C-006

分散レジスタファイル向け静的命令スケジューリング
Static Instruction Scheduling for a Distributed Register File

鈴木健一†
Ken-ichi Suzuki

1. はじめに

命令レベル並列性 (ILP) は，マイクロプロセッサに搭載された複数の演算器を並列に動作させる上で重要な並列性である．命令実行時に抽出することも比較的容易であることから，スーパースカラプロセッサに代表される多くのプロセッサが，これを利用している．ところが，単純に ILP を利用して多数の演算器による並列処理を行なう場合，演算器群が一つのレジスタファイル (RF) を共有しているため，多ポートの RF が必要となる．多ポート RF は，基本的にポートのハードウェアを多重化することで構成しなければならないため，ハードウェアコストと消費電力が大きくなってしまう．

本研究では，多ポート RF の問題点を緩和するため，RF を分割し，複数の RF 群からなる構成を対象とする．例えば，クラスタ化アーキテクチャ [1] は，少数の演算器群毎にハードウェアを分割し，処理を局所化することを狙った構成方式であるが，このようなアーキテクチャでは，演算器群毎に，分割 RF を配置することが考えられる．分割されたそれぞれの RF は，ローカルの演算器群とのデータ授受だけをすればよいことから，少ポート数の RF で十分であり，ハードウェアコストと消費電力の点で有利である [3]．以下では，RF を分割する構成のことを分散 RF と呼び，また，分割された演算器群のことを PE と呼ぶことにする．図 1 に，n 個の PE からなる分散 RF を持つクラスタ化プロセッサの構成例を示す．

分散 RF を用いる場合 [2]，RF 間通信が多発しないように，処理をローカル RF に局所化することが重要である．というのは，命令が必要としているオペランドがローカル RF に存在せず，リモートの RF にある場合，RF 間で通信を行なって，そのオペランドを入手する必要があるからである．命令をどの RF に割り当てるかを決める処理を命令ステアリングと呼ぶ．適切な命令ステアリングを行なって，できるだけ RF 間通信が発生しないようにしなければならない．

また，分散 RF では，ローカル PE 内の演算器数が限られているため，特定の PE に多数の命令を割り当ててしまうと，演算器待ちが多発してしまうことがある．上述のように，分散 RF によるクラスタ化プロセッサでは，PE 間通信を避けるため，一つの PE に多数の命令が集中的にステアリングされがちである．この場合，集中的に使用される PE の RF において，局所的に演算器をはじめとする計算資源の不足が起こる．

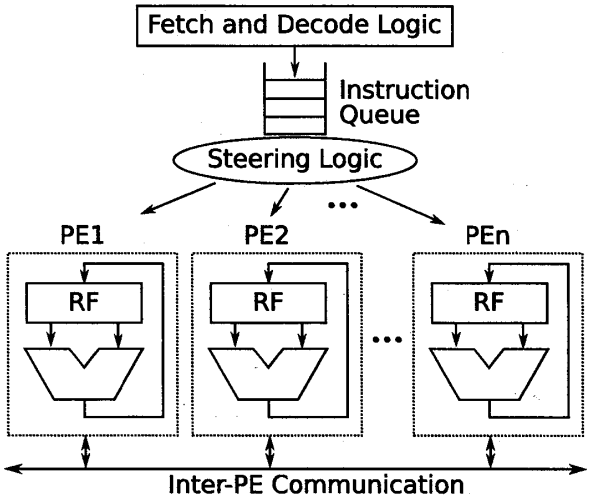

図 1　分散 RF を持つクラスタ化プロセッサ

本研究では，分散 RF によるプロセッサにおいて，コンパイル時に命令間の依存関係から静的クリティカルパス解析を行ない，その情報を実行時の命令ステアリングに利用して PE 間通信を減少させることで，IPC (Instructions Per Cycle) の向上を目指す．さらに，クリティカルパス上にない命令については，PE 間負荷を分散させるために，負荷の低い PE に割り当てることで，PE 間通信と負荷分散のよいトレードオフを達成する．

2. 命令ステアリング

分散 RF を持つクラスタ化プロセッサにおいて，並列性の利用だけを狙って，異なる PE に命令を分散させてしまうと，それらの命令間にデータ依存があった場合に PE 間通信が多発し，通信による遅延が深刻となってしまう．逆に，PE 間通信を避けるために，特定の PE にだけ命令を集中させると，並列性を十分に利用できなくなってしまう．したがって，このようなクラスタ化プロセッサでは，命令ステアリング方式が特に重要である．

データ依存関係を考慮するステアリング方式として，Dependence Based (DB) [4] がある．この方式では，各 PE の負荷を考慮せずに，データ依存のある命令を同一の PE に割り当てる．図 2 を用いて，データ依存関係を考慮するステアリングについて簡単に述べる．ここでは `i1`〜`i5` の 5 個の命令か

† 東北工業大学, Tohoku Institute of Technology

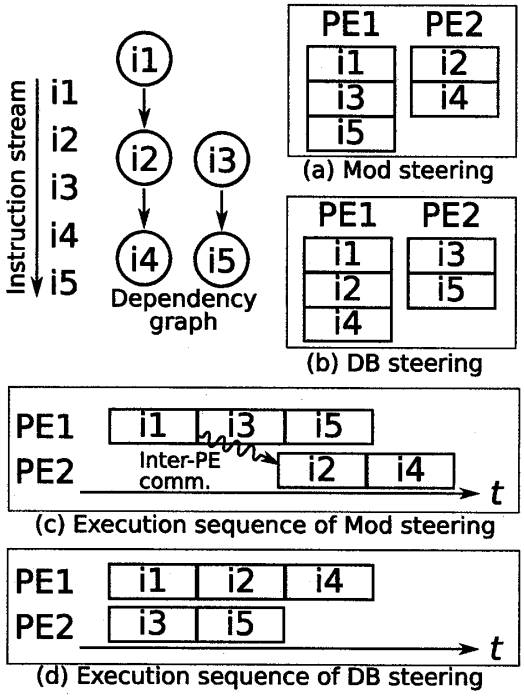

図 2　データ依存を考慮した命令ステアリング

らなる命令列を PE1 および PE2 の 2 個の PE にステアリングすることを考える．各命令のデータ依存関係を表すデータ依存木が図の左上に示されている．これらの命令をラウンドロビンで割り当てる Mod ステアリングでは同図 (a) のように命令が割り当てられる．この場合，命令実行は同図 (c) のように，PE 間通信による遅延を大きく受けてしまう．一方，DB 方式によるステアリングでは，命令間の依存関係により命令を割り当てるので，同図 (b) のように命令割り当てがなされ，同図 (d) のように PE 間通信の遅延を避けることができる．

しかし，先述したように，依存関係だけを重視してステアリングを行なうと特定の PE に命令が集中して割り当てられ，レジスタや演算器といった資源がその PE において局所的に欠乏するという問題がある．そこで，資源競合も考慮したステアリング方式として，Advanced RMBS [5] (ARMBS) が提案されている．ARMBS では，基本的に各命令を依存関係のある PE に割り当てるが，負荷バランスが悪くなったときには，もっとも負荷の低い PE に割り当てる．

命令ステアリングは，つまるところ，PE 間通信と負荷分散のトレードオフ点を探すという問題に帰着されるが，実行時の命令発行に影響を与えない短い時間で，ハードウェアコストをかけずに解決しなければならないという点に困難さがある．

3. クリティカルパスと命令ステアリング

データ依存関係を重視して命令ステアリングを行なうと，命令列に本来備わっている並列性を維持したまま，PE 間通信を削減できる．しかしながら，特定の PE に多数の命令を割り当てると，負荷の集中した PE で資源欠乏の問題が発生する．たとえば，リネーミングレジスタや演算器の空きがなくなり，命令発行ができなくなる．

この問題を緩和するために，既存手法では，各 PE に割り当てられた命令数などの指標を用い，負荷の均衡を図ってきた [5]．ところが，このような方式では，実行のセマンティクスに無関係に，他 PE への命令の割り当てが行なわれてしまうため，必ずしも良いステアリングになるとは限らない．

そこで本報告では，命令列からクリティカルパスを抽出し，クリティカルパスに含まれる命令を優先的に，負荷状態は無視して，依存関係のある命令の置かれた PE に割り当てることを考える．命令のクリティカリティとは，その命令の実行レイテンシが 1 サイクルでも増加したら，プログラム全体の実行時間が増加してしまうという性質として定義される [6]．したがって，クリティカルでない命令は，他の PE に割り当てられて PE 間通信のペナルティを受けたとしても，全体の実行時間が受ける影響は少ないと考えられる．しかしながら，現実の計算機における命令実行は，命令間の制御依存ならびにデータ依存，またキャッシュを含むメモリアクセスなどの予測不能な要素が関係することから，真のクリティカルパスを求めることは極めて困難である [6] [7]．

本報告では，クリティカルパス情報を用いた命令ステアリング方式の有効性を検証するために，コンパイル時に基本ブロック内の静的なクリティカルパス情報を取得し，実行時の命令ステアリングを行なう．コンパイル時には，基本ブロックを越えた命令列の解析ができないことが欠点となるが，基本ブロック内の全命令についてのデータ依存木を作成することができることから，詳細なクリティカルパス解析ができる．以下では，コンパイル時にデータ依存木を作成し，クリティカルパスを抽出する処理のことを静的命令スケジューリングと呼ぶ．

4. 静的命令スケジューリング

静的命令スケジューリングでは，コンパイル時に基本ブロック内の命令列からデータ依存木を作成し，それを基にクリティカルパス解析を行なう．本報告では，コンパイラを作りなおすのではなく，コンパイラが生成したコードにクリティカルパス情報を追加するポストプロセッサを作成し，追加されたクリティカルパス情報を実行時にステアリングロジックが利用するという方式をとる．

本報告で採用する静的命令スケジューリングと命令ステアリングの処理手順を以下に示す．

- 静的命令スケジューリング

（1） コンパイラが生成したコードを分岐命令を区切りとして分割する．

（2） 分割されたコードブロックについて，データ依存木を生成する．

表 1　主要なアーキテクチャパラメタ

Branch Predictor	Tournament predictor	Fetch Unit	16 insts/cycle
Inst. Window Size	256	Total Issue Width	8
Number of Int. PEs	8 (with one ALU and MUL/DIV)	Number of FP PEs	1 with 1 ADD and 1 MUL
Int-FU Latencies	ALU=1, MUL=7, DIV=12 (cycles)	FP-FU Latencies	ADD, MUL=4, DIV=12 (cycles)
Inter-PE Comm. Delay	4 cycles	No. of Regs per PE	64
Load/Store Queue	64-entry load and 64-entry store queues		
I1 Cache	256KB, 2-way, 64byte lines, 3cycles	D1 Cache	256KB, 2-way, 64byte lines, 1cycles
L2 Cache	Unified, ideal (No miss occurs.)		

（3） データ依存木の中で，最長の遅延時間を持つパスをクリティカルパスとし，当該パスに含まれる命令にマークを付加する．

- 実行時の命令ステアリング

（1） マークのない命令は，負荷が最小の PE に割り当て．

（2） マークのある命令は，依存している命令と同じ PE に割り当てる．ただし，2 個の命令に依存している場合には，マークされている依存命令 (もしあれば) を優先する．

5. 性能評価

Alpha 21264 のシミュレータである Sim-alpha [8] を基礎として，クラスタ化プロセッサの実行ドリブンシミュレータを構築した．パイプラインステージは 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．整数 PE 数は 8 とし，各 PE には ALU と乗除算器を 1 個ずつ配置した．本報告では，整数アプリケーションだけを扱うこととし，浮動小数点演算を行なう PE を 1 個だけ用意した．以上の構成に合わせて，表 1 のようにアーキテクチャパラメタを設定した．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 PE の RF からのオペランド読み込みに，やや大きめの 4 サイクルを要することとし，メモリ遅延の影響を少なくするために，L2 キャッシュではミスが発生しないものとした．

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，ARMBS, DB, Mod, 静的スケジューリング，および，ldist, gdist [9] を用いた．ldist と gdist は，命令の依存関係とそれらの発行時間差に着目したステアリング方式である．

各ベンチマークプログラムについての IPC (Instructions Per Cycle) を Mod ステアリング方式の IPC で正規化したものを図 3 に示す．ARMBS は，依存関係と負荷分散を考慮した方式であり，ほとんどのプログラムについて，よい IPC を

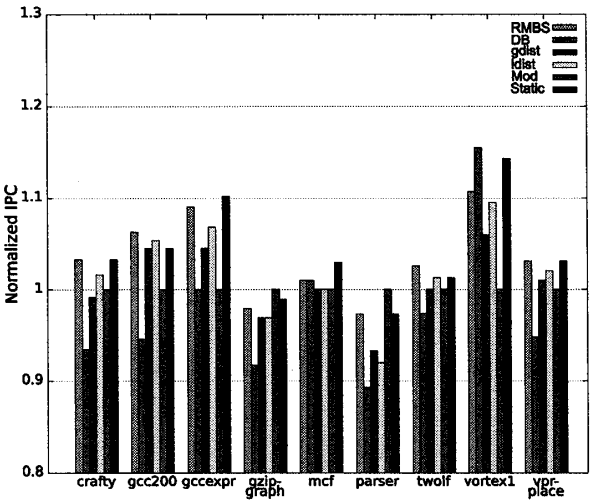

図 3　SPECint2000 での正規化 IPC
(Mod ステアリングの IPC で正規化)

達成している．DB は依存関係だけを考慮するステアリング方式であるため，vortex, mcf については高い IPC を得ることができているが，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．ldist および gdist については，分散 RF を対象に提案された方式ではないものの，ほとんどのプログラムについて，かなり高い IPC を得ている．本報告で提案している静的スケジューリング方式は，全てのプログラムについて，ARMBS と同等かより高 IPC を得ることができた．

結果をさらに分析するために，1 命令当たりの平均 PE 間通信回数と 1 サイクル当たりの演算待ち命令個数をそれぞれ図 4，図 5 に示す．まず，DB と Mod はそれぞれ，通信量の低減と負荷分散だけを考慮する方式であることが明瞭に示されている．ARMBS と静的方式は，両者のバランスを取る手法である．図 5 について ARMBS と静的方式を比較すると，どちらが良いかはプログラムにより変わるのに対し，図 4 について見ると，常に ARMBS のほうが優れていることが分かる．したがって，静的方式は，PE 間通信の削減の点において，難が残っているといえる．

これは，現在の静的命令スケジューリング方式が，クリティカルパス上にない命令については，依存関係を完全に無視し

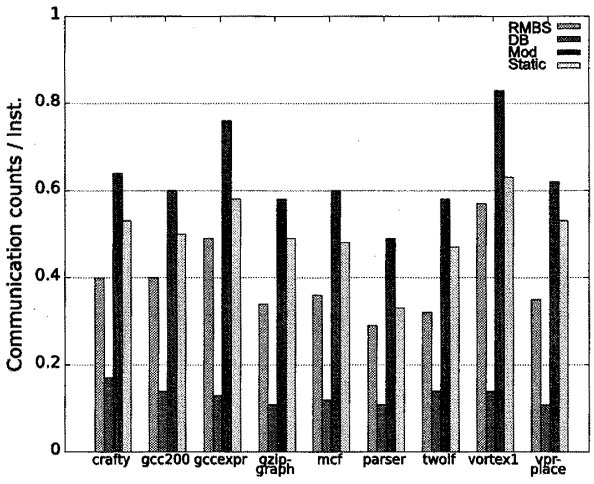

図 4　SPECint2000 での 1 命令当たり平均 PE 間通信回数

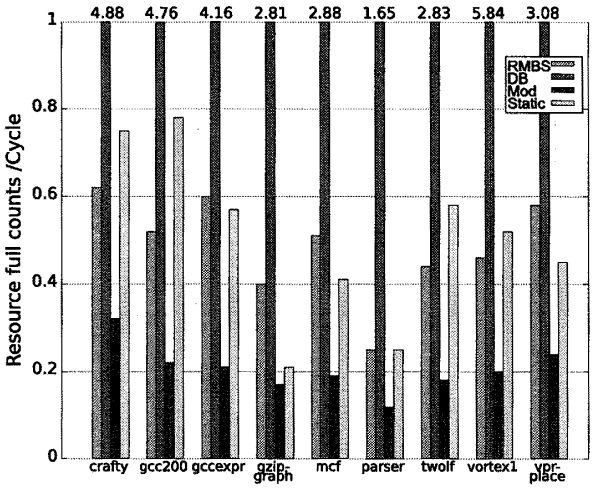

図 5　SPECint2000 での 1 サイクル当たり平均演算器待ち命令個数

て，最小負荷の PE に割り当て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を依存関係を無視して割り当ててしまうと，PE 間通信の遅延が加わり，動的にはクリティカルパスになってしまう，という現象が発生する．実際，動的に数え上げると，ほとんどのプログラムでクリティカルパス上にない命令が過半数を越えており，それらの命令のステアリングにも十分な配慮が必要である．たとえば，クリティカルパス上にある命令は無条件で依存関係のある PE に割り当てを行ない，それ以外の命令については，ARMBS や ldist, gdist によるステアリングを行なうことが考えられる．

なお，ARMBS や ldist, gdist が，静的スケジューリング方式よりも高い IPC を記録することも多いが，これらの方式は，スケジューリングにいくつかの経験的な値を用いており，本質的にどんなプログラムにも適用できるかは不明である．一方，静的スケジューリング方式は，静的クリティカルパスという命令列が本来持っている性質を利用するものであり，経験的な値は一切必要としないということも重要である．

6. おわりに

本報告では，分散 RF を持つクラスタ化プロセッサについて，コンパイル時の静的クリティカルパス解析の結果を動的命令ステアリングに利用することを提案した．SPECint2000 ベンチマークのサイクル精度シミュレーションの結果から，提案方式が多くのプログラムについて，PE 間通信と負荷分散を両立して高い IPC を実現できることを示した．しかしながら，本方式は PE 間通信の削減の点でまだ改良の余地があることも明らかになった．今後，静的クリティカルパス解析と既存のステアリング手法の組み合わせ，あるいは動的クリティカルパス解析器との連携を視野に入れて，研究を続ける予定である．

文　献

[1] S.Palacharla, N.Jouppi, and J.Smith, "Complexity-effective superscalar processors," ISCA-24, pp.206–218, 1997.

[2] J.Llosa, M.Valero, and E.Ayguade, "Non-consistent dual register files to reduce register pressure," HPCA-1, pp.22–31, 1995.

[3] Y.Sato, K.Suzuki, and T.Nakamura, "Partitioned register file designs for clustered architectures," Journal of Information, Vol.9, No.1, pp.119–134, 2006.

[4] A.Baniasadi, and A.Moshovos, "Instruction distribution heuristics for quad-cluster, dynamically-scheduled, superscalar processors," MICRO-33, pp.337–347, 2000.

[5] R.Canal, J.Parcerisa, and A.Gonzalez, "A cost-effective clustered architecture," PACT-99, pp.160–168, 1999.

[6] 小林良太郎，市川彰孝，島田俊夫，スラック命令数を増加させるスラック共有化手法，情報処理学会研究報告，2007-ARC-172, pp. 2007.

[7] 千代延昭宏，佐藤寿倫，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，情報処理学会研究報告，2002-ARC-149-2, 2002.

[8] R.Desikan, D.Burger, and S.Keckler, "Measuring Experimental Error in Microprocessor Simulation," ISCA-28, pp.266–277, 2001.

[9] 服部直也，高田正法，岡部淳，入江英嗣，坂井修一，田中英彦，発行時間差に基づいた命令ステアリング方式，情報処理学会論文誌:コンピューティングシステム，Vol.45, No.SIG11 (ACS 7), pp.80-93, 2004.