ICOT Technical Report: TR-342

TR-342

KLI疑似並列処理系における実時間GC方式のキャッシュ特性の評価

西田健次、木村康則、松本　明、後藤厚宏

February, 1988

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 擬似並列処理系における実時間 GC 方式のキャッシュ特性の評価

Evaluation of Real-Time Garbage Collection Scheme: Cache Characteristics on KL1 Pseudo Parallel Emulator

西田健次*　木村康則、松本 明、後藤厚宏

Kenji NISHIDA, Yasunori KIMURA, Akira MATSUMOTO, Atsuhiro GOTO

新世代コンピュータ技術開発機構†

Institute for New Generation Computer Technology (ICOT)

概要

並列論理言語 KL1 のようなメモリセルの破壊的書き換えを許さない言語を単純に実装するとメモリ領域を急速に消費してしまい、一括型 GC を頻発させる。並列プロセッサシステムにおいては、一括型 GC は、起動/終了時の PE 間同期のオーバヘッド、GC 処理中のキャッシュ特性の悪化による処理効率の低下などの理由により、システム全体の性能を著しく損なう。一方、実時間 GC は、実行時のオーバヘッドを伴うがキャッシュアクセスの局所性は保存されるため、キャッシュ特性の悪化による処理効率の低下は避けることができる。したがって、並列プロセッサシステムにおいては、オーバヘッドの小さい実時間 GC 方式の採用が必須であると考えられる。KL1 処理系における効率的な実時間 GC 方式として、ポインタタグに持たせた多重参照管理ビット (MRB) による実時間 GC 方式が提案されている。KL1 逐次処理系に MRB による実時間 GC を組み込んだ場合には、メモリ領域を使い捨てて行く場合に比べ、キャッシュ特性が大幅に改善される。そのため、実時間 GC を行った場合の方が、かえって、処理効率が高くなることが示されている。本稿では、KL1 擬似並列処理系に MRB による実時間 GC を組み込み、そのメモリ参照特性から、並列キャッシュ上での特性を評価し、KL1 並列処理系における MRB による実時間 GC の有効性を示す。

1　はじめに

ICOT では、第五世代コンピュータプロジェクトの一環として並列推論マシン PIM[2] の研究開発を行っている。PIM の核言語 KL1 は、制限を加えた GHC(Flat GHC)[8] に基づく並列論理型言語の一種である。KL1 は、メモリセルの書き替えといった副作用を持たない言語であり、並列処理システムに不可欠な同期や通信等の基本操作が自然に記述できる。その反面、メモリ消費速度が速いと言われる欠点を持つ。

逐次 Prolog の場合は、メモリをスタック的に使用し、バックトラック時に失敗した選択肢の実行で使用したメモリ領域を動的に回収することができるため、メモリ消費速度は深刻な問題とならない。また、メモリ領域の割り付けと回収がスタックトップ付近でを繰り返されるため、メモリアクセスの局所性を高く保つことができ、キャッシュのヒット率を高くすることができる[7]。一方、KL1 のような AND 並列論理プログラミング言語では、スタック的なメモリの動的回収が行えないため、単純に実装するとメモリ領域を単調、かつ、急速に消費してしまう。

このようなメモリの急速な消費は、一括型 GC を頻発させてしまう。さらに、メモリアクセスの局所性を悪化させるために、キャッシュのミスヒットやページフォールトが増加する。

PIM は、クラスタを用いた階層構造で、クラスタ内は、共有メモリ共有バス構成の並列プロセッサである。クラスタ内では、各 PE に持たせたキャッシュにより、バス競合を軽減して並列に処理を行うことを目指している。そのため、キャッシュのミスヒットによるバストラフィックの増加は、バス競合による性能低下をもたらす。また、一括型 GC は、メモリ領域全体にわたる処理であるため、キャッシュはミスヒットを繰り返すものと考えられる。そのため、GC の処理ではバス競合が頻発し、PE の並列動作はほとんど期待できない。したがって、一括型 GC の処理は、通常の処理に比べ実効効率が極めて低いことが考えられる。

並列プロセッサにおいては、処理の並列性、メモリ参照の局所性を得ると言う意味では、実時間 GC 方式が望ましいと言える。しかし、各データオブジェクトの参照カウントを管理し、更新するための実行時のオーバヘッドが問題となる。

KL1 では、ある節を実行した際に、そこで使用されているデータオブジェクトに対する参照数の増減を、コンパイル時に知ることができる。そのため、参照数管理用の命令を用意することにより、参照数の管理は比較的容易に行うことができる。また、各データオ

*JUNET: nishida@icot.junet, CSNET: nishida%icot.jp@relay.cs.net

†108 東京都 港区 三田 1-4-28 三田国際ビルヂング 21F., Phone: 03(456)3193

Telex:ICOT J32964

ブジェクトに対する参照は、大部分が単一参照であることが経験的に知られている[3, 5]。

このようなKL1の特徴を利用した実行時オーバヘッドの小さい実時間GC方式として、多重参照ビット(MRB)による実時間GC方式が提案されている[1]。この方式では、データオブジェクトではなく、ポインタ側で参照数を管理する。また、参照数を単一参照か多重参照かの2種類に区別するための1ビットのみとすることにより参照数管理のオーバヘッドを軽減しようとするものである。

MRBによるGC方式では、一度多重参照になったデータオブジェクトは、回収できないため、メモリの再利用の点では完全でない。そのためMRB方式は、一括GCと併用する必要がある。

単一プロセッサにおけるKL1処理系では、MRB-GCは、60-70%のメモリ回収率を持ち[3]、バス使用率を大幅に削減することが示されている[6]。しかし、並列プロセッサ上に実装した場合には、PE間通信、PE間でのメモリ回収の不均衡等がキャッシュ特性に影響を与えることが考えられる。

本稿では、KL1の並列実行時におけるMRBによる実時間GC方式の回収効率、及び、キャッシュ特性をKL1疑似並列処理系を用いて評価し、MRBによるGC方式の有効性について述べる。

2 GC方式

並列プロセッサシステムにおいて、一括型GCを実行する場合の問題点としては、以下の3点が考えられる。

- GC起動時/終了時に全PE間で同期が必要となる: GC起動時には、全PEにそれまでの処理を中断させ、同期をとらなくてはならない。全PEでの同期操作のオーバヘッドは、非常に大きいと考えられる。また、GC終了時にも、全PEでGCが終了したことを確認するため、同期が必要となる。
- GCのための処理中にデータアクセスに排他制御が必要である: 共有データに対する排他制御は、GC処理の並列性を低下させる。
- バスネックとなり処理効率が低下する: 共有バス構成をとる並列プロセッサシステムでは、GC処理中にシステム全体からのメモリアクセス要求がバスに集中するためバスに大きな負荷がかかる。また、これは、メモリ領域全体に対するアクセスであり、アクセスの局所性が期待できないため、キャッシュメモリでバストラフィックを軽減することは困難である。

以上の考察から、一括型GCは並列プロセッサシステムの処理効率を大きく損なうと考えられる。(付録A.参照)

したがって、実時間GC方式を導入し、一括型GCの発生を防ぐ、或いは、発生頻度を下げることによって並列プロセッサシステムの処理効率を改善することが重要である。

実時間GCでのオーバヘッドとしては、

1. メモリセル回収に必要なメモリアクセス
2. 実行時のメモリの参照数の管理

の2点が考えられる。

メモリ回収を行い、メモリセルの再利用を行うことにより、メモリアクセスの局所性を高め、キャッシュのヒット率を改善することができる。そのため、メモリアクセス数の増加は、バストラフィックの増加とはならないため、1のオーバヘッドが、処理効率に対する影響は少ないと考えられる。

KL1では、ある型の実行時に発生するデータオブジェクトへの参照数の増減が、型のコンパイル時に知ることができる。また、大部分のデータオブジェクトへの参照は、単一参照であると考えられている。KL1のこの性質を利用して、上記2のオーバヘッドを軽減した実時間GC方式として多重参照管理ビット(Multiple Reference Bit: MRB)による実時間GC方式が提案されている。

MRBは、データオブジェクトではなくポインタ側にデータへの参照数の情報を持たせたものである。また、参照数の管理を1ビット(単一参照か多重参照か)で行うため、GCのオーバヘッドを小さく抑えることができる。しかし、一旦多重参照になったものに対しては、参照数の管理を行えないため、回収することができない。そのため、MRBによる実時間GCは、一括型GCと併用する必要がある。

2.1 MRBによるGC方式

2.1.1 MRBの定義

MRBは、ポインタに付加された1ビットのリファレンスカウンタである。MRBビットをREF○ ("白い"と呼ぶ)、または、REF● ("黒い"と呼ぶ)であらわすことにすると、その意味は以下の様になる。

- REF○ : このデータへのポインタはこれ一本である。
- REF● : このデータへのポインタが他にもあるかも知れない。

ただし、未定義変数セルに対するMRBは扱いが少し異なり、以下の様になる。

- REF○ : この未定義セルに対する他のポインタは、白いものが1本か、或いは、黒いものが複数本存在する。
- REF● : このデータへのポインタが他にもあるかも知れない。

図1に未定義セル、具体化セルに対するポインタのMRBの許される組み合わせについてまとめる。

2.1.2 MRBの管理

KL1の実行では、以下のような操作によってデータオブジェクトへの参照パスが生成、消滅する(参照数が変化する)。

1. 変数、構造体などの生成
2. 変数、構造体などのゴール間にわたる分配
3. 変数のデレファレンス
4. ユニフィケーション

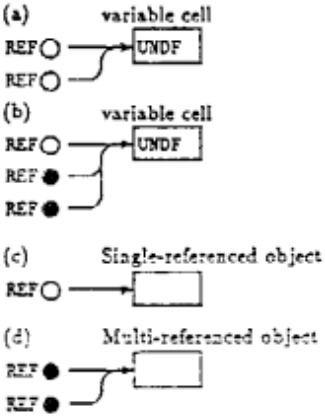


図 1: MRB の例

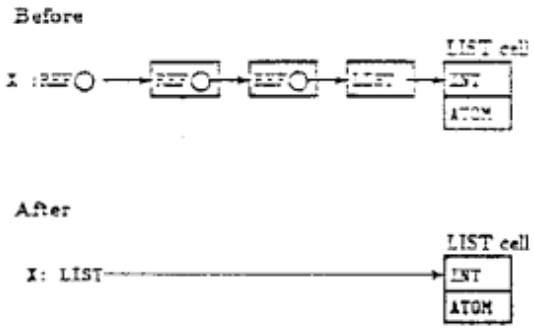


図 2: デレファレンスによる GC

5. 構造体要素の参照

上記の場合にデータオブジェクトを MRB 管理すると、KL1 の実行中にあるデータへの参照が最後であるかどうかが分かる。その場合、MRB が白であれば、そのデータは回収し再利用できる。

2.1.3 MRB を用いた GC 方式

MRB によるデータ回収が可能な場合を以下に示す。

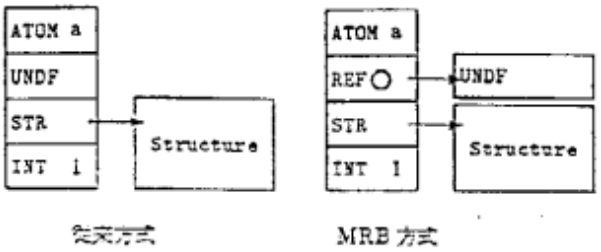
- 変数のデレファレンス: 変数のデレファレンス時の間接ポインタは、その間接ポインタセルへの参照が単一で (ポインタ MRB は白)、この間接ポインタの MRB が白であれば、デレファレンス時に回収できる (図2)。
- 構造体のユニフィケーション: 受動部で、ゴール引き数が構造体とのユニフィケーションに成功した場合、ゴール引き数として与えられた構造体 (下線部) の MRB が白であれば回収できる。

```
p([X|Y]) :- true | b(X), c(Y).
```

- ボイド変数とのユニフィケーション: ゴール引き数がボイド変数とユニファイした場合、そのゴール引き数 (下線部) の MRB が白であれば、回収できる。

```
p(foo, X) :- true | true.
```

- 回収される構造体の要素: ある構造体が回収される時、その構造体の要素もそれぞれの MRB が白であれば、回収される。以



従来方式　　MRB 方式

図 3: 構造体中の未定義セル

下の例では、第2引き数に構造体が与えられ、構造体同士のユニフィケーションが成功した場合には、その構造体は回収され、更に要素も再帰的にたどって回収できる。

```
p(X,Y) :- X=Y | true.
```

以上の他、コンパイル時に回収できるタイミングがわかる場合には、そのクローズのコミット時に回収用の命令を発生する。

2.1.4 MRB-GC におけるメモリ管理方式

MRB-GC では、回収されたメモリセルを、データオブジェクトのサイズごとのフリーリストで管理する。本稿の評価では、MRB-GC の対象となっているのは、変数とコンスセルであるため、1 ワードセルと 2 ワードセルの 2 種のフリーリストを使用する。メモリアクセスの局所性を高めるため、最も最近にフリーリストに回収されたセルから使用して行くが、フリーリストが空の場合には、メモリ上のヒープ領域に新たなセルを割り付けて行く。

MRB-GC を PIM のような共有メモリを持つ並列プロセッサに実装する場合、フリーリストを全システムで共通のものとすると、各 PE からのアクセスが集中してしまい、メモリセルを回収する度に PE 間でのバストラフィックが発生する。そこで、フリーリストを PE ごとに管理する。これにより各 PE 上でのキャッシュのヒット率が向上し、バストラフィックを削減すること期待できる。ヒープ領域についても同様に PE ごとに別の領域を割り当てる。

また、構造体の要素に対する MRB の管理を完全に行うために、構造体の中に未定義セルを直接置かずに、間接ポインタを通して構造体の外に未定義セルを割り付ける (図3)。

MRB-GC におけるメモリ管理方式の問題点を以下にまとめる。

- メモリの回収率に PE 間で不均衡が生じた場合には、他の PE には、充分なフリーリストが存在するにもかかわらず、自 PE のフリーリストが空のため新たなセルをヒープ領域に割り付けなくてはならない。この場合には、使用するメモリ領域が増加してしまう。
- 構造体の中に未定義セルを置かずに、間接ポインタを使用するため、その分のメモリ使用量が増加する。ただし、MRB が白であれば、この増加分は回収できる。

3 評価条件、及び、評価項目

3.1 評価手順

ベンチマークプログラムを、KL1擬似並列処理系上で実行し、GCを行った場合と、GCを行わない場合の2種類のメモリ参照情報を取り出した。次に、メモリ参照情報を元にキャッシュシミュレーションを行いキャッシュ特性の評価を行った。

今回の評価に使用したKL1擬似並列処理系は、以下に挙げる特徴を持つ。

- 1リダクションを処理の単位とする
- 1リダクションごとにPEを切り替えることにより擬似並列的にプログラムを実行する

メモリセルに対するロック操作などは、1リダクションの中に閉じた処理であるため、PE間でのロック競合などを反映した実行は行うことができない。しかし、全体のキャッシュ特性などの測定には充分な精度を持っていると考えられる。

KL1擬似並列処理系において、MRB-GCを用いてメモリの再利用を行うのは、変数、及び、コンスセル領域のみである。ゴールレコード、サスペンドレコードなど各種レコードに対してはフリーリスト管理を行い、再利用を行っている。

KL1擬似並列処理系は、コンパイル時のオプション設定により、GC-ON(MRBによるGCを行う)と、GC-OFF(MRBによるGCを行わない)の2種類の設定が可能である。GC-OFFの場合、メモリ回収とメモリ回収時に必要なメモリアクセスが抑制される。しかし、メモリ回収を行わない場合には必要のないMRBの管理によるメモリアクセスは抑制されない。また、構造体に対する扱いは共通であるため、間接ポインタセルによるメモリ領域の増加は測定できない。同様に、実行するベンチマークプログラムは、どちらの設定でも共通のものとしたため、メモリ回収命令に対するフェッチ(メモリアクセス)も抑制されない。

キャッシュシミュレーションは、以下に述べる2種類のキャッシュ構成について行った。

- 256カラム、4セット、4ワード/ブロック (PIMに実装することが予想される程度の容量)
- 16Kカラム、4セット、1ワード/ブロック (無限大キャッシュの近似)

今回の評価に使用したベンチマークプログラムは 、構文解析プログラムBUPで、約36000リダクション、キャッシュアクセス数は、約130万である。

3.2 並列キャッシュプロトコル

並列キャッシュとしては、5状態並列キャッシュを用いる[4]。5状態並列キャッシュは、KL1用に最適化されたキャッシュプロトコルを持つインバリデート型のライトバックキャッシュである。

キャッシュの5つの状態を以下に示す。

- EC: Exclusive Clean PE間で共有されておらず、変更されていない
- EM: Exclusive Modified PE間で共有されておらず、変更されている
- SC: Shared Clean PE間で共有されており、変更されていない
- SM: Shared Modified PE間で共有されており、変更されている
- I: Invalid 無効化された状態

また、CPUコマンドとして、以下に示す3種を持つ。特にKL1用に最適化されたコマンドとしては、メモリを急速に消費するというKL1の特徴を考慮したDWコマンド、また、PE間通信が頻繁に発生するというKL1の特徴を考慮して設計されたER、RPコマンドがある。

- R: Read 読みだし
- W: Write 書き込み
- DW: Direct Write 書き込みに先立ってキャッシュブロックをフェッチせずに、直接自プロセッサのキャッシュに書き込む。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に対してのみ使用可能である。KL1処理系でDWコマンドを適用できるのは、メモリ領域を新たに確保する場合、および、次に述べるER、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。DWコマンドを使用することにより、バストラフィックを発生せずにメモリ領域を新たに確保する事が出来る。
- ER: Exclusive Read 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する。
- RP: 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する。

3.3 GC特性における評価項目

KL1を並列に実行した場合、メモリセルの回収率に影響を与える項目としては、節の実行順序の変化によるメモリ参照特性の差、PE間のメモリセル回収の不均衡の2つが挙げられる。節の実行順序の影響は、REFセルの増加、ゴールのサスペンドなどの影響により全体のメモリ回収率を悪化させる可能性がある。メモリ回収の不均衡は、回収したメモリセルをPEごとのフリーリストで管理するため、メモリを消費するPEと回収するPEが異なる場合に発生する。メモリセルの回収に不均衡が生じた場合、見掛け上のメモリ

回収率は変化しないが、実効的な回収率が低下し、必要なメモリ量が増加する。

今回の評価では、

- 並列実行時のメモリ回収率
- 実系に割り付けたメモリセルの個数

を測定し、並列実行時のメモリ回収率の変化と、メモリ回収の不均衡の影響を評価する。

3.4 キャッシュ特性における評価項目

キャッシュ特性を評価する際、従来はヒット率が用いられてきた。しかし、今回評価に用いたキャッシュプロトコルでは、BWコマンドのようにミスヒットすることを前提として使用され、ミスヒットによるバストラフィックが発生しないものがある。そのため、ヒット率は、キャッシュ特性を評価する指標としては、極めて大雑把な物でしかない。

共有バスを使用する並列プロセッサでは、バス競合による性能低下を予測するために、バスにかかる負荷が問題となる。そこで、バス負荷を示す指標として名目バス使用率を評価する。

名目バス使用率は、必要とされるバスサイクル数を、プロセッサの速度(PE台数 * 単対当能)で正規化したものである。名目バス使用率の計算式を式1に示す。

$$R_{usage} = \frac{B_{cy} B_{per}}{R_t/(S_r N)} \qquad (1)$$

ただし

- B_{cy} : バスサイクル数
- B_{per} : バス周期
- R_t : 総リダクション数
- N : PE台数
- S_r : PE単体の速度 reduction/second (RPS)

とする。必要とされるバストラフィックが一定の場合、名目バス使用率はPE台数に比例して大きくなる。

次に、変数、及び、コンスセル領域に対して使用されたバスサイクル数を測定し、MRB-GCのキャッシュ特性に対する得失を評価する。

4 評価

4.1 メモリ回収率の変化

表1に、変数セル、コンスセルの延べ使用量と回収率の変化を示す。これによると、メモリの延べ使用量はPE台数が増えた場合、僅かだが減少している。この原因は、逐次実行時と並列実行時の節の実行順序が異なるため、間接参照セルの数が変化したためと思われる。メモリ回収率は、メモリ延べ使用量の減少のために、PE8台で2%悪化しているが、未回収メモリセルは増加していない。即ち、並列化によってメモリ回収率は、変化しないことが示された。

PE台数	延べ使用量	未回収セル	回収率(%)
1	71400	20016	72
2	69642	20017	71
4	68712	20018	71
8	66513	20022	70

表 1: メモリ回収率の変化

PE台数	GCあり	GCなし
1	20092	71400
2	20144	69642
4	20245	68712
8	20530	66513

表 2: 実系に割り付けた領域の変化

表2に、実系に割り付けた変数、コンスセル領域の大きさを示す。メモリセルの回収率にPE間での不均衡が生じると、実系に使用するメモリ領域が大きくなると考えられるが、メモリ領域の増加は約2.5%であり、今回の評価では回収の不均衡による影響は出ていない。しかし、これに今回実行したベンチマークプログラムの規模が小さいため、一般的な傾向を示しているとは言えない。

4.2 キャッシュ特性の評価

図4にキャッシュアクセス数の変化を示す。キャッシュアクセス数の増加は、メモリ回収を行う際のオーバヘッドを示す。これにより、メモリ回収に関わるオーバヘッドは、PE台数に関わらずほぼ一定で、約5%と小さいものであることが示された。

図5にBUPでのヒット率の変化を、図6に名目バス使用率を示す。GCを行った場合の方が、PE1台から8台までヒット率は高い。しかし、名目バス使用率は、PE8台においてGCを行った場合に悪化している。この結果は、キャッシュ特性の指標としては、ヒット率よりも名目バス使用率の方が適当であることを示してい

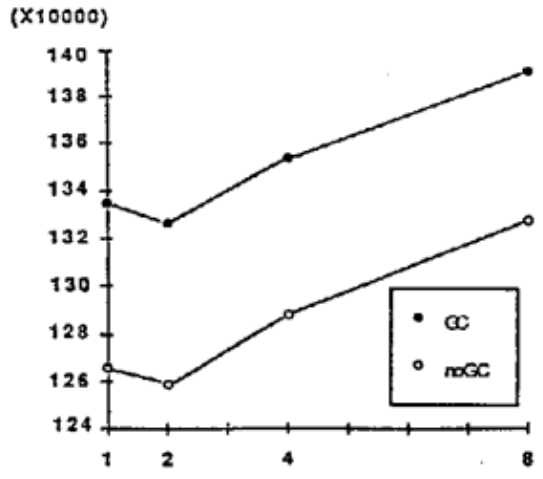


図 4: キャッシュアクセス数の変化

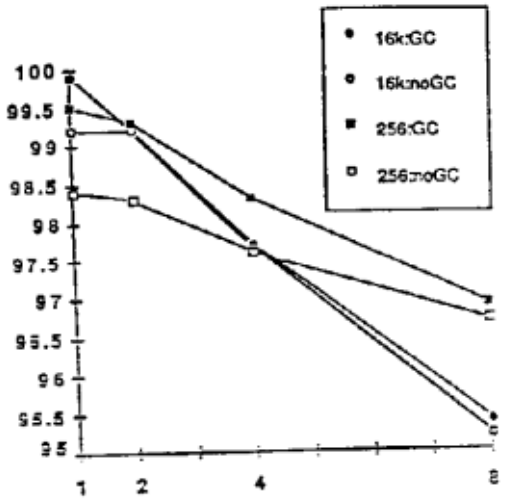


図5: ヒット率の変化

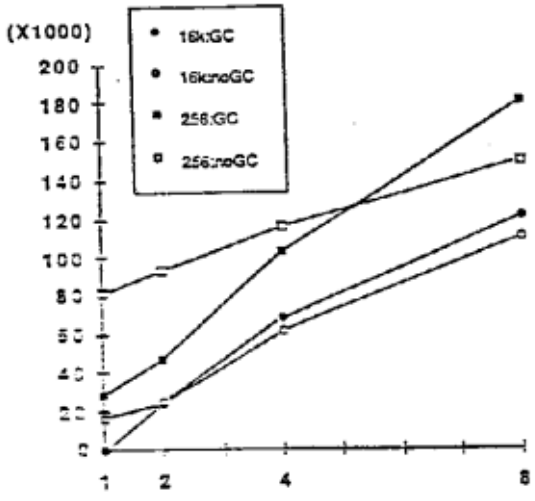


図7: バスサイクル数の変化

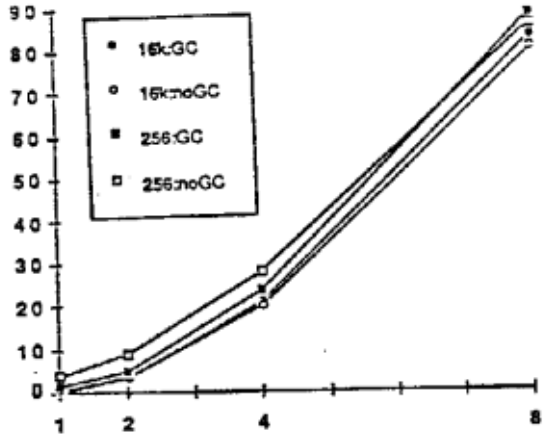


図6: 名目バス使用率の変化

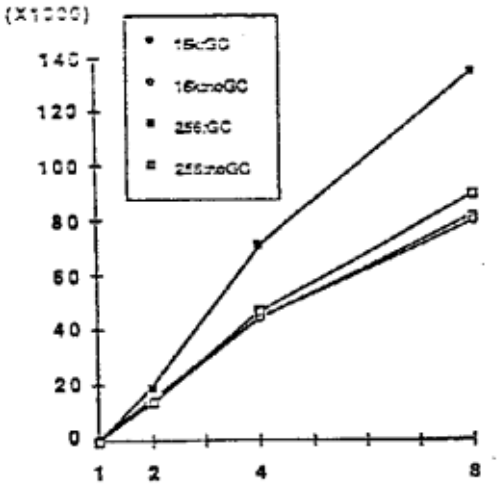


図8: キャッシュ間転送の変化

る。

PE8台における名目バス使用率の増加は、約2.5%であり実効性能に大きな影響を与えるものではない。

図7に変数、コンスセル領域に対するバスサイクル数の変化を示す。 4ワード/ブロックのキャッシュ構成では、PE1台においては、GCを行うことによってバスサイクルを約3分の1に削減できているが、PE台数が増えるにつれ、急激にバスサイクル数が増加し、PE8台では約20%もGCを行わない場合よりも多い。 1ワード/ブロック構成でも、同様の傾向が見られるが、その差は、4ワード/ブロックの場合よりも小さい。

MRB-GCを行わない場合、使用するメモリ領域はMRB-GCを行った場合の約3倍を必要とする。しかし、新たなメモリセルを割り付ける際のメモリアクセスは、DWコマンドを使用することにより、バストラフィックには現れない。一方、MRB-GCを行った場合、フリーリストからメモリセルを取り出す場合には、DWコマンドが使用できないため、ミスヒットした場合にはバストラフィックを発生する。フリーリストがミスヒットする原因としては、フリーリストを含むキャッシュブロックにある変数セルをPE間通信に使用したため、他のPEフリーリストがによって無効化されてしまうことが考えられる。

そこで、PE間通信に使用されるキャッシュ間転送のバスサイクルの変化を図8に示す。

1ワード/ブロックのキャッシュ構成ではMRB-GCの有無は、キャッシュ間転送によるバスサイクル数にはほとんど変化がない。この約8万サイクルがPE間通信に本質的に必要なバスサイクルであると考えられる。PE8台の場合に注目すると、 4ワード/ブロックのキャッシュ構成では、MRB-GCを行わない場合、約10%バスサイクルが増加する。これは、本来PE間通信に必要のない(キャッシュブロック内の)メモリセルも転送しなければならなくなったことによるオーバヘッドと考えられる。 4ワード/ブロックでMRB-GCを行った場合、キャッシュ間転送によるバスサイクルは、更に約40%増加する。これは、メモリセルの再利用により、1つのキャッシュブロックが複数のPE間で共有される確率が高くなったためと考えられる。即ち、メモリセル回収のためのメモリ書き込み操作によって他のPEのキャッシュブロックを無効化してしまうことが、MRB-GCを行った場合のバスサイクル数増加の主な原因と考えられる。また、無効化を行うためのバスサイクルもMRB-GCを行うことにより、ほぼ2倍に増加している(図9)。これは、読み出しによりキャッシュ間転送が発生し、その後で書き込みによって他のPEのキャッシュを無効化していることを示す。

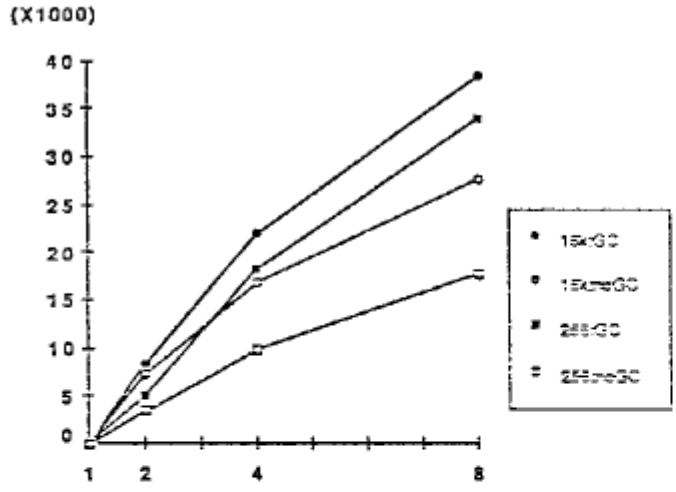


図 9: 無効化コマンド数の変化

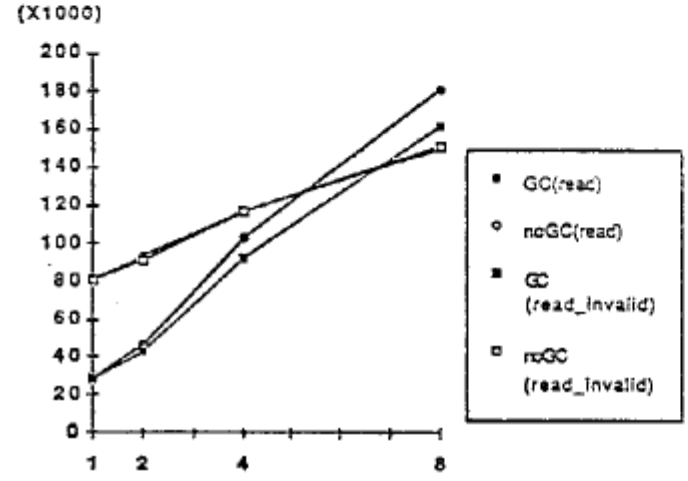


図 10: 無効化を読みだし時に行った場合のバスサイクル数

無効化コマンドの発生を抑えるため、変数、コンスセル領域からの読み出しと同時に他のPEのキャッシュを無効化を行うようにすることにより全体のバスサイクルを削減することができる(図10)。このような最適化は、現在のところは実装されていないが、将来、KL1処理系の最適化を行う際に取り入れることが検討されている。

5 結論

KL1を並列に実行した際のMRBによる実時間GC方式の得失がまとめると

- 並列実行時にもメモリの回収率は悪化しない
- PE間のメモリ回収の偏りは問題とならなかった
- PE台数が少ない時は、バス使用率を削減できる
- PE台数が増えた場合は、キャッシュ間転送の増加によりバス使用率が悪化するが、その度合いは約2.5%と僅かである
- キャッシュ構成、キャッシュプロトコルを意識することにより更にバス使用率を改善することが可能性がある

となる。

以上より、MRBによる実時間GC方式は、PEの実効性能を損なうことなく、一括型GCの間隔を2倍から3倍に引き伸ばすことができる。一括型GCの実効性能低下に与える影響を考慮するとMRBによる実時間GCはKL1処理系を実装する際に有効な方式であると考えられる。

6 おわりに

今回の評価では、PE間のメモリ回収率の片寄りは問題とならなかったが、今後、より大きなプログラムで評価を行う場合には、実質的な回収率の低下が大きな問題となる。そこで、フリーリストの形で回収したメモリセルをPE間で融通することにより、片寄りを是正する方式が検討されている。また、実時間GCでの回収率を向上する方式として、多重参照時のみ参照カウントを導入する方式も検討されている。

謝辞

本論文を執筆するにあたり、ICOTのPIM・マルチPSI開発グループの諸氏、特に、ICOT第4研究室の近山隆氏、中島克人氏、佐藤正俊氏、今井明氏、三菱電機の吉内信仁氏に貴重なコメントを頂いたことを感謝致します。また、本研究の機会を与えて頂いたICOT渕所長、内田第4研究室長に感謝致します。

参考文献

[1] T. Chikayama and Y. Kimura. Multiple Reference Management in Flat GHC. In *Proceedings of the Fourth International Conference on Logic Programming*, pages 276–293, 1987. (also in ICOT Technical Report, TR-248).

[2] A. Goto and S. Uchida. Toward a high performance parallel inference machine –the intermediate stage plan of pim–. TR 201, ICOT, 1986. (also in Future Parallel Computers, LNCS 272, 299–320, Springer-Verlag).

[3] Y. Kimura, K. Nishida, S. Miyauchi, and T. Chikayama. Realtime GC by Multiple Reference Bit in KL1. In *Proceedings of the Data Flow Workshop 1987*, pages 215–222, Oct. 1987. (in Japanese).

[4] A. Matsumoto et al. Locally Parallel Cache Designed based on KL1 Memory Access characterestics. TR 327, ICOT, 1987.

[5] N. Miyauchi, Y. Kimura, T. Chikayama, and K. Kumon. Multiple Reference Management by MRB –GC Characteristics on KL1 Emulator–. In *35th Meeting of Information Processing Society*, Sept. 1987. (in Japanese).

[6] K. Nishida, A. Matsumoto, Y. Kimura, and A. GOTO. Multiple Reference Management by MRB –Cache Characteristics

on KL1 Emulator-. In *35th Meeting of Information Processing Society*, Sept. 1987. (in Japanese).

[7] K. Taki, K. Nakajima, H. Nakashima, and M. Ikeda. Performance and architectual evaluation of the PSI machine. In *Proceedings of the Second International Conference on Architectual Support for Programming Language and Operation Systems (ASPLOS)*, pages 128–135, 1987.

[8] K. Ueda. Introduction to Guarded Horn Clauses. TR 209, ICOT, 1986.

A　一括型GCの性能に与える影響

並列推論マシンPIMにおける一括型GCの性能に対する影響を簡単に試算してみる。[1]

- N : PE台数
- S_r : PE単体の速度を reduction/second (RPS)
- C_m : PE単体でのメモリ消費速度を word/reduction
- A_m : メモリ領域の大きさを word

とした場合、一括型GCの頻度I_{gc}は (バス競合などによる性能低下を無視した場合)

$$I_{gc} = \frac{A_m}{NS_rC_m} \tag{2}$$

次に、全メモリ領域を検索して、必要なセル(生きセル)のみを集めるスライディングGCに必要な時間を求める。スライディングGCに必要なメモリアクセスは、

1. マーキングのために生きているセル全部に1回ずつ read/write
2. スライディングの準備のため全メモリ領域を1回 read
3. 生きセルの内上向きポインタを含むセルに対し1回 write、1回 read/ write
4. 3でのポインタに指されているセルに対し1回 read/write、1回 write
5. 実際のスライディングのため、生きセルに対し、1回 read、1回 write

である。

ここで示したように、スライディングGCでは、全メモリ領域に対して1回、生きセルに対して4回、生きセルの内上向きポインタを含むものにはさらに2回のメモリアクセスが必要である。また、このようにメモリ全域に対する処理では、キャッシュ機構がうまく働かないため、バススループットが処理ネックとなると考えられる。

そこで、

- S_b バススループット (word/second)
- R_l メモリに占める生きセルの比率
- R_{lup} 生きセルの内上向きポインタであるセルの比率

とすると、スライディングGCにかかる時間T_{sgc}は

$$T_{sgc} = A_m(1 + 4R_l + 2R_lR_{lp})/S_b \tag{3}$$

また、メモリ領域を2等分して、生きセルのみを旧領域から新領域に移すコピーイングGCで必要なメモリアクセスは、

1. 旧領域の生きセルに対し1回 read/write
2. 新領域に生きセルを1回 write
3. 新領域にコピーしたセルをスキャンする (1回 read)
4. 3で読んだセルが旧領域へのポインタなら旧領域を read
5. 4で読んだセルが既に新領域に移っていれば3のセルに write 旧領域に残っていれば1へ

となる。したがって、全生きセルに対して3回、ポインタに対しては更に1回のメモリアクセスが必要となる。コピーイングGCに必要な時間T_{cgc}は、

- R_{lop} 旧領域へのポインタである確率

とすると

$$T_{cgc} = \frac{A_mR_l(3 + R_{lop}}{2S_b} \tag{4}$$

現在設計中のPIMは、単体性能200KRPSのPE8台で1クラスタを構成する。1クラスタ当たりのメモリは、32Mワード程度実装される予定である。また、バススループットは、10Mword/sec程度である。これまでの評価から、実時間GCを行わない場合のリダクション当たりのメモリセル消費速度は、2-3ワードである事がわかっている。これを元に3-4式よりGC間隔、GC時間を求めてみる。(簡単化のためR_{lup}, R_{lop}は共に0とする)

スライディングGCの場合、

$$I_{gc} = 6.7 \ \ (C_m = 3) \tag{5}$$

$$T_{sgc} = 3.2 + 12.8R_l \tag{6}$$

となる。一方、コピーイングGCではGC間隔は、スライディングGCの場合の1/2,GC時間は、

$$I_{gc} = 3.4T_{cgc} = 4.8R_l \tag{7}$$

となる。PIMの実効性能は、

$$S_e = \frac{I_{gc}S_r}{I_{gc} + T_{gc}} \tag{8}$$

と考えられるので、スライディングGCではR_lが約25%で、、コピーイング GCでもR_lが70%(メモリ領域全体に対しては、35%)で、実行性能は半減してしまう。このように一括型GCの頻度と効率が、PIMの性能を決定してしまうといっても過言ではない。また、ここでの見積もりは、生きセル中のポインタに対する手間を無視しているため、実際には一括型GCによる性能低下は、より大きいと考えられる。

[1] 本検討は、ICOT第4研究室近山隆氏の内部メモによる評価に基づいている