2G-09

スレッドレベル並列投機実行のためのデータ依存解析機構
An Analyzing Mechanism of Data Dependencies for the Thread-Level Speculation

出島 貴史† 布目 淳‡ 平田 博章‡
Takashi Dejima Atsushi Nunome Hiroaki Hirata

1. はじめに

プログラムの実行時間は主にループで消費されることが知られている．このため，ループを並列に処理することで実行時間の短縮が期待できる．我々は，ループの各イタレーションをスレッドとして並列かつ投機的に実行するシステムを開発中であり，本稿では，スレッド間で共有するメモリ上のデータについて，依存解析を行う専用ハードウェア機構を提案する．

2. データ依存解析機構

2.1 システム概要

ループの$i-1$番目までのイタレーションの実行が完了しているものとする．このとき，i番目のイタレーションを実行するスレッドと並列に，$i+1$番目以降のイタレーションを投機的に実行するスレッドを**投機スレッド**と呼ぶことにする．図1の各ノードの1次キャッシュに，スヌープキャッシュプロトコルを利用してスレッド間の依存解析を行うための専用機構を設ける．ハザードを検出した場合には，それに関連する投機スレッドのみをアボートして，それらのスレッドを最初から再実行する．

スレッドの実行中は，一般に，スレッド間の依存関係の原因となるデータアクセスを多数回行うが，その多くは書き込みで始まることがわかっている[1]．そこで，我々は，各プロセッサの1次キャッシュに，スレッドごとのデータの別バージョンを格納することで，出力依存と逆依存の関係を除去する**メモリリネーミング機構**[2]を提案している．キャッシュ間をリング状に接続し，依存解析を行うための制御信号を1サイクルで各ノードに伝播する．しかし，この構成では，ノード数を増やすとサイクル時間が長くなってしまう．また，リング状に接続された順に，各ノードにスレッドを割り付けなければならないという制約がある．

そこで，本稿では，リング状の制御線を用いることなく，(i) ノード数に依存しない固定のサイクル時間で，依存解析に必要な制御信号を全てのノードに伝播するとともに，(ii) 任意の空ノードに新たなスレッドを割り付けることを可能とするシステム構成を提案する(図2)．イタレーションの前後関係を表す**イタレーションID(IID; Iteration ID)**を，そのイタレーションを実行するスレッドに割り当てる．各ノードに**イタレーション制御ユニット(ICU; Iteration Control Unit)**を設け，IIDを用いてICUがスレッド間の依存解析を制御する．

2.2 識別子制御ユニット(ICU)

2.2.1 イタレーション ID(IID)

ICU内に，そのノードで実行するスレッドに割り当てられたIIDを記憶する**IIDレジスタ**と，空ノードに次に割り当てるIIDを記憶する**NIIDレジスタ**を設ける．IIDを通し番号(整数値)で表現する方法[3]が提案されているが，スレッド間の前後関係を判定するための算術演算が必要となる．そこで，本稿では，IIDをビットベクタで表現し，スレッド間の前後関係を論理演算のみで判定することによって高速化を図る．全ノード数がnのとき，IIDレジスタはnビットのレジスタで，また，NIIDレジスタは$n+1$ビットのシフトレジスタで，それぞれ構成する．ループの投機実行を始める前に，NIIDレジスタは，最上位ビットは1に，他のビットは0に，それぞれ初期化する．

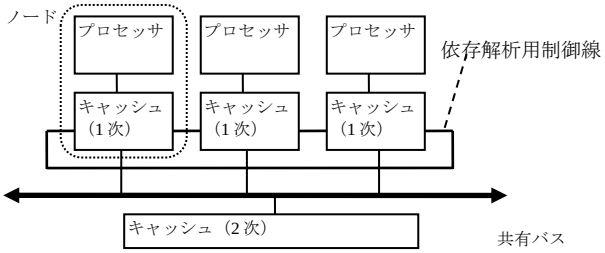

図1. 現在のシステム構成

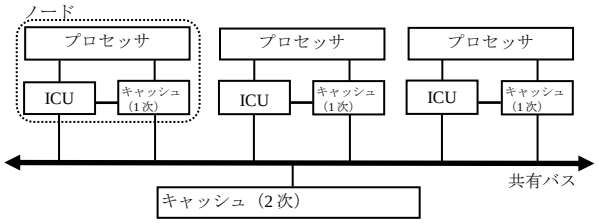

図2. 改良後のシステム構成

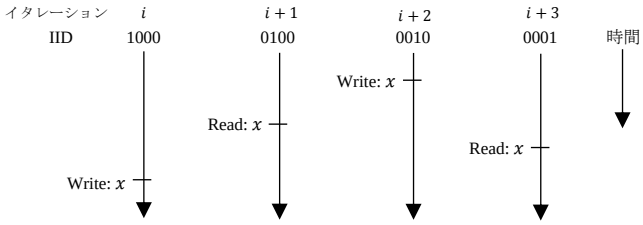

図3. 投機実行におけるイタレーションID

図3に，4個のスレッドを並列実行している場合のIIDを示す．スレッド割り付け時に，ICUはNIIDレジスタの上位nビットをIIDレジスタへコピーし，NIIDレジスタの内容を右に1ビットシフトする．また，これと同時に共有バスに割り当て通知を出力する．一方，投機スレッドをコミットするとき，NIIDレジスタの内容を左へ1ビットシフトすると同時に，共有バスに完了通知を出力する．バス上の割り当て通知や完了通知をスヌープした他のノードのICUは，各自のNIIDレジスタを右，左にそれぞれ1ビットシフトする．

2.2.2 スレッド間の依存解析

あるスレッドがメモリへの書き込みを行うとき，その

†京都工芸繊維大学工芸科学部情報工学課程
‡京都工芸繊維大学情報工学・人間科学系
Kyoto Institute of Technology

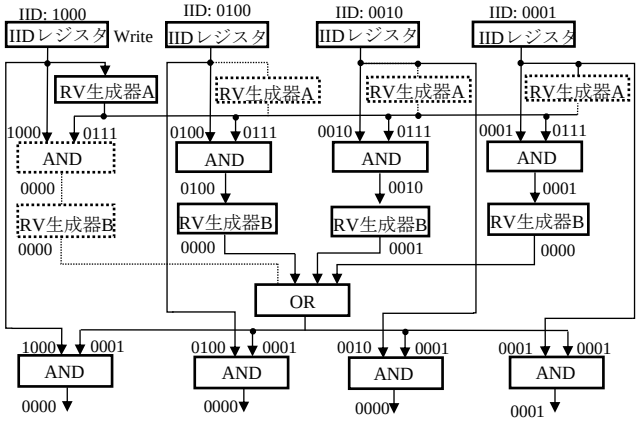

図4. ICUにおける依存解析機構

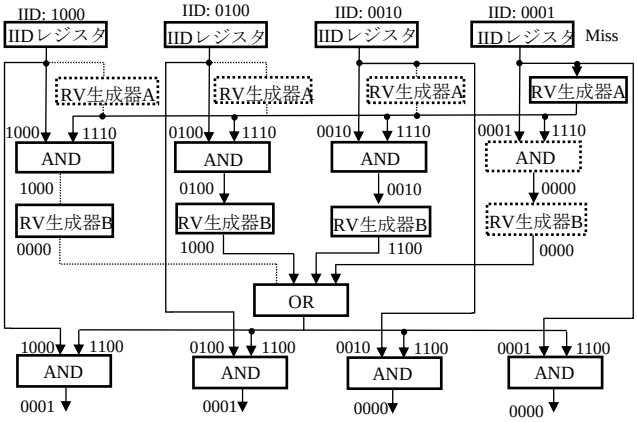

図5. ICUにおける最新データロード制御機構

スレッドに後続する全てのスレッドは，書き換えられる前の値を読み込んでいないか検査する必要がある．そこで，1次キャッシュのライン毎に，ライン中の各バイト領域に書き込みを行ったことを示す**W(Written)ビット**と，ライン中のある領域に対して，そのスレッドにおける最初のアクセスが読み出しであったことを示す**FR(First Read)ビット**を設ける．

図3の例では，IIDが1000のスレッドが変数xに書き込みを行う前に，IIDが0100と0001の後続スレッドが先に変数xの読み出しを行っている．そのため，単純には，これら2個のスレッドをアボートする必要がある．しかし，IIDが0001のスレッドについては，IIDが0010の先行スレッドが変数xに書き込んだ後の値を読み出していたのであれば，アボートする必要はない．

図3において，IIDが1000のスレッドが書き込みを行う際の依存解析の処理を図4に示す．上位kビット目が1であるIIDのスレッドがメモリへ書き込みを行うとき，依存解析を行う必要のある後続スレッドを指定するために，上位$k+1$ビット以降がすべて1である**応答要求ベクタ(Responsibility Vector; RV)**を共有バスに出力する(RV生成器A)．他のノードでは，自分のIIDとバス上のRVとの(ビット毎の)論理積が0000でなければ依存解析を行う．ただし，IIDが0001のスレッドのように，アボートする必要のないスレッドも後続スレッドの中に存在し得る．そこで，論理積の結果が0000ではなく，かつ，書き込みアドレスに対するキャッシュラインのWビットがセットされていて，同時に，FRビットがセットされていない場合は，RV生成器Bにおいて，さらにRVを求めて出力する(ただし，上記の条件を満たさない場合は0000を出力する)．これらの論理和が，アボートする必要のないスレッドの集合を表し，図4ではそれが0001と示されている．自スレッドのIIDがその集合に含まれているかどうかは論理積を用いて判定できるので，その結果が0000とならなければ，そのスレッドについては依存解析を行わない．それ以外のノードは依存解析の対象となり，キャッシュにヒットして，そのラインのFRビットがセットされていれば，スレッドをアボートする．

2.2.3 最新データロード機構

あるスレッドのメモリ読み出しにおいてキャッシュミスが発生した際には，そのスレッドにとって最新バージョンのデータをロードする必要がある．図3の例において，IIDが0001のスレッドを実行するノードでキャッシュミスが発生した際の処理を図5に示す．なお，IIDが0100と0001のスレッドを実行するノードのキャッシュに，キャッシュミスを発生したアドレスのデータが存在するものと仮定する．

キャッシュミスを発生したノードのICUは，RV生成器AにおいてRVを求め，これを共有バスに出力する．ここでは，ロード要求に応答する必要がある先行スレッドを示すために，上位kビット目のみが1であるビットベクタに対して，上位$k-1$ビット目までのすべてのビットが1のビットベクタをRVとして用いる．他のノードについては，自分のIIDレジスタの内容と共有バス上のRVとの論理積が0000でなく，かつ，キャッシュにヒットする場合は，ロード要求への応答責任がある．ただし，応答責任を有するノードが複数個存在する場合がある．そこで，このようなノードの中で最も後のイタレーションを実行しているノードを選択するために，各ノードのICUは，応答を抑止するためのRVを出力する(RV生成器B)．これらの論理和が，応答を抑止すべきスレッドの集合を表し，図4ではそれが1100と示されている．これとIIDレジスタの内容との論理積が0000のノードが，実際にロード要求に応答する候補となる．

3. むすび

本稿では，並列投機実行時のデータ依存解析において，ノード数に制限を設けることなく，依存解析に必要な制御信号を1サイクルで伝播する機構を提案した．今後は，フロー依存の影響を最小化するための方策を検討し，システム全体の性能評価を行う予定である．

謝辞

本研究の一部は日本学術振興会科学研究費補助金（基盤研究（C）15K00076，同15K00169）の補助による．

参考文献

[1] 森田清隆, 布目淳, 平田博章, 柴山潔, “スレッドレベル並列化のためのスレッド間依存関係の分類,” 第 9 回情報科学技術フォーラム論文集, vol. 1, pp. 81-86 (2010).

[2] 平田博章，藤井崇弘，藤皓平，森田清隆，布目淳，柴山潔，“スレッドレベル並列投機実行のためのメモリリネーミング機構,” 第11回情報科学技術フォーラム論文集，Vol. 1, pp. 31-34 (2012).

[3] J. Steffan, C. Colohan, A. Zhai, T.C. Mowry, “The STAMPede Approach to Thread-Level Speculation,” ACM Transactions on Computer Systems, Vol. 23, No. 3, pp. 253-300 (2005).