6W-6

並列処理システム－晴－における構造体処理の実現

山名早人，　草野義博，　村岡洋一
（早稲田大学　理工学部）

1．はじめに

本稿では，並列処理システム－晴－〔1〕における構造体処理方式〔2〕の実現方法について述べる．－晴－では，実行方式にＣＤフロー(Controlled Dataflow) 方式〔3〕を採用している．ＣＤフロー方式では，マクロブロックと呼ぶ処理単位間でコントロールフロー制御をおこない，マクロブロック内でデータフロー実行をおこなう．

データフロー実行には記憶の概念が存在しないが，実際に計算機を構成するにあたっては，大規模な構造体を格納するための構造体記憶が必要不可欠である．従来，構造体処理に関してI-ストラクチャ〔4〕等が提案されている．しかし，これらの方式はデータフロー方式の持つ単一代入則を厳密に実現したものであって，参照は複数回できるが，定義は１回のみという制限を持つ．したがって，二重定義時には，構造体をコピーしなければならず，オーバヘッドが発生する．－晴－では，構造体記憶（以下大域記憶と呼ぶ）に対して複数回の定義及び参照を可能とし，二重定義時のコピーオーバヘッドを無くした構造体処理方式を提案している〔2〕．本方式では，アクセス順序の保証を複数マクロブロックに及ぶ定義・参照に対しておこない，マクロブロック内に閉じた定義・参照は対象としない．これは，マクロブロック内で定義されたデータを同一マクロブロック内で使用する場合には，定義されたデータを参照ノードに直接送るからである．

本稿では，まず－晴－の大域記憶及び構造体管理方式について述べ，－晴－における具体的な実現方法を詳述する．

2．－晴－の大域記憶

－晴－の大域記憶の構成を図１に示す．大域記憶は，複数の要素プロセッサ(PE)からアクセスされるため，アクセス競合が問題となる．そこで，このアクセス競合を回避する解決法の１つとしてマルチリードメモリ方式を採用する．図１に示すように，同一の内容をもったメモリモジュールを複数持ち，読み出しのアクセス競合を無くす．書き込み時には，全メモリモジュールに対して書き込みをおこない，メモリモジュール間でのデータの同一性を保証をする．しかし，書き込み時にはメモリモジュール間でアクセス競合が発生する．この競合を軽減する為にメモリモジュールをバンク化し，異なったバンク間での並列書き込みを可能とする．詳細は，現在検討中であり，別途報告する．

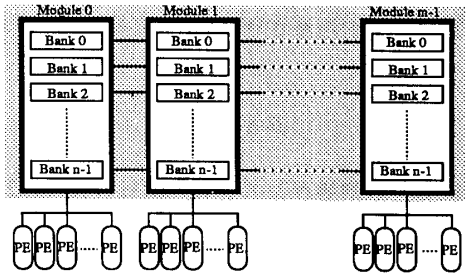

図１　－晴－の大域記憶構成

A Structure Handling Scheme
of Parallel Processing System -Harray-
Hayato Yamana, Yoshihiro Kusano, Yoichi Muraoka
Waseda University

3．構造体管理方式〔2〕

－晴－の構造体処理方式として，複数回定義・参照が可能な処理方式をこれまでに提案している．この方式を用いることによって，I-ストラクチャ等で問題となる構造体二重定義時のコピーに起因するオーバヘッドを無くすことができる．本方式では，アクセス順序保証のレベルとして，配列単位での保証方法を採用している．

定義後の参照及び参照後の定義を保証する方法を図２に示す．定義後の参照保証は，各配列について定義世代という概念を導入し，その世代において定義が終了したことを，ハードウェアで確認後，参照を開始することにより実現する．定義世代とは，定義後の参照を保証するための境界であり，世代単位で定義と参照の順序を保証する．この世代は参照が終了して再び定義が開始される時点でハードウェアにより更新される．

次に参照後定義の保証について述べる．参照後定義を保証するためには，その配列に対する参照が終了したことを確認後，定義を開始すればよい．参照が終了したことを検知する方法として，－晴－では二つの方法を用意する．一つは，参照終了をマクロブロックの終了として検出する方法であり，もう一つはソフトウェア上で参照終了したことを検知する方式である．前者の方法は，マクロブロックの終了と参照終了がほぼ同時刻となる場合に利用し，後者の方法は，マクロブロックの終了と参照終了の時刻が大きく異なる場合に用いる．これらの方法により，参照終了したマクロブロック数を数え，各配列を参照する全てのマクロブロックで参照が終了したことを確認後，再定義を開始させる．これによって，参照後の定義を保証する．

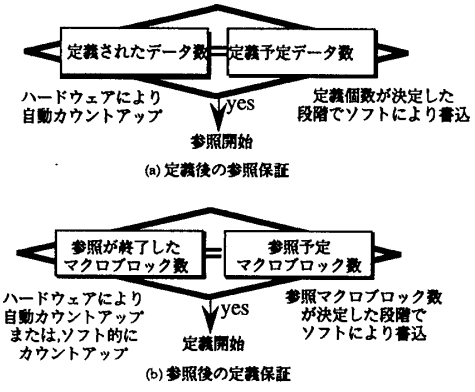

図２　参照後定義・定義後参照の保証方法
（各配列，同一世代についておこなう）

－晴－では，前節で述べたように複数のメモリモジュールを持つため，上述の機能を実現するアクセス順序保証テーブル〔2〕を各メモリモジュールに持ち，メモリモ ジュール単位で定義及び参照の保証をする．これによって，メモリモジュール間のデータ転送をPEからの書き込みと非同期におこなうことができる．したがって，メモリモジュール間の通信競合を軽減することができる．

4．具体的実現方法

本節では，これまでに述べてきた構造体処理方式をどのように実現するかについて述べる．

4．1　構造体処理命令

以下に構造体処理をおこなうための命令群を示す．命令群は(1)アクセス順序保証に用いる命令と，(2)その他の命令（主にデバッグに使用）に分類される．

(1)アクセス順序保証に用いる命令群

・アクセス順序保証命令
　参照可能／定義可能かどうかを調べる命令．
SRR　(check Structured data Ready for Read)
SRW　(check Structured data Ready for Write)
・定義・参照回数設定命令
　定義されるデータ数あるいは参照回数を定義する命令
SNS (Set the Number of written/read Surutured data)
・参照回数インクリメント命令
　参照終了したことを報告する命令．マクロブロック終了を参照終了としない場合に使用する．
INCS (INCrement the number of read Structured data)

(2)その他の命令群

・アクセス順序保証取消命令
　SRR/SRW命令を取り消すための命令．
CSRR (Cancel SRR)　　CSRW (Cancel SRW)
・参照・定義強制命令
　アクセス順序保証テーブル状態設定命令．
SSRR (Set Structured data Ready for Read)
SSRW (Set Structured data Ready for Write)
・状態把握命令
　アクセス順序保証テーブルの状態を読む命令
RST　(Read Structure Table)

4．2　配列名の判断

本方式では，大域記憶への書き込み時に，各配列に対応するアクセス順序保証テーブルの値を更新する．したがって，大域記憶へデータを書き込む際，そのデータの所属配列を示す情報が必要となる．解決策として，大域記憶への書き込み時に配列情報を付加する方法が考えられるが，PEと大域記憶間のバンド幅が大きくなるので好ましくない．このため，書き込みアドレスから直接に配列情報を得る方法をとる．

具体的には，大域記憶において仮想アドレス→実アドレスへの変換をする際に，ページ番号から配列情報（配列識別番号）を同時に求める（図３）．この方法は，配列の単位（正確にはアクセス順序を保証する最小単位）がページの倍数になるという欠点を持つが，PEと大域記憶間のバンド幅を小さくできるので，PE数が多い並列処理計算機では有効である．

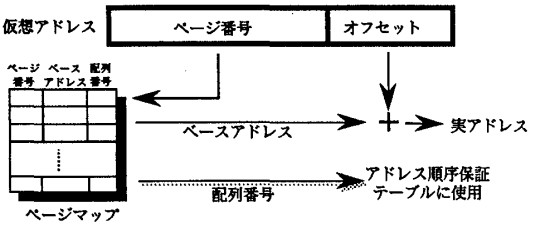

図３　アドレスから配列名を得る方法

4．3　配列の世代

配列の世代とは配列の定義された世代を識別するものである．配列の世代は，一世代の参照が終了し，再定義が始まった段階で自動的にカウントアップされる．この世代には4bit程度を用意する予定であり，世代がオーバフローする場合には 0世代に戻すものとする．このように，複数の世代を用意することによって，複数世代にわたる定義・参照要求を同時に受け付けることが可能となる．しかしながら，－晴－では上位レベルでコントロールフロー制御をおこなっているため，世代の開きを制御することが可能である．したがって，実際には同時に数世代の要求を処理できればよい．

また，世代情報は，配列の定義マクロブロックから参照マクロブロックへ伝達させ，現在の処理対象世代を各マクロブロックにおいて把握する．

図４に，アクセス順序を保証するための手順を示す．図４に示すように，まず，参照・定義が終了しているかどうかを確認し，確認後，定義・参照を開始する．

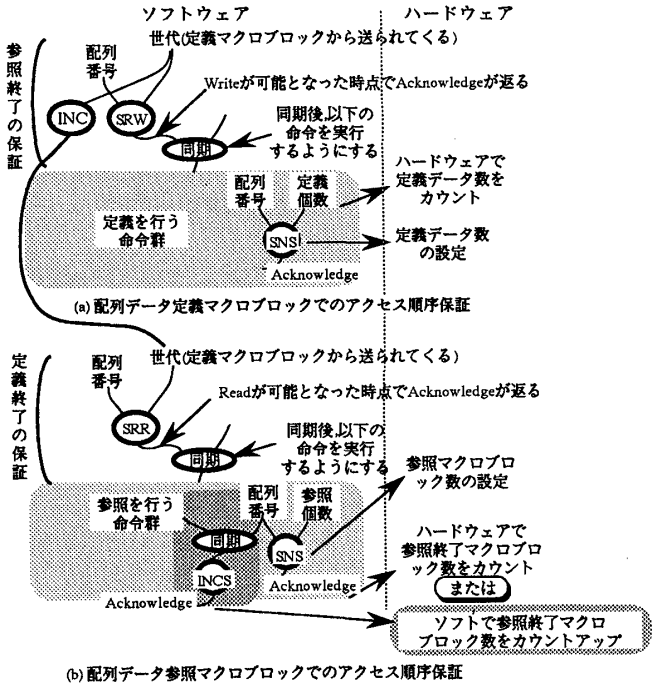

図４　アクセス順序保証の手順

5．おわりに

－晴－における構造体処理方式の実現方法について述べた．今後は，構造体を記憶する大域記憶を複数のPEから高速にアクセスする方法について具体的に検討を進める．

参考文献

(1)H.Yamana,et al:"System Architecture of Parallel Processing System -Harray-," Proc. of Int. Conf. on Supercomputing, pp.76-89 (1988)

(2)山名他："並列処理システム－晴－における構造体管理方式",信学技報,CPSY-89-3,pp.17-24 (1989)

(3)山名他："並列処理システム－晴－におけるＣＤフロー(Controlled Dataflow)方式",信学技報,(並列処理にに関する指宿ﾐﾆｼﾝﾎﾟｼﾞｳﾑ投稿中) (1989.8)

(4)Arvind:"A Critique of Multiprocessing von Neuman Style," Symp. on Computer Architecture, pp.426-436 (1983)