5K-2

フロントエンド・パイプラインを最小化する命令キャッシュ・アーキテクチャの置換アルゴリズムの改良

宮永 瑞紀† 伊達 三雄‡ 倉田 成己‡ 五島 正裕‡ 坂井 修一‡

†東京大学 工学部 ‡東京大学大学院 情報理工学系研究科

1 はじめに

近年広く普及しているマルチコア・プロセッサにおいて，熱密度の抑制や面積効率の改善が性能向上には重要になっている．そのコアとして用いられているスーパースカラ・プロセッサの中でも，レジスタ・リネーミングとディスパッチという2つのロジックは面積が大きく，また利用頻度も高いために発生する熱量も相当に大きい．

これらのロジックを省略する手法として，ディスパッチされたあとの命令列をキャッシュに格納して再利用するDispatched Image Cache(DIC)[1]が提案されている．

現在DICのキャッシュ置換アルゴリズムにはキャッシュ・ラインへの直近のアクセスのみに着目したLRU(Least Recently Used)が用いられているが,「再利用できるキャッシュ・ライン」が「再利用されないキャッシュ・ライン」によって追い出されてしまい，キャッシュ・ヒット率が低下しがちである．DICの容量を増やせばヒット率も向上するが，容量の増加に伴いDICの回路面積も増加してしまう．

そこで本研究ではDICの容量を低く抑えながらその性能を向上させるために，キャッシュ置換アルゴリズムに2つの改善を行っている．ひとつは，再利用の可能性の高いキャッシュ・ラインを残すことによりDICヒット率を高めることである．もうひとつは，キャッシュ・ラインへのアクセス履歴以外の情報も利用し，DICのミス時の性能低下が小さく，ヒット時の性能向上が大きくなるようなキャッシュ・ラインを優先してDICに残すことである．

An improved replacement algorithm for an instruction cache architecture that minimizes the front end pipeline

Mizuki Miyanaga†, Mitsuo Date‡, Naruki Kurata‡, Masahiro Goshima‡and Syuichi Sakai‡

†Faculty of Engineering, The University of Tokyo

‡Graduate School of Information Science and Technology, The University of Tokyo

2 Dispatched Image Cache

DICはリネーミングが完了し各サブ・ウィンドウにディスパッチされた後の命令列の「イメージ」を格納する命令キャッシュである．

DICヒット時には「リネーミング及びディスパッチ済み」の命令列が得られるのでそれらの処理を行う必要がない．またそのロジックに充てられていたパイプライン・ステージを省略することができ，分岐予測ミス等のペナルティは削減され性能向上が図れる．DICミス時にのみ通常のリネーミングとディスパッチを行うが，このときの処理を少ないリネーミング/ディスパッチ幅で時間をかけて行うならば，その面積を大幅に削減することもできる．その場合，性能低下を抑えるには十分なDICヒット率が要求される．

ただし，DICではその構造により2つの原因からキャッシュの利用効率が低下している．ひとつは，同一の命令であってもそれまでの実行経路によって依存する命令が異なることがあり，実行経路毎に区別してキャッシュしているからである．もうひとつは，DICは各サブ・ウィンドウに対応するようにセグメントを区切っており，命令種の出現頻度の偏りによってはキャッシュ・ラインが充填されない可能性があるからである．

3 DICでのキャッシュ置換アルゴリズム

前述のようにDICではキャッシュ利用効率が低下しており，本研究はこれを改善することを目的としている．

現在のDICで用いられているキャッシュ置換アルゴリズムであるLRUは，最も長い期間使用されていなかったキャッシュ・ラインを置換対象とする．[2]そのため長い実行経路を順次，非反復的に通った場合には，それが「その後しばらくは利用されない経路」であっても，それまでにキャッシュされていた「頻繁に通る経路」のキャッシュ・ラインが追い出されてしまう．特にキャッシュの容量が大きくない場合には多くのキャッシュ・ラインが一度も再利用されないまま追い出されてしまいやすい．

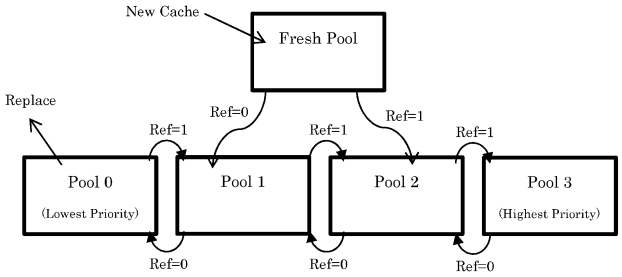

図 1: ジェネレーショナル・リプレースメントの構造

本研究では DIC の容量を小さく抑えながら性能を向上させるため，LRU のようなキャッシュ・ラインへのアクセス履歴の利用に加え

- 再利用されたキャッシュ・ラインを優先して残すことにより DIC ヒット率を向上させる
- DIC ヒット /ミス時の影響を考慮し DIC の効果を向上させる

ようなキャッシュ置換方法を提案する．

3.1 キャッシュ・ラインの再利用可能性

プログラムのループ構造により再利用されたことのあるキャッシュ・ラインは，その後も再利用される可能性が高い．そこで，「最近に一度だけ使われたキャッシュ・ライン」より「過去に頻繁に再利用されたキャッシュ・ライン」を優先して DIC に残せば，まれな実行経路によって再利用可能なキャッシュが追い出されてしまうことを防ぐことができる．

そのようなキャッシュ置換アルゴリズムの一つとして，**ジェネレーショナル・リプレースメント** [3] がある．ジェネレーショナル・リプレースメントでは **図 1** のようにキャッシュを複数の階層（世代）に分け，一定サイクルの後にアクセスのあったものを上位の階層へと昇格，アクセスの無かったものを下位の階層へと降格する．それまでに頻繁にアクセスのあったキャッシュ・ラインは上位の階層に分類され，逆に「最近に追加されたが再利用されていない」キャッシュ・ラインは下位の階層へと押しやられ，先に追い出される．これにより，DIC ヒット率を向上させることができると考えている．

3.2 DIC ヒット /ミス時の影響

LRU では命令の種類や依存関係にかかわらずキャッシュ優先度が決められるが，2 章 で述べたようなパイプラインの省略などの DIC の効果が必ずしも発揮されない場合がある．

たとえば他の命令の実行結果に依存する命令の場合，データ・ハザードが発生してこの命令をすぐに実行できないことがある．また先行するロード /ストア命令などのキャッシュ・ミスにより命令ウィンドウが詰まり，構造ハザードにより命令ウィンドウに命令を書き込めないことがある．これらのような場合，DIC がミスしても命令が実行されるまでの猶予が長く，その間に最小限のレジスタ・リネーミング，ディスパッチ・ロジックで時間を掛けて解析することができる．つまりこのような命令は DIC の キャッシュ・ミスの影響が小さいと言え，逆に ディスパッチ後すぐに実行されるような命令列は DIC ヒットの効果が大きく，優先して残すべきと考えられる．これについては，キャッシュされた命令列が実行されるまでの時間をキャッシュ優先度の評価に用いることで実現できる．

本研究ではこのようにキャッシュ・ヒット/ミス時の影響をキャッシュ優先度の評価に反映し，DIC ヒットの効果を高める．

4 まとめ

本稿では DIC とそのキャッシュ置換アルゴリズムの改善方法について述べた．これによりスーパースカラ・プロセッサの性能を保ちつつ熱効率と面積効率を向上させる DIC の効果を改善することができる．

今後は DIC における命令種や命令間の関係ごとのキャッシュ効果の評価を行い，置換アルゴリズムを改良，評価する予定である．

参考文献

[1] 伊達 三雄，倉田 成己, 塩谷 亮太, 五島 正裕, 坂井 修一: レジスタ・リネーミングとディスパッチ・ネットワークを最小化するプロセッサ・アーキテクチャ, 先進的計算基盤システムシンポジウム SACSIS2012, pp.280-288 (2012) .

[2] John L. Hennessy and David A. Patterson, 成田 光彰 訳: コンピュータの構成と設計 第 2 版 [下], 454p, 日経 BP 社, 東京 (1999) .

[3] Erik G. Hallnor and Steven K. Reinhardt: A fully associative software-managed cache design, ISCA '00, pp.107-116 (2000) .