TR-322

KL1のメモリ参照特性に適した並列キャッシュ機構

松本　明、中川貴之、佐藤正俊、木村康則
西田健次、後藤厚宏

November, 1987

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

KL1のメモリ参照特性に適した並列キャッシュ機構

Parallel cache mechanism optimized for KL1 memory access characteristics

松本 明、中川 貴之、佐藤 正俊、木村 康則、西田 健次、後藤 厚宏

Akira MATSUMOTO, Takayuki NAKAGAWA, Masatoshi SATO,
Yasunori KIMURA, Kenji NISHIDA, Atsuhiro GOTO

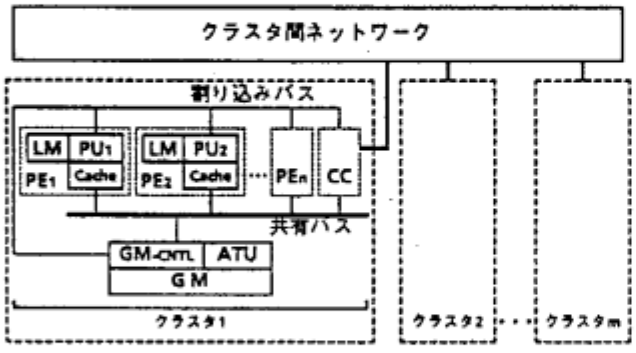
(財)新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology (ICOT)

1．はじめに

ＩＣＯＴでは，図１に示すようなクラスタを用いた階層構造の並列推論マシンＰＩＭの研究開発を進めている[1]。ＰＩＭの対象言語である並列型核言語ＫＬ１のクラスタ内処理系は，共有メモリ型の密結合マルチプロセッサでの実行を前提に設計している[2]。そこで，クラスタには，共有メモリへのアクセスを高速化するためにプロセッサ毎にキャッシュ・メモリを設けた[3]。

ＰＩＭの設計と評価は，初めにクラスタ内ＫＬ１処理系のシミュレータから全てのメモリ参照情報の履歴を取り出す。次に，それを並列キャッシュ・シミュレータへ入力して，ＫＬ１処理におけるメモリ参照特性を種々の尺度から評価する。この評価結果を処理系の設計，および，ハードウェアの設計にフィードバックをかけながら研究開発を進めている[4]。

本稿では，初めに，ＰＩＭのクラスタ用に設計中の並列キャッシュ機構と，共有メモリのロック機構について述べる。さらに，ヒープ使い捨て型のクラスタ内ＫＬ１処理系と並列キャッシュのシミュレーションによって収集したメモリ参照特性について報告する。



PE: 要素プロセッサ　　CC: クラスタ･コントローラ
PU: プロセッサ･ユニット　　ATU: アドレス変換ユニット
LM: ローカル･メモリ　　GM-CNTL: GMコントローラ
GM: 共有メモリ

図1. PIMのハードウェア構成

2．並列キャッシュ機構

2．1．並列キャッシュ機構設計のポイント

共有メモリの参照を高速化するために導入したプロセッサ毎のキャッシュ・メモリ（以後，並列キャッシュと呼ぶ）では，ヒット率の向上よりも，むしろ共有バス使用頻度の低減が大きな設計ポイントである。ＫＬ１処理では，書き込みの比率が比較的高いため，ストアスルー方式より共有バス使用頻度を低く抑えることのできるライトバック方式を採用した。ライトバック方式は，表１に示すような２種類の分類ができる。

第一の分類は，共有キャッシュ・ブロックへの書き込み時に，他のプロセッサのキャッシュ・ブロックを無効化する方式と，他のプロセッサのキャッシュ・ブロックも同時に書き換えるブロードキャスト・ライト方式である。無効化する方式は，共有キャッシュ・ブロックへの書き込み頻度が少ない場合に，共有バスの使用頻度が低い。一方，ブロードキャスト・ライト方式は，複数のプロセッサが共有キャッシュ・ブロックへ何回も書き込む場合に有利である。

第二の分類は，変更の有るキャッシュ・ブロックをキャッシュ間転送する時に，共有メモリに書き戻さない方式と，共有メモリに書き戻す方式である。共有メモリに書き戻さない方式は，共有メモリのビジー率を低く抑えることができる。共有バスの使用率は，書き戻す方式よりも若干増えるが，書き込みデータの読み出しによる共有率が低ければ軽微である。

ＰＩＭでは，以上の点を考慮した結果，共有ブロックへの書き込み時に他のプロセッサのキャッシュを無効化し，変更の有るブロックのキャッシュ間転送時に共有メモリへ書き戻さない方式を採用することにした。この方式を，キャッシュ・ディレクトリの状態数から５状態並列キャッシュ・モデルと呼ぶことにする。

なお，ＢＥＲＫＥＬＥＹ[5]では，キャッシュ間転送の供給元は，オーナー権の有る場合だけに制限してい

457

表1．ライトバック方式の比較

共有ブロックへの書き込み時の処理	他プロセッサのキャッシュを無効化する	ブロードキャスト・ブライトする
変更の有るブロックのキャッシュ間転送時に共有メモリに書き戻さない	BERKELEY[5] 5状態モデル	DRAGON[7]
変更の有るブロックのキャッシュ間転送時に共有メモリに書き戻す	ILLINOIS[6]	FIREFRY[7]

るのに対して，5状態モデルでは，いかなる状態でもキャッシュ間転送の供給元となれる。

2．2．5状態並列キャッシュ

逐次型のライトバック方式では，(ⅰ)Valid/Invalid，(ⅱ)Clean/Modifiedの状態フィールドが必要である。5状態並列キャッシュ・モデルでは，複数プロセッサのキャッシュ間で一貫性を保ち，さらに，できるだけ共有バスの使用頻度を下げるために，(ⅲ)Exclusive/Sharedを加えた3つの状態フィールドで，次の5状態を識別することにした。

①EM：Valid，Exclusive，Modified
　共有されていないことが明らかな状態で，変更の有る状態。

②EC：Valid，Exclusive，Clean
　共有されていないことが明らかな状態で，変更の無い状態。

③SM：Valid，Shared，Modified
　共有されている可能性のある状態で，変更の有る状態。

④S　：Valid，Shared
　共有されている可能性のある状態。

⑤I　：Invalid
　無効（未使用ブロックを含む）状態。

（備考）
複数プロセッサが，同一アドレスのキャッシュ・ブロックのコピーを持つ場合，SM状態が1個に対して，S状態が1個以上の場合と，複数のS状態からだけなる場合が有る。ExclusiveであるEM（およびEC）状態のキャッシュ・ブロックは，決して共有されることはない。

2．3．メモリ操作用のCPUコマンド

①R　：Read
　データの読み出し。

②W　：Write
　データの書き込み。

③DW：Direct Write
　通常のWコマンドがミスヒットした場合，書き込みに先立ってキャッシュ・ブロックをフェッチし，かつ他のプロセッサ上のキャッシュを全て無効化する必要がある。これに対して，DWコマンドでは，共有メモリからスワップインするオーバヘッド無しに，直接自プロセッサのキャッシュに書き込む。また，他プロセッサのキャッシュを無効化しないので，共有バスの使用頻度を大きく低減する効果のあるメモリ操作コマンドである。

DWコマンドは，自プロセッサを含めた全プロセッサのキャッシュ上にエントリのないことが明らかなブロックで，かつ，そのキャッシュ・ブロックの先頭番地にだけ使用可能である。キャッシュ・ブロックの先頭番地以外に出されたDWコマンドは，ハードウェアが通常のWコマンドに置き換えるものとする。これは，キャッシュ・ブロックがスワップアウトされたり，若しくは，キャッシュ間転送によって，他のプロセッサと共有されても，一貫性を保証するためである。

以上述べたように，DWコマンドの使用に際しては，制限がある。従って，KL1処理系で適用できるのは，ヒープ領域を新たに確保する場合，および，次に述べるRB（およびRP）コマンドと組み合わせてゴールレコード領域や通信バッファ領域へ書き込む場合等の用途に限られる。

④RB：Read Buffer
　自プロセッサや他プロセッサのキャッシュ・ブロックが，不要となった時点で，速やかに無効化する。これにより，不要ブロックがスワップアウトやスワップインされることを抑止して，共有バスの使用頻度を下げるメモリ操作コマンドである。

RBコマンドは，共有メモリを介したプロセッサ間通信で受信側プロセッサが送信側プロセッサのデータを実際に読み出す場合，および，一つのプロセッサ内だけでローカルに使用されるデータ領域を読み出す場合に使用できる。

RBコマンドは，読み出す番地のキャッシュ・ブロック内相対位置によって，次に述べる二通りの動作をする。

(ⅰ)　ブロック内の最後以外のワード
　ミスヒットし，かつ，送信側プロセッサからのキャッシュ間転送を行った場合に，キャッシュ間

転送と同時に送信側プロセッサのキャッシュ・ブロックを強制的に無効化する（ＲＩ：Read Invalidateと呼ぶ）。

（ii） ブロック内の最後のワード

自プロセッサのキャッシュ・ブロック内の最後のワードを読み出し終わった時点で，そのブロックを強制的に無効化する（次に述べるＲＰ：Read Purgeと同じ）。

⑤ＲＰ：Read Purge

キャッシュ・ブロックから読み出し後に，そのブロックを強制的に無効化するメモリ操作コマンドである。これにより，もはや不要となったブロックがスワップアウトされることを抑止して，共有バスの使用頻度を下げることができる。読み出しミスヒットの場合には，キャッシュ・ブロックのフェッチ時に，他プロセッサのキャッシュを全て無効化し，さらに，自プロセッサの読み出し後に，自プロセッサのキャッシュをも無効化する。

ＲＰコマンドは，ＲＢコマンドで自プロセッサのキャッシュ・ブロックを無効化できない場合，つまり，読み出す領域の長さがキャッシュ・ブロック長の整数倍でない場合に，最後のデータを読み出すために使用する。

２．４．メモリ操作用のバスコマンドと応答

並列キャッシュ・コントローラは，自プロセッサからの要求に応じるのと同時に，他プロセッサが発するバスコマンドにも応答しなければならない。

①Ｆ　：Fetch

該当キャッシュ・ブロックのフェッチを要求するバスコマンドである。

②ＦＩ：Fetch and Invalidate

該当キャッシュ・ブロックのフェッチ要求と共に，キャッシュ間転送の送信側プロセッサを含め，他の全てのプロセッサの該当ブロックの無効化要求をするバスコマンドである。

③Ｉ　：Invalidate

他の全てのプロセッサの該当キャッシュ・ブロックの無効化要求をするバスコマンドである。

④Ｈ：Hit

Ｆ，ＦＩ要求に対する応答で，自プロセッサのキャッシュにヒットしたことを示す。

２．５．５状態並列キャッシュの状態遷移

表２と表３に，それぞれＣＰＵコマンド，バスコマンドに対する５状態並列キャッシュ・モデルの状態遷移表を示す。

３．ロック機構

３．１．ロック機構設計のポイント

ＫＬ１処理系において，共有する実行環境への１ワード・ロックは，頻度が高いのでハードウェア・ロックを用いて高速化する。また，変数同士の単一化を行うために，最低２箇所の同時ロック機構を設けた方が良いと考える。

ロック操作に必要な，共有バスの使用頻度を下げるために，（ｉ）並列キャッシュのＥＭ，ＥＣ状態，（ii）ロック・ディレクトリのＬＷ状態（後述）を利用した最適化が可能である。

３．２．ロックディレクトリの状態

ロックディレクトリは，同時にロック可能な数分のエントリが必要で，次の状態を表わす。

①Ｌ　：Lock without Waiter

自プロセッサがロックしている状態。

②ＬＷ：Lock with Waiter

自プロセッサがロックしていて，かつ，この番地を含むキャッシュ・ブロックを参照待ちしている他プロセッサが存在する状態。Ｌ状態の時に，他プロセッサが，該当ブロックを参照するバスコマンドを発した場合に，ＬＷ状態へ遷移する。

③Ｅ　：NotLocked(Empty)

ロックしていない（未使用エントリ）状態。

３．３．ロック操作用のＣＰＵコマンド

次のロック操作用ＣＰＵコマンドを用意した。

①ＬＲ：Lock and Read

ロックをかけた読み出し。

②Ｕ　：Unlock

アンロック。

③ＵＷ：Write and Unlock

書き込み後にアンロック。

３．４．ロック操作用のバスコマンドと応答

次にロック操作用のバスコマンドと応答を示す。

①ＬＫ：Lock

ロック要求をするバスコマンドである。自プロセッサがＥＭ（またはＥＣ）状態，つまり，他プロセッサにキャッシュ・ブロックの写しが無いことが明らかな場合には，ＬＫバスコマンドを共有バスに出さない最適化を行う。

②ＵＬ：Unlock

ロックしていた番地をアンロックしたことを示すバスコマンドである。ＬＷ状態で無い場合，つまり，他プロセッサが，その番地を参照要求していない場合には，ＵＬバスコマンドを共有バスに出さない最適化を行う。

表２．５状態並列キャッシュの状態遷移表（ＣＰＵコマンド）

状態＼CpuCmd	ＥＭ	ＥＣ	ＳＭ	Ｓ	Ｉ	Remark
Ｒ	-／EM	-／EC	-／SM	-／S	SO,F(a)／ H(d)→S GM(d)→EC	Read Other Cache Hit Swapin from GM
Ｗ	-／EM	-／EM	I(a)／EM	I(a)／EM	SO,FI(a)／ H(d)→EM GM(d)→EM	Write Other Cache Hit Swapin from GM
ＤＷ	MCHK	MCHK	MCHK	MCHK	SO／EM	Direct Write
ＲＩ	-／EM	-／EC	MCHK	MCHK	SO,FI(a)／ H(d)→EM GM(d)→EC	Read Invalidate Other Cache Hit Swapin from GM
ＲＰ	-／I	-／I	MCHK	MCHK	SO,FI(a)／I	Read Purge
ＬＲ	-／EM	-／EC	LK,I(a)／ EM	LK,I(a)／ EM	SO,LK,FI(a)／ H(d)→EM GM(d)→EC	Lock Read Other Cache Hit Swapin from GM
ＵＷ	(UL)／EM	(UL)／EM	MCHK	MCHK	SO,(UL),FI(a)／ EM	Unlock Write Swapin from GM
Ｕ	(UL)／EM	(UL)／EM	MCHK	MCHK	(UL)／I	Unlock

（略号の説明）
XXX／YYY: XXXバスコマンドを発して，YYY状態に遷移することを示す。
F(a),FI(a),I(a): アドレスと共に，Ｆ，ＦＩ，Ｉの各バスコマンドを送出することを示す。
H(d)→YYY:　他プロセッサのキャッシュにヒットしたので，キャッシュ間転送でブロック・データを受け取り，YYY状態に遷移することを示す。
GM(d)→YYY: 共有メモリからのスワップインを行い，YYY状態に遷移することを示す。
MCHK: マシンチェックを示す。
SO:　スワップアウトが生じる可能性があることを示す。
(UL): ＬＷ状態ならば，ＵＬコマンドを出すことを示す。

表３．５状態並列キャッシュの状態遷移表（バスコマンド）

状態＼BusCmd	ＥＭ	ＥＣ	ＳＭ	Ｓ	Ｉ	Remark
Ｆ	H(d)／SM	H(d)／S	H(d)／SM	H(d)／S	-／I	Fetch
ＦＩ	H(d)／I	H(d)／I	H(d)／I	H(d)／I	-／I	Fetch Invalidate
Ｉ	-／I	-／I	-／I	-／I	-／I	Invalidate

（略号の説明）
XXX／YYY: XXXバスコマンドを受け取り，YYY状態に遷移することを示す。
H(d): 自プロセッサにヒットしたので，キャッシュ間転送でブロック・データを送ることを示す。

⑬ＬＨ：Lock Hit

Ｆ，ＦＩ，または，ＬＫバスコマンドに対する応答で，自プロセッサがロック中の番地を含むブロックへの参照を示す。ＬＨ応答を発したプロセッサは，Ｌ状態からＬＷ状態に遷移する。ＬＨ応答を受けた要求元プロセッサは，メモリ参照をビジー・ウェイトする。その後，ＵＬバスコマンドを受け取った時点で，メモリ参照をリトライする。なお，ビジー・ウェイトしている間は，共有バスを全く使用しない。

4．評　価

4．1．評価方法

評価手順は，初めに，ＫＬ１のクラスタ内処理シミュレータが，ベンチマーク・プログラムを実行する際にメモリ参照情報を全てファイルに出力する。次に，これを並列キャッシュ・シミュレータへ入力して，５状態並列キャッシュ・モデルとＫＬ１のメモリ参照特性の評価を行った。今回の評価では，ローカルメモリに格納可能できるように，プロセッサ毎にローカルなメモリ管理を行っているゴールレコード領域，および，命令コード領域も，全て一つの共有メモリ上に格納するものと仮定した。

4．2．並列キャッシュの構成

プロセッサ数は，８台で，プロセッサ当たりのキャッシュ容量は，４Ｋワード固定を標準にした。キャッシュ構成は，２５６カラム，４セット，４ワード・ブロックを基準にした。

バスサイクル数は，データのスワップイン，スワップアウト，キャッシュ間転送，無効化等に要するバスサイクル数である。共有メモリからのスワップインが１３サイクル，他プロセッサからのブロック・データのキャッシュ間転送が７サイクル，他プロセッサのキャッシュ・ブロックの無効化だけが２サイクルと仮定した。また，一つのメモリ操作が完了するまで，共有バスを解放しないと仮定した。

共有バスのビジー率の目安となる名目バス使用率は，上記の仮定によるバスサイクル総数と，リダクション総数から，次のように計算した。

名目バス使用率 ＝ バス使用時間 ／ 実行時間

ここで，バス使用時間は，バスサイクル数×50[nSec]，実行時間は，リダクション総数／目標性能の200[Kリダクション]／プロセッサ台数である。従って，名目バス使用率は，バス競合による待ち時間を除いたバス使用率といえる。

4．3．評価プログラムの特性

メモリ参照特性の評価には，メモリ参照回数，および，コード量がある程度大きなベンチマーク・プログラムが必要である。そこで，表４に示した特性の構文解析プログラムのＢＵＰをベンチマークに用いた。ＫＬ１Ｂは，ＰＩＭの機械語である[8]。

表４．評価プログラムの特性（８プロセッサ）

プログラム名	ＢＵＰ
ソースプログラム行数	3.2K
総リダクション回数	35.6K
ＫＬ１Ｂ命令の静的ワード数	3.3K
（内，一回以上参照された数）	1.9K
ＫＬ１Ｂ命令の実行ワード数	545.4K
サスペンド回数	1.6K
総メモリ参照回数	1,312.9K

ＢＵＰプログラムを実行した時の静的メモリ参照特性を表５に示す。全メモリ参照の内，約５０％は命令コードの読み出しである。命令コードを除いた領域のメモリ参照は，従来型言語に比べて書き込み比率が高い。複数のプロセッサ間で共有されるヒープ領域へのメモリ参照頻度は，全メモリ参照の１５％程度である。共有ヒープ領域中の未定義変数への具体化を排他制御するために行ったロック操作は，全ヒープ領域参照の約２０％である。

表５．ＢＵＰの静的メモリ参照特性（８プロセッサ）

領　域	書き込み	読み出し	ロック＆読み出し
環境（ヒープ領域）	7.5%	4.3%	2.9%
命令コード	0 %	49.5%	0 %
ゴールレコード	13.9%	13.8%	0 %
サスペンジョンレコード	1.1%	1.1%	0 %
メタコールレコード および通信用フラグ	0.6%	2.7%	0.6%
通信バッファ	1.0%	1.0%	0 %
合　計	24.1%	72.4%	3.5%

5．ＫＬ１のメモリ参照特性の評価

5．1．名目バス使用率とキャッシュ・ミスヒット率

1) キャッシュの総容量一定という条件，つまり，キャッシュのブロック長を増加するに従って，カラム数を減少させた場合の，名目バス使用率とキャッシュ・ミスヒット率の変化を図２に示す。ミスヒット率が一見高いが，スワップインが無いので，オーバヘッドがヒットに近いＤＷコマンドを，全てヒットと見做して計算し直すと，２５６カラム，４セット，４ワード・ブロックでのミスヒット率は，７．９％ではなく，２．７％となる。

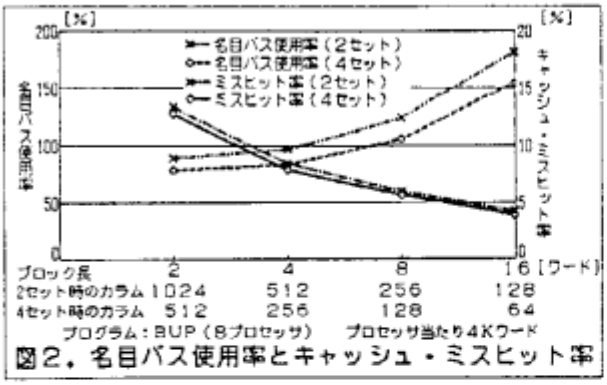


図2．名目バス使用率とキャッシュ・ミスヒット率

2） 一般に，キャッシュのブロック長を長くすると，キャッシュのミスヒット率は低くなるが，名目バス使用率は上昇する傾向がある。ブロック長を長くすると，ミスヒット率が低くなるということは，ローカリティが高いことを示している。しかし，それにもかかわらず，名目バス使用率が上昇するということは，スワップイン，スワップアウトのオーバヘッドが大きいことを示している。

3） ４ワード・ブロック固定で，２５６カラム，４セット構成と，５１２カラム，２セット構成を比較すると，２セット構成の方が約１６％程名目バス使用率が高くなる。従って，４セット化の効果は大きい。

５．２．キャッシュ容量と名目バス使用率

1） キャッシュ構成を変えて，プロセッサ当たりのキャッシュ容量を増加した場合の，名目バス使用率の変化を図３に示す。

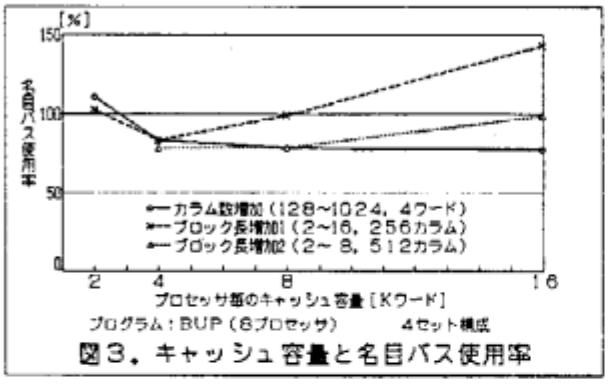


図3．キャッシュ容量と名目バス使用率

2） キャッシュに使用するメモリは，キャッシュ・ディレクトリ部を格納するアドレスアレイと，キャッシュ・データ部を格納するデータアレイに分けられる。キャッシュ容量を増加するには，ブロック長を固定して，カラム数（またはセット数）を増加する方法と，カラム数（およびセット数）を固定して，ブロック長を増加する方法がある。

ブロック長を固定して，カラム数（またはセット数）を増やせば，名目バス使用率は必ず下がる。しかし，この方法では，アドレスアレイの容量が増加する。マシンサイクルを短縮するために，アドレスアレイのオンチップ化を考えた場合には，できれば，アドレスアレイの容量を一定に保ちつつ，キャッシュのデータアレイ容量を増加したい。このためには，キャッシュのブロック長を長くしなければならない。しかし，ＫＬ１処理では，ブロック長を８ワード以上にすると，キャッシュ容量を増加しても，名目バス使用率は，低下しないことが分かった。従って，キャッシュのブロック長は，４ワードが良い。

3） ベンチマークに使用したＢＵＰプログラムの特性では，４Ｋワードよりもキャッシュ容量が少ないと，ヒット率が極端に下がりスワップイン，スワップアウトが増える。また，プロセッサ当たり８Ｋワード以上にすると，プロセッサ台数が８台程度でも未使用キャッシュ・エントリが生じる。以上の点を考慮して，２５６カラム，４セット，４ワード・ブロックで，プロセッサ当たり４Ｋワードを基準とした。

５．３．各領域毎のメモリ参照特性

1） ヒープ使い捨て型処理系におけるヒープ，ＫＬ１Ｂ命令コード，ゴールレコード，サスペンジョンレコード，メタコールレコード，通信バッファの各領域毎に，使用した領域の静的ワード数，アクセス回数，バスサイクル数，実行終了後に各領域かどれだけのエントリをキャッシュ上に占めているかを図４に示す。

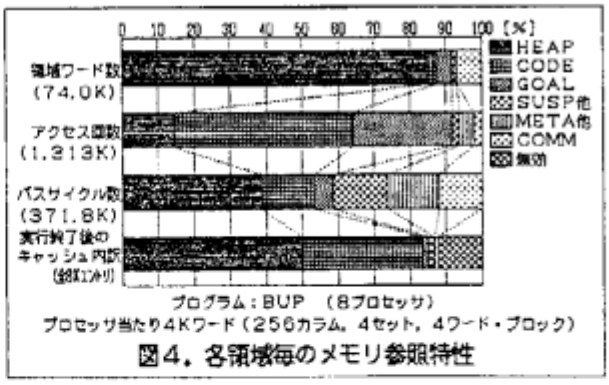


図4．各領域毎のメモリ参照特性

2） ヒープ領域は，全メモリ空間の約８５％を占めるが，アクセス回数は，全体の１５％程度しかない。従って，他の領域に比べてローカリティが低い。実際，バスサイクルの４０％近くは，ヒープ領域のアクセスに起因するものである。なお，ＭＲＢ（多重参照管理方式[9]）を用いた実時間ＧＣを行えば，ヒープ領域を使い捨てではなく，すぐに再利用できる。これにより，ヒープ領域のローカリティを上げることが提案されており，現在評価を進めている[10]。

3） 一方，ＫＬ１Ｂ命令コード領域は，全メモリ・スペースの２．６％程の領域に対して，全メモリ参照の

約５０％が集中する。従って，命令コード領域のローカリティは，他の領域に比べて極めて高いことが確認された。

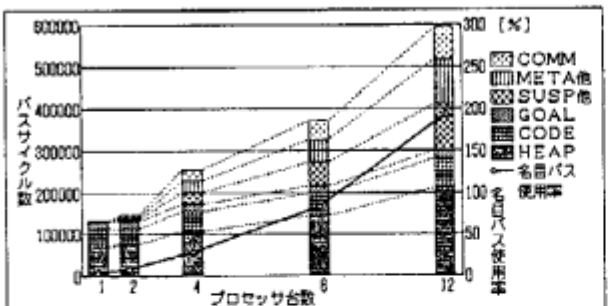
4) ヒープ領域は，アドレス空間が広いため，そのメモリ参照特性は，処理系の性能に大きな影響を与えるので，さらに，詳細な特性を調べた（表６）。その結果，ヒープ領域は，１ワード当たり平均３回のアクセスがあり，書き込みの数と読み出しの数がほぼ同数である。また，読み出しの内約４０％はロック操作が必要であった。従って，ハードウェア・サポートによって，ロックをかけた読み出し操作を高速化することが有効である。

表６．ヒープ領域の参照特性（８プロセッサ）

プログラム（ＢＵＰ） リダクション数＝35,752	全　体	リダクション当たり
ヒープ領域の静的ワード数 ヒープ領域の総参照数	62.8K 192.8K	1.8[語] 5.4
ヒープ領域への書き込み数 ヒープ領域からの読み出し数 内ロックを伴う読み込み数	98.1K 94.6K 38.0K	2.7 2.6 1.1

５．４．キャッシュ構成と台数効果

1) プロセッサ台数を増加した場合に，バスサイクル数と名目バス使用率がどのように変化するかを図５に示す。



プログラム：BUP （1～12プロセッサ）
プロセッサ当たり4Kワード（256カラム，4セット，4ワード・ブロック）
図5，プロセッサ台数によるバス使用量の変化

2) 名目バス使用率の上昇率が，バスサイクル数の上昇率よりも大きな理由は，ｎ台のプロセッサで実行すれば単純にｎ分の１の時間で実行できると仮定したからである。４～８台程度のプロセッサならば，現仕様の共有バス１本で接続可能であるとの見通しを得た。

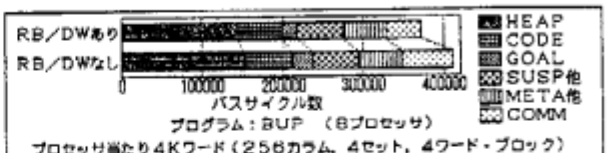
3) キャッシュ容量は，各プロセッサ当たり４Ｋワードと仮定したので，プロセッサ台数を増やすとその分総キャッシュ容量は増える。しかし，プロセッサ台数を増やすとキャッシュの総容量が増えた利点を打ち消す並列化によるアクセスの分割損が発生して，バスサイクル数が増加する。例えば，８台のプロセッサでＢＵＰプログラムを実行すると，約１０％が他のプロセッサのヒープ領域への参照となり，その分バスサイクル数が増える。

4) 命令コード領域に必要なバスサイクル数は，プロセッサ台数が１台から２台に増える時に一旦減少する。これは，２台にすると，共有メモリからのスワップインよりも高速なキャッシュ間転送により命令コードを転送できる確率が高くなるからである。しかし，プロセッサ台数を４台，８台と増やしていくと逆に増加する。これは，命令コード以外の領域は，ｎ台のプロセッサで実行した場合，アクセスの分割損が生じた分しかアクセス総量は増加しない。これに対して，命令コード領域は，ｎ台のプロセッサにコピーされるのことが多いので，バスをｎ倍近く使用するからであると考えられる。

なお，今回の評価では，クローズ・インデキシングを行っていない。そのため，ＫＬ１Ｂ命令の分岐頻度が高く，実行ＫＬ１Ｂ命令数が多い。今後，インデキシングを入れた評価を行えば，ＫＬ１Ｂ命令のフェッチに要するバスサイクルは減少すると予測される。

５．５．ＲＢコマンドの効果

1) ゴールレコード領域，および，プロセッサ間通信バッファ領域の管理に，ＲＢ（およびＲＰ）コマンドとＤＷコマンドを用いた場合の比較を図６に示す。



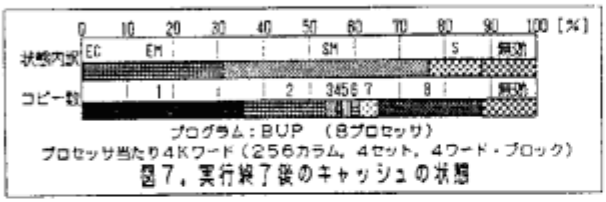
プログラム：BUP （8プロセッサ）
プロセッサ当たり4Kワード（256カラム，4セット，4ワード・ブロック）
図6，RB/DWコマンドの効果

2) ＲＢ（およびＲＰ）とＤＷコマンドを使用せずに，それぞれ単なる読み出しと書き込みコマンドに置き換えると，約１１％バスサイクルが増加してしまう。従って，ＲＢ（およびＲＰ）とＤＷコマンドは，有効である。

3) ＲＢコマンドの効果は，実際にＲＢ（およびＲＰ）コマンドを適用した，ゴールレコード領域とプロセッサ間通信バッファ領域だけでなく，他の領域にもある。これは，ＲＢ（およびＲＰ）コマンドによって，予め無効化されたエントリが増えているため，新しくキャッシュ・エントリを確保する際のスワップアウト頻度が下がるためであると考えられる。

5.6. 実行終了後のキャッシュ状態の分析

1) ＢＵＰプログラム実行終了後のキャッシュのキャッシュ状態，実行終了後に何台のプロセッサでデータを共有していたかを図７に示す。



プログラム：BUP （8プロセッサ）
プロセッサ当たり4Kワード（256カラム，4セット，4ワード・ブロック）

図7，実行終了後のキャッシュの状態

2) 無効キャッシュ・ブロックの総量は，全ブロックの１２％に達する。無効ブロックは，共有ブロックへの書き込みとロックを伴った読み込みによってできる場合と，ＲＢ（およびＲＰ）コマンドの強制的無効化によってできるものがある。無効ブロックが多いことは，一見無駄そうだが，次に利用する場合には，スワップアウトが不要であるというメリットがある。

3) 実行終了後に全プロセッサのキャッシュ上に残っているデータのコピー数を調べてみると，１台のプロセッサだけが所持していたキャッシュ・ブロックが約３６％で最も多いが，８台のプロセッサ全部でコピーを持っていたブロックも２３％に達する。このように多くのプロセッサで共有されていたブロックの大半は，命令コードであると考えられる。

6. おわりに

本稿では，まず，ＰＩＭのクラスタで採用する予定の５状態並列キャッシュ機構とロック機構について述べた。次に，ＰＩＭの対象言語であるＫＬ１のヒープ使い捨て型クラスタ内処理系と５状態並列キャッシュのシミュレータを用いてＫＬ１のメモリ参照特性を評価した。その結果，現在検討中の並列キャッシュ機構とＫＬ１処理系は，充分実用になるとの見通しを得た。

ＭＲＢ（多重参照管理方式）を用いた実時間ＧＣを行った時のメモリ参照特性については，現在評価中である。今のところ，１プロセッサの評価結果が出たところであるが，プログラムによっては，３５～９０％という非常に大きなバスサイクル低減効果が確認されている[10]。

今後，プロセッサ台数を増やして，ＭＲＢを用いた実時間ＧＣの評価を進めると共に，大規模プログラムに関しても定量的評価を行う予定である。

<参考文献>

[1]後藤，“並列推論マシンＰＩＭ”，昭和62年電気・情報関連学会連合大会，31-1，1987

[2]M.Sato, et al., "KL1 Execution Model for PIM Cluster with Shared Memory", ICLP '87, pp.338-355, 1987

[3]松本 他，“並列推論マシンＰＩＭ —並列キャッシュとロック機構—”，情報処理学会第34回全国大会予稿集 2P-6，1987

[4]松本 他，“並列推論マシンＰＩＭ —ＫＬ１のメモリ参照特性—”，情報処理学会第34回全国大会予稿集 3C-1，1987

[5]P.Bitar, A.M.Despain, "Multiprocessor Cache Synchronization Issues, Innovations, Evolution", ISCA '86, pp.424-433, 1986

[6]M.S.Papamarcos, J.H.Patel, "A low-overhead coherence solution for multiprocessors with private cache memories", ISCA '84, pp.348-354, 1984

[7]J.Archibald, J.Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model", ACM Trans. on Computer Systems, Vol.4, No.4, pp.273-298, 1986

[8]Y.Kimura, T.Chikayama, "An abstract KL1 machine and its instruction set", SLP '87, 1987

[9]木村 他，“ＫＬ１の多重参照ビットによるＧＣ方式について”，本ワークショップ 5B-2，1987

[10]西田 他，“ＭＲＢによる多重参照管理方式 —ＫＬ１処理系におけるキャッシュ特性の評価—”，情報処理学会第35回全国大会予稿集 2Q-8，1987