L2 キャッシュではミスが発生しないと仮定した．

(2) ベンチマークプログラムは，整数アプリケーションである SPEC2000int に含まれる crafty, gcc, gzip, mcf, parser, twolf, vortex, vpr を用い，それぞれ，1G 命令をスキップした後の 10M 命令についてサイクル精度でのシミュレーションを行なった．ステアリング方式としては，RMBS, DB, Mod, static を用いた．

(3) 各ベンチマークプログラムについての IPC (Instructions Per Cycle) を Mod 方式の IPC で正規化したものを図２に示す．RMBS は依存関係と負荷分散の両方を考慮した動的ステアリング方式であり，ほとんどのプログラムについてよい IPC を達成している．DB は依存関係だけを考慮する動的ステアリング方式であり，vortex と mcf についてだけ高い IPC を得ているが，他のプログラムについては，負荷の集中のため，IPC が低下してしまう．本研究で提案している static は，全てのプログラムについて RMBS と同等か，より高い IPC を達成できていることが分かる．

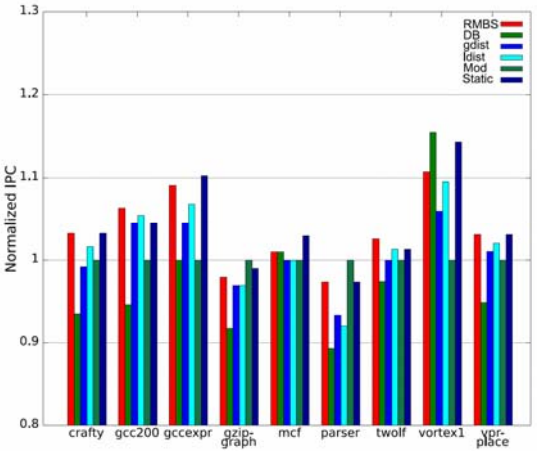

図 2:SPECint2000 での正規化 IPC
(Mod の IPC で正規化)

(4) この結果を詳細に分析するために，1 命令当たりの平均通信回数と 1 サイクル当たりの演算待ち平均命令個数を調べ，図 3 および図 4 に結果を示す．まず，DB は，依存関係を考慮して，データ依存のある命令を同一演算器に割り当てることから，図 3 に示されるように，通信量は最小にできている．しかし，その代償として，図 4 に示されるように，負荷分散が全くうまくいっておらず，各演算器に大量の待ち命令が発生してしまうことが分かる．逆に，Mod 方式は，負荷分散が最高になる(図 4)ものの，依存のある命令も別々の演算器に割り当ててしまうため，通信量が大きくなってしまっている(図 3)．

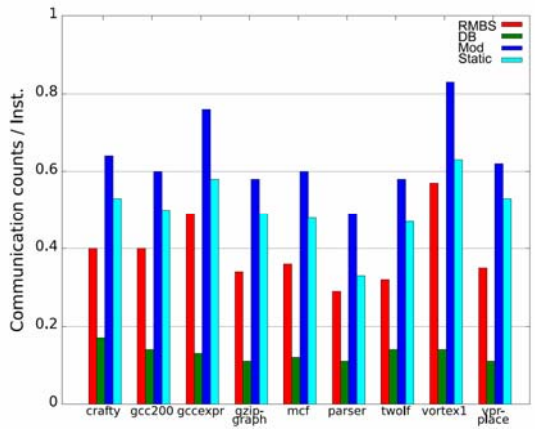

図 3: SPEC2000int での 1 命令当たり平均通信回数

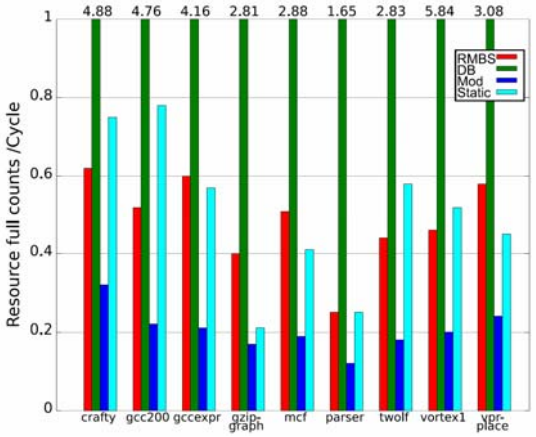

図 4: SPEC2000int での 1 サイクル当たり平均演算器待ち命令数

(5) これらの方法とは異なり,RMBS と static は，通信量と負荷分散のバランスを取ることで，高い IPC を実現する．図 4 について RMBS と static を比較すると，どちらが良いかはプログラムにより異なるのに対し，図 3 について見ると，常に RMBS のほうが優れていることが分かる．したがって,static 方式には，通信量の削減において，改良の余地があると言える．

(6) これは，static が，クリティカルパスに含まれない命令について，依存関係を全く無視してステアリングしていることに原因があると思われる．静的にはクリティカルパス上になかった命令が，実行時にはクリティカルパスに含まれるということも往々にして起こりうるからである．これには，分岐の影響だけでなく，通信遅延も影響を与える．実際，シミュレータ上で動的に数え上げを行なうと，ほとんどのプログラムで，クリティカルパスにない命令が過半数を超えており，それらの命令のステアリングも性能に大きな影響を及ぼしていることが推測される．

(7) 以上が本研究の現時点での主な成果であるが，本研究を進めていくうちに，いくつかの新しい課題を見出すことができた．

① クリティカルパス情報を実行時のステアリングに生かす方法が必要である．static のような単純な方法では既存の RMBS などの性能を超えることは困難である．しかし，実行時にハードウェアの力を借りて，クリティカルパス情報と負荷の分散について考慮することで，より高い IPC を達成できる可能性は極めて高い．すなわち，静的クリティカルパス探索と動的ステアリングを融合させたハイブリッド方式である．

② 本研究は，クラスタ化プロセッサを念頭に置いて行なってきたが，今後はメニーコアプロセッサが主流となっていくと予想されることから，その観点での見直しが必要である． SimpleScalar はマルチコアに適応できないことから，これを独自に拡張するか，Simics などを利用することが考えられる．

③ メモリの影響の考慮が必要である．多くのプログラムでメモリアクセス遅延は無視できない．これまでの評価では，キャッシュをほぼ完璧なものとして扱ってきたが，実際には，メモリ命令でのストールは頻発する．例えば，全てのメモリ命令をクリティカルパスに含まれると仮定した評価などが考えられる．

④ 命令の並べ替えに関して試行する価値がある．これまでの評価では，クリティカルパスの解析しか行なっておらず，命令は元のコンパイラが出力した通りの順でフェッチされる．命令列の解釈に問題がない範囲で，クリティカルパス解析に基づき，命令を並べ替えることで，ステアリングの自由度を拡大することが可能になる．

⑤ 基本ブロックを超えたデータ依存解析の可能性を探る．これまでの静的解析では，基本ブロック内での依存しか考慮していなかった．これによりツールは単純になったが，基本ブロックはそれほど多くない数の命令で構成されていることから，クリティカルパスの分析の精度が低くなってしまう．その結果(6)で分析したような結果となっていた．何らかのハードウェアによる支援が必要となるが，分岐を超えてクリティカルパスをつなぐ仕組みを作れれば，その精度は大きく向上するはずである．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）
① Ken-ichi Suzuki, Yoshiyuki Kaeriyama, Kazuhiko Komatsu, Ryusuke Egawa, Nobuyuki Ohba, and Hiroaki Kobayashi, ″A Fast Ray-Tracing Using Bounding Spheres and Frustum Rays for Dynamic Scene Rendering,″ IEICE Transactions on Information and Systems, Vol. E93-D, No. 4, pp. 891-902, 2010, 査読有.

〔学会発表〕（計５件）
① 澁谷宏，鈴木健一，分散レジスタファイルを持つプロセッサのハイブリッド命令スケジューリング，平成 23 年東北地区若手研究者研究発表会，平成 23 年 3 月 12 日，仙台高等専門学校.
② 若松広大，鈴木健一，分散レジスタファイルを持つプロセッサの静的命令スケジューリング，平成 22 年東北地区若手研究者研究発表会，平成 22 年 2 月 26 日，東北学院大学.
③ 平間聖隆，鈴木健一，Valgrindによるマイクロプロセッサ評価，平成 22 年東北地区若手研究者研究発表会,平成22年2月26日，東北学院大学.
④ 鈴木健一，分散レジスタファイル向け静的命令スケジューリング，第８回情報科学技術フォーラム(FIT2009),平成21年9月4日，東北工業大学.
⑤ 鈴木健一，クラスタ化プロセッサの静的命令スケジューリングに関する研究，電子情報通信学会東北支部先端技術シンポジウム，平成 21 年 3 月 16 日，東北工業大学.

６．研究組織
(1)研究代表者
鈴木　健一　（SUZUKI KEN-ICHI）
東北工業大学･工学部･准教授
研究者番号：50300520

(2)連携研究者
小林　広明　（KOBAYASHI HIROAKI）
東北大学･サイバーサイエンスセンター･教授
研究者番号：40205480