クラスタ型スーパスカラ・プロセッサにおける
分散投機メモリフォワーディング手法の提案

入江 英嗣

1 はじめに

情報処理の中核となるマイクロプロセッサには常に性能向上が期待されている。マイクロプロセッサは、デバイス微細化、スーパパイプライニング技術、並列実行技術によって性能を向上させてきた。しかし、これらの技術の熟成が進んだ結果、現在のスーパスカラ方式は、肥大したデータパスに配線遅延が大きく影響することが予想され、性能向上の限界が指摘されている [1]。

この問題へのアプローチとして、次世代プロセッサの魅力的な選択肢として注目されている方式がクラスタ型スーパスカラ・プロセッサである。この方式では実行コアを複数のクラスタで構成し、発行キュー、レジスタ、データパス等を分散する。処理が複数クラスタに分散することによって生じるオーバヘッドを許容すれば、従来クリティカルパスとなっていた発行キューやデータパス等のユニットが分散小型化し、更なるスーパパイプライニング効果が期待できる。

このような動機によってクラスタ化スーパスカラ方式が採用された時、クラスタ化による高速化に追随できないフロントエンド処理やメモリ参照処理の遅延は従来よりも相対的に大きくなることが予想される。本論文ではこのような高クロック指向のクラスタ型スーパスカラ・プロセッサを想定し、メモリ参照によるオーバヘッドに注目する。また、クラスタ構成に適した手法として、“分散投機メモリフォワーディング” を提案し、メモリ参照オーバヘッドの軽減を目指す。

2 関連研究

Palacharla ら [2] は微細化に伴って生じる配線遅延がプロセッサのどの部分に強く影響するかを調べ、プロセッサクラスタ化の利点を示した。クラスタ化されたプロセッサモデルについて様々に検討が行われたが、近年では現行のプロセッサとコード互換性を持ち、動的に命令を分散処理するクラスタ化スーパスカラ方式が、様々な要素技術のベースラインモデルとして用いられている。Parcerisa ら [3] は命令をどのクラスタで実行するかを決定するステアリング方式について、高性能な方式の提案を行った。一方、関連研究の多くがメモリ参照を理想化しているのに対して、Rajeev ら [4] は、メモリ参照のオーバヘッドにより、複数クラスタによる並列実行が効率よく稼動しないことを指摘している。

3 ベースライン・モデル

3.1 ベースライン・モデルの概要

ベースラインモデルでは、小規模で高速に動作できるクラスタと深いフロントエンド処理パイプラインを持つ、高クロック指向のクラスタ化設計を想定した。また、メモリ参照処理によって引き起こされるオーバヘッドの影響を調べるために、この部分を理想化せず、長いキャッシュ参照遅延と、曖昧な依存関係による発行待機を実装している。図 1 にベースライン・モデルのブロック図を示す。

ベースライン・モデルは 1 命令実行幅のクラスタ 8 個がネットワークで接続された構成となっている。各クラスタには発行キュー、複製されたレジスタファイル、演算器、フォワードパスが備わっている。各命令はフェッチ、デコード、リネーミング等のフロントエンド処理の後、ステアリングロジックによりどのクラスタで実行するかが決定され、いずれかのクラスタの発行キューへディスパッチされる。通常、命令はそのオペランドが生成されるクラスタへステ

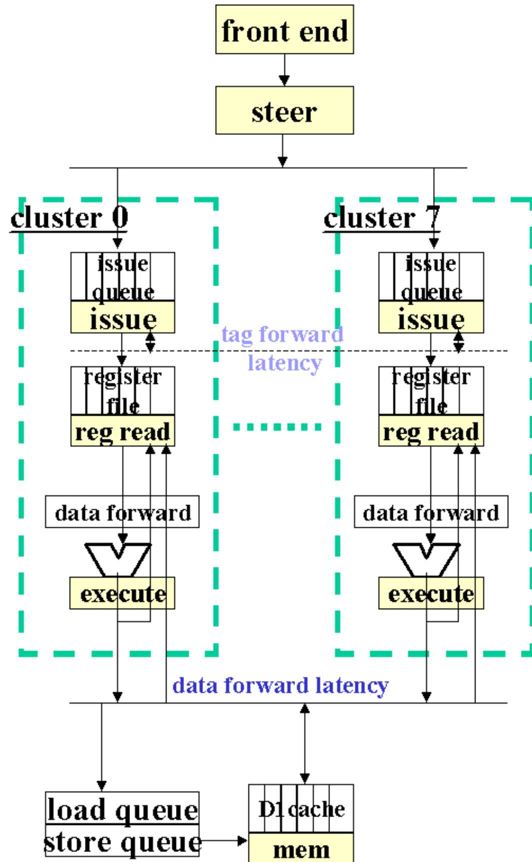

図 1: ベースライン・モデル

アされる。しかし、最低負荷クラスタと最高負荷クラスタの命令数の差が閾値を越えると、命令は強制的に最低負荷クラスタへ割り当てられる。

命令は条件が揃い次第、アウト・オブ・オーダに実行される。実行結果は全クラスタへブロードキャストされ、各クラスタのレジスタファイルの内容を更新する。クラスタ間の通信遅延のため、異なるクラスタでの実行結果の反映には数サイクルを要する。依存関係のある命令同士は、同じクラスタにステアされた時のみ連続したサイクルで実行することができる (2(a))。命令の wake up は図 2(b) のように、通信遅延やキャッシュ遅延を予測して行われる。

ベースライン・モデルは集中型の D1 キャッシュを備えており、全てのクラスタのメモリステート操作が反映される。

3.2 評価環境

表 3、4 に評価に用いたベースライン・アーキテクチャの設定を示す。

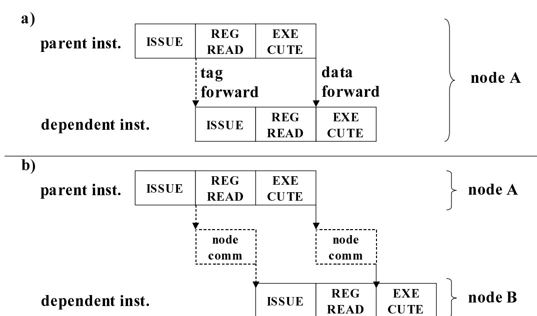

図 2: タグとデータのフォワーディング

fetch/retire width	up to16insts/cycle
I-cache	perfect
branch predictor	gshare 16k-entry,10bit history Unconditional control instructions predicted perfectly
memory disambiguation	16kentry wait table, 100k cycle refresh interval
cluster setting	8nodes
node components	64entry IQ, 256physical registers, 1ALU execute 1 inst/cycle
D-cache	64kB, 2-waySA, 8cycle hit latency 64byte lines 3 read ports 1read/write port

図 3: ベースライン・アーキテクチャ構成

ベースライン・モデルの命令セットは DEC Alpha21264 に準じ、シミュレータは実行トレースを入力とする。評価には SPEC95int(train) から compress、gcc、go、ijpeg、li、m88ksim、perl(jumble、primes、scrrable)、vortex の 10 種類を用い、先頭から最大 16M 命令について動作を計測した。

4 メモリ参照のオーバヘッド

クラスタ化スーパスカラ・プロセッサでは、積極的なクラスタ化により実行コアが非常に高速化される一方、キャッシュ等の集中型ユニットが高速化に追随できず、大きな遅延を持つと予想される。本研究では、以下に示すようなメモリ参照処理に関するオーバヘッドに着目する。

Total pipeline depth	16stages
frontend latency(fetch – dispatch)	11cycles
issue – issue latency	1cycle
execute latency(int)	1cycles
execute latency(int MULT)	15cycles
execute latency(float)	4cycles
inter cluster forwarding latency	2cycles

図 4: ベースライン・パイプライン設定

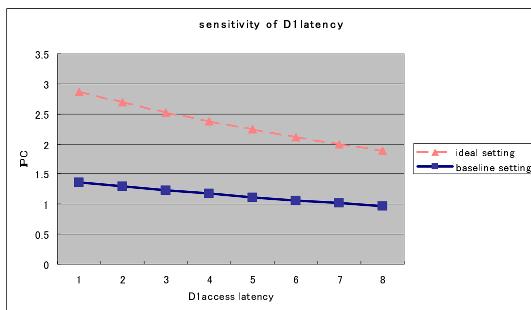

図 5: キャッシュ参照遅延増加の IPC への影響

4.0.1 キャッシュ参照遅延

ロード命令処理は通常の命令処理に加えてキャッシュ参照による追加の遅延があり、現行のプロセッサでもオーバヘッドとなっている。高クロック化が進むと更に追加の遅延は大きくなると予想され、性能に大きな影響を与える。

図 5 にキャッシュ参照遅延を 1 サイクルから 8 サイクルまで変化させたときの IPC の様子を示す。ideal setting は注目している要素以外のオーバヘッドを排した場合の値である。

ideal setting、baseline setting 共に大きな傾きのほぼ線形のグラフとなっており、メモリ参照遅延が実行のクリティカルパスに影響していることが分かる。双方の setting で、1 サイクルのキャッシュ参照遅延増加は約 5%の IPC 低下を引き起こしている。

4.0.2 曖昧な依存関係によるメモリ依存チェイン

メモリ参照は実行時まで依存関係が判明しないため予防的に全てのストア命令に対して依存関係を仮定しなければならない。メモリ依存チェインの発行タグの伝播には、各所でクラスタ間通信遅延が追加されてしまうため、曖昧なメモリ依存関係の解決は従来よりも長い遅延を伴う事が予想される。

図 6 に曖昧なメモリ依存関係の IPC への影響を示す。左側の perfect memory disambiguation の設定は、フロントエンド時点でメモリ依存関係が判明しているものとして、ロード命令が最速で発行する理想モデルである。右側の設定では、Wait Table 予測のみを用いてロード命令の先行発行を行っている。

曖昧な依存関係は、ideal setting で 18%、baseline setting で 39%の IPC 低下を引き起こしている。baseline setting における差が大きいため、この部分の改善により、大きな性能向上が期待できる。

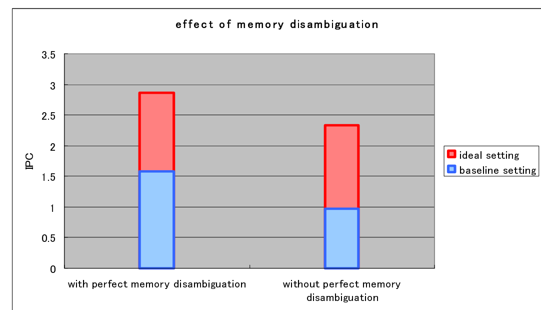

図 6: メモリ参照命令発行チェインによる影響

二つの解析結果より、メモリ参照命令によるオーバヘッドはクラスタ化によって更に増大しており、この部分の改善によってクラスタ化スーパスカラ・プロセッサの性能が大きく向上する余地があることが分かる。

5 分散投機メモリフォワーディング

本論文ではクラスタ化スーパスカラ・プロセッサに適したメモリ参照オーバヘッド軽減手法として、分散投機メモリフォワーディングを提案する。

実行コアのクラスタ化に追随できるスピードでメモリ参照を可能とするためには、メモリ参照処理もクラスタ化による高速化を利用しなければならない。しかし、全クラスタのステートを反映し、正確な値を供給しなければならないキャッシュ方式では、分散配置を高速化に結びつけることは難しい。

提案手法では、投機メモリフォワーディング手法の考え方を用い、ストア命令とロード命令のコミュニケーションに着目する。メモリ参照命令は、メモリ空間を用いた値の受け渡しであり、キャッシュから読み出されるデータの起源は、いずれかのクラスタで生成されたストア値である。そこで、各クラスタに “ローカルフォワードバッファ” と呼ばれる小容量バッファを配置し、ストア値を生成した時にこのバッファに保持する。メモリ依存関係にあるロード命令は、読み込むストア値を持っているクラスタへステアし、ローカルフォワードバッファから値を読み込む。このようにして、従来のキャッシュ階層をバイパスし、ローカルな参照とすることができる。ストアとロードのメモリ依存関係を高確率で予測する手法は提案されており、フォワーディング適用の

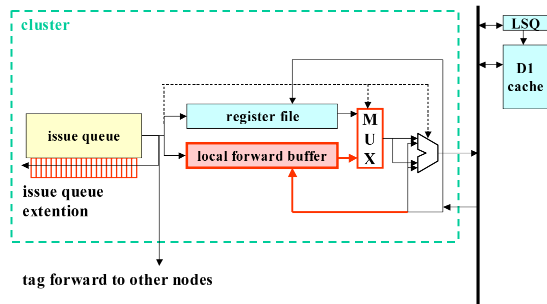

図 7: 追加ハードウェア

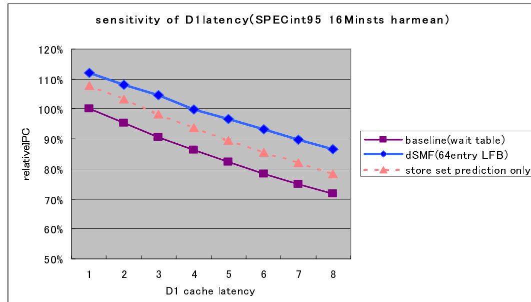

図 8: 分散投機メモリフォワーディングによる性能向上

判断やステアリングに利用できる。クラスタに保持するストア値は、依存予測可能なストア命令に限定することで、無駄なデータをクラスタ内に置くことなく、バッファ容量を有効に利用することができる。

各クラスタに追加するハードウェアを図 7 に赤色で示す。またフロントエンド、バックエンド部分にメモリ依存予測機構が追加される。

6 提案手法の評価

分散投機メモリフォワーディングをベースライン・モデルに実装し、シミュレータによる評価を行った。その結果、各クラスタに数エントリのローカルフォワードバッファを配置することにより、全ロード命令の 36%にフォワードが適用可能であることが分かった。図 8 に提案手法を導入したときの IPC 向上の様子を示す。キャッシュ参照遅延を 1〜8 サイクルの間で変化させ、それぞれの性能を計測した。クラスタ内にごく少量のバッファを追加することにより、キャッシュ参照遅延を一律に数サイクル短縮した場合と同じ効果が得られることが確認できる。提案手法はベースライン・モデルに対して 20%以上の性能向上を達成している。

ベースラインとして設定したパラメタでは、クラスタ化によって得られる実行並列幅を有効に活用できていなかった。提案手法によって、メモリ参照のオーバヘッドを軽減することにより、同じ遅延パラメタ設定でも、4〜8 クラスタ構成という比較的大きなプロセッサにおいて分散並列実行が有効となった。

7 おわりに

分散投機メモリフォワーディング手法を提案し、クラスタ化スーパスカラ・プロセッサの高速性を損ねることなく、数エントリのバッファを追加することで、キャッシュ参照遅延の影響の軽減できることを示した。今後キャッシュ参照遅延は増加することが予想され、このようなローカルバッファの利用は更に重要になっていくと考えられる。

参考文献

[1] V.Agarwal, M.S.Hrishikesh, S.W.Keckler, Doug Burger, "Clock Rate versus IPC:The End of the Road for Conventional Microarchitectures",27th Int.Symp.on Computer Atchitecture, pp.248-259, Jun 2000.

[2] S.Palacharla, N.P.Jouppi, J.E.Smith, "Complexity-Effective Superscalar Processors", 24th Int.Symp.on Computer Architecture, pp.1-13, Jun 1997.

[3] J.M.Parcerisa, J.Sahuquillo, A.Gonzalez, J.Duato, Efficient Interconnects for Clustered Microarchitectures, In Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques(PACT2002), pp291-390.

[4] R.Balasubramonian, S.Dwarkadas, D.H.Albonesi, "Dynamically Managing the Communication-Parallelism Tradeoff in Future Clustered Processors", 30th Int.Symp.on Computer Architecture, pp.275-286, Jun 2003.