ICOT Technical Report: TR-448

TR-448

KL1並列処理系の評価
－メモリ消費特性とGC－

佐藤　正俊(神定知)、後藤　厚宏

January, 1989

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 並列処理系の評価 – メモリ消費特性と GC–

佐藤正俊*, 後藤厚宏
新世代コンピュータ技術開発機構 (ICOT)

概要

共有メモリ結合のマルチプロセッサ上に並列論理型言語 KL1 の並列処理系を開発し、KL1 プログラムのメモリ消費特性を評価した。KL1 のような単一代入型の言語では、メモリ資源が急速に消費され、ガーベジコレクション性能が大きな問題となる。そこで、コピー方式による一括型 GC を並列実行する機能を KL1 並列処理系に実装し、KL1 プログラムの並列実行時のメモリ消費特性を考察すると共に、ガーベジコレクションを並列実行する効果と性能への影響を評価した。その結果、メモリ消費スピードに性能向上とほぼ同じ割合で伸びていき、アクティブセルの割合はプロセッサ台数が増すに従って増加する傾向があることがわかった。また、GC 性能の台数効果はリダクション性能の台数効果より低いことが分かった。

1 はじめに

ICOT では並列推論マシン (PIM)[1] を開発中である。PIM のクラスタ内での KL1[2,3] の言語処理方式を検討するために、共有メモリ結合マルチプロセッサシステム上に KL1 の実並列エミュレータを実装し、KL1 の並列処理系の評価 [4,5,6] を行っている。KL1 は、制限を加えた GHC[7](フラット GHC) に基づく並列論理型言語であり、並列プロセス間の同期や通信を副作用なしに自然に記述できるという特徴を持つ。しかし、この副作用を持たない特徴によりメモリ資源を急速に消費し、ガーベジコレクションが実行性能に与える影響は、大きくなると予想される。

KL1 のガーベジコレクション方式には、実行時に不要となったメモリを逐一再利用する GC と実行を中断してメモリを回収する一括型 GC の 2 つの方式に分類できる。前者の方式で、完全にゴミを収集するには、実行時に大きなオーバヘッドが要求され、実行性能上大きな負担となる。そこで、PIM の GC 方式として、実行時には小さなオーバヘッドで、大部分のゴミを回収できる MRB[8,9] 方式と、残りのゴミを一括型 GC で回収する方法を組み合わせることを検討している。

一括型 GC の 1 方式であるコピーイング GC の GC コストは、GC 時のアクティブなセル量で決まる。ここで、KL1 のような副作用を持たない単一代入型言語では、単一代入規則により生成されるゴミがメモリ消費の大部分であると予測でき、アクティブセルの割合は少ないと考えられる。つまり、KL1 処理系では、コピーイング GC が、少ないコストで実現できることが期待できる。

本稿では、コピーイング GC の並列版を、実並列 KL1 処理系に実装し、KL1 の並列実行時のメモリ消費特性を評価するとともに、実装した並列コピーイング GC の評価を行ったので報告する。以下では、2 章で KL1 処理系の概要としてメモリ管理方法と並列コピーイング GC の実現方式を述べる。3 章では、評価に使用したベンチマークの概要を述べ、4 章では、そのベンチマークの並列実行時のメモリ消費特性として、メモリ消費スピードとアクティブセルについて評価する。5 章では、並列コピーイング GC の評価として GC 性能と台数効果を示し、そのオーバヘッドを分析する。

2 KL1 処理系の概要

2.1 KL1 処理系とメモリ管理

KL1 の実行はゴールリダクションであり、その実行は、ゴールレコードで表されるゴールのコンテクストをもとに、プログラムである定義節に従いヒープ上のデータをアクセスしながら進められる。また、ゴールの実行順序はゴール間で共有するデータの依存関係のみにより決まり、実行を試みた時点でそのゴールが実行可能でなければ、サスペンドレコードを用いた bind-hook 構造により、そのゴールは実行を待たされる。

KL1 並列処理系 [6] は、共有メモリ結合マルチプロセッサシステム (Sequent Inc. Symmetry System[10]) 上に言語 C を用いて実現している。エミュレートするプロセッサや共有メモリは、処理系の起動時にシステム関数を用いて実際のプロセッサやメモリを割り当てている。各プロセッサは、実行時のメモリアクセスのローカリティを上げるために、レディゴールスタックやメモリ資源のフリーリストを各プロセッサで個別に管理しており、ゴールのスケジューリングは、各々のレディゴールスタックをもとに LRDF(Left-to-Right,Depth-First) で行っている。負荷分散方法は、アイドルプロセッサがグローバルなデータをもとにレディゴールを多く持つプロセッサにゴールを要求する

*現在、沖電気工業株式会社、〒108 東京都港区芝浦 4-11-22 1 号制御総合システム研究所、ソフトウェア研究室 (TEL: 03-454-2111,ex.2735,masatosi@sun1.oki.lab.oki.junet)

方法である。プロセッサ間通信やメモリ管理は、処理系でエミュレートしている。KL1 並列処理系で扱うデータとその管理方法は、表1のように分類される。[1]

これらのデータは実行するプログラムにより、また、並列実行時には、プロセッサ毎にその使用量が異なる。そこで、本処理系では、全メモリをコピーイング GC 用に2面に分け、各面をさらにブロックとして全プロセッサの共通資源として管理する。各プロセッサは、必要に応じてこのブロックを獲得し、このブロック(以下、ヒープとも呼ぶ)から必要に応じて各データを割り当てる方法を採っている。つまり、データが必要なときに、フリーリストが空ならば規定量のレコードをプロセッサの持つヒープから割り当てる。ここで、評価した処理系では、1ブロックは、5120 ワードで、1面は、64 ブロックとしている。

例えば、KL1 のデータのような一括 GC のみで GC を行うデータは、ヒープから直接割り当て、ゴールレコードのような、フリーリストにより再利用を行っているデータは、フリーリストが空の場合のみ要求に応じて共有資源より割り当てて使用している。また、ゴールレコードの管理は、各プロセッサのメモリアクセスのローカリティを保ち、かつ、分散環境での資源管理のオーバヘッドを少なくする目的で、プロセッサ間に渡るゴールレコードの積極的な再配置を行っていない。つまり、プロセッサ間のゴールレコードの融通は行わず、一括 GC により一度共有資源に戻し、他のプロセッサが必要に応じて共有資源より割り当てることで実現される。

2.2 並列コピーイング GC

並列コピーイング GC は、単一プロセッサ用コピーイング GC を密結合マルチプロセッサ上で並列に実現するもので以下の実現上のポイントがあり、これらのポイントについてその実現方法を述べる。

- リダクションフェーズと GC フェーズ間での同期制御: リダクションフェーズでのメモリ不足をリダクション間で検出し、他の全てのプロセッサにメッセージ通信で通知し、GC フェーズに入る。このフェーズの切り替えの同期は、共有メモリ上のカウンタを用いて実現する。同期の必要となる回数は、3回である。
- 並列にコピーする時の排他制御: 排他制御は、マーキング時のみ必要となり、コピーイング時は新たなヒープをプロセッサ毎に獲得しコピーイングを行うので不要となる。
- 負荷バランス方法: GC のルートは各プロセッサ上のレディゴールスタックのゴールレコードであり、GC の負荷バランスの粒度はこのゴールレコードとした。負荷の分配は各プロセッサ毎のレディゴールスタックを1本のスタックとして扱い、各プロセッサが、GC ルートであるゴールレコードを、順次取り出し GC していく方法を実装した。この方法の評価は、5節で行う。

一括型並列 GC 方式のアルゴリズムは、以下のとおりである。

1. リダクション実行中の1プロセッサでヒープが不足し共有資源から新たなブロックを得ることができない場合を GC の契機とする。[2] この時、このプロセッサが GC マスタとなり GC を開始する。
2. GC マスタは、他のプロセッサに GC 開始メッセージを送り、他の全てのプロセッサが GC 状態に入るのを待つ。他のプロセッサは、リダクションの切れ目で GC 開始メッセージを受け、GC に入る。(同期1)
3. 全てのプロセッサが GC 状態に入ると GC マスタは、共有の資源プールを旧資源から新資源へ切り替える。(ここではコピーイング GC であるので、資源プールを2面使用する。)他プロセッサはこれを待ち合わせる。(同期2)
4. 各プロセッサは、各自の資源管理テーブルをリセット(ヒープトップや各レコードのフリーリストの初期化)し、GC を開始する。
5. 各自の GC は、1本に統合したレディゴールスタックからゴールを取り、そのゴールを GC ルートとしコピーイング GC を行う。この時、サスペンドゴールは、共有のスタックに積み、分散の対象にする。各自の GC に必要なヒープの割り当ては、共有資源プールから必要に応じて割り当てる。(リダクション時のメモリ管理と同じ方法である。)
6. 各プロセッサは、全てのプロセッサの GC 終了を待ち合わせ、リダクションに戻る。(同期3)

3 ベンチマーク

KL1 並列処理系の評価に使用したベンチマークは、以下のものである。これらのベンチマークは、GC が評価できる程度の大きさであり、そのプログラム規模(実行時間)と処理系の性能(プロセッサ1台での RPS と台数効果)を表2に示す。

- 積木パズル: 大きさは 5x4x3。(puzzle[11])
- KL1 コンパイラ:BUP のプログラムをコンパイルする。(klicmp)
  プログラム中で、ベクタを多く使用する。

[1] 本処理系では、MRB による GC はサポートしていない。

[2] 実際に、GC を開始するのは、そのリダクションの終了後のリダクションの切れ目である。

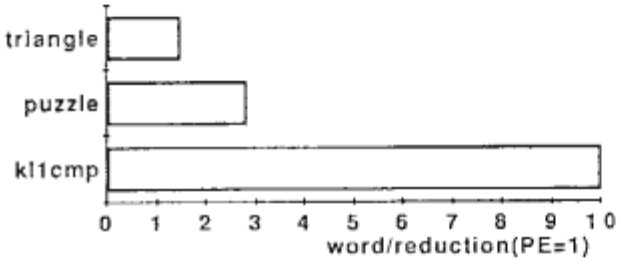
表 1: データと管理方法

KL1 のデータ	変数セル	(1 ワード)	一括 GC
	ベクタ	(n ワード)	一括 GC
	リスト	(2 ワード)	一括 GC
	コード		現在、GC の対象外
制御用データ	ゴールレコード	(32 ワード)	フリーリスト管理、及び一括 GC
	サスペンションレコード	(2 ワード)	フリーリスト管理、及び一括 GC
	サスペンションフラグ	(1 ワード)	一括 GC
	通信レコード	(2 ワード)	フリーリスト管理、及び一括 GC

表 2: ベンチマークのプログラム規模と性能

	PE=1		PE=8
	sec	RPS	speedup
puzzle	606.02	1891	7.5
klicmp	126.83	1508	5.9
triangle	365.15	1825	6.4

図 1: ベンチマークのメモリ消費スピード



- トライアングル問題: 大きさは 15 ピン (三手後より)。 (triangle[11])

  Prolog プログラムを KL1 プログラムに変換したもので、典型的な OR 並列型 探索。

4 メモリ消費特性

メモリ消費特性を知るために並列コピーイング GC が起動された時点でメモリの使用状況をモニタし、以下のデータを収集した。

4.1 メモリ消費スピードと実行性能

各ベンチマークの台数プロセッサでの 1 リダクション当たりのメモリ消費スピードを図1に示す。この表より、メモリ消費スピードは、ベンチマークに依存し、特に klicmp のようにプログラム中でベクタを多く使用するプログラムではメモリ消費スピードが速い。[3]

[3]本処理系では、MRB[4] によるベクタの再利用は行っていない。

図 2: 実行性能対メモリ消費スピード

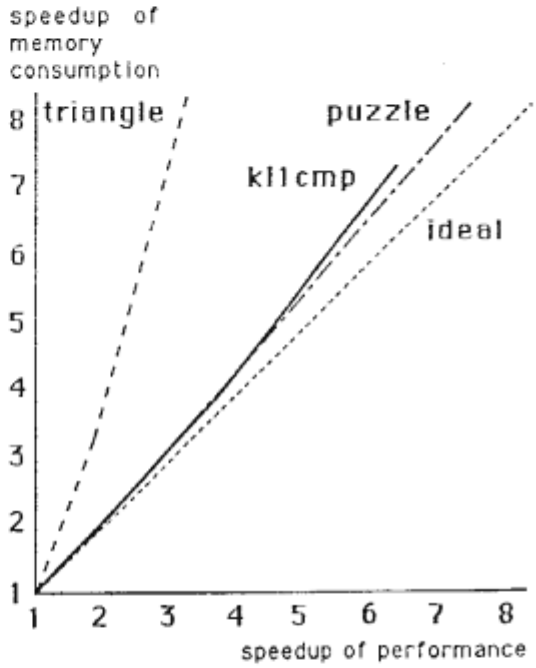


図2にプロセッサ台数を変え実行性能を向上させた場合のメモリ消費スピードの変化を示す。この図は、横軸がプロセッサ 1 台に対しての性能向上を示し、縦軸がプロセッサ 1 台に対してのメモリ消費スピードの増加を示している。この図より、triangle を除くベンチマークでは、性能向上とメモリ消費スピードの比は、ほぼ 1:1 であることが分かる。ここで、メモリ消費スピードが性能向上に比べ若干大きい理由は、並列実行によるサスペンションの増加によると考えられる。しかし、これらの増加によるメモリ消費の増加は、それほど大きくないといえる。

一方、triangle の場合は、メモリ消費スピードが性能向上に比べ著しく増加している。この理由は、ゴールレコードによるメモリ消費量の増加によると考えられる。triangle は典型的 OR 並列型プログラムで、プログラムの実行は、探索空間を広げながら、計算木の枝をほぼ独立に実行する。プロセッサが 1 台の場合は、これらの枝を左から右へ順次実行し、複数プロ

表3: リダクション当たりの計算木を広げる割合

	kl1cmp	puzzle	triangle
計算木を広げる割合	0.33	0.17	0.85

セッサの場合は、同時に別の枝を実行していく。しかし、プロセッサ台数が増加するに伴いプロセッサ当たりの使用可能メモリ量は見掛け上減少し、GCの起動される間隔は狭まる。つまり、プロセッサが1台のとき、1回のGC間隔で複数の計算木をゴールレコードを再利用しながら実行できていたのに対し、複数プロセッサでは、各プロセッサで各計算木を実行するために、ゴールレコードを各プロセッサで割り当てることが必要になる。さらに1回のGC間隔が狭まるために複数の計算木間でのゴールレコードの再利用が し難くなっている。(GCによりゴールレコードのフリーリストを共有メモリに開放しているため。)

表3に各ベンチマークのリダクション当たりの計算木を広げる割合[6]を示す。表よりtriangleは、他のベンチマークに比べ非常に大きな割合で計算木を広げていることが分かる。

4.2 アクティブセル

図3に各ベンチマークのプロセッサ台数毎のアクティブセルの時間変化を示す。ここで横軸の時間の単位は、プログラム実行時のGC起動回数を用いており、実際の実行時間と対応していない。例えば、プロセッサ1台と8台ではGC起動間隔時間は異なるが同じスケールとして表示している。実際の処理時間は、横棒グラフで示す。縦軸は、GC後のアクティブセルの総量を示している。アクティブセルの測定は、実並列処理系を用いて測定しているためタイミングの影響でバラツキが発生する。ここでは、それぞれ3回データを収集し表示した。

この図より、3つのことが分かる。第一は、プロセッサ台数が増加するに従ってアクティブセルの総量が増加する傾向にあることである。アクティブセルの総量が増加する理由は、各プロセッサが同時に計算木の別の枝を実行することにより、プロセッサ台数分だけアクティブセルが増加するためと考えられる。つまり、システム全体からみると計算木をbreadth-firstに実行するために計算途中の状態を多く保持しなければならないためである。

アクティブセルの変化の様子をさらに詳しく調べるためにアクティブセルの内訳の変化を図4に示す。図は、横軸がプロセッサ台数の変化を表し、縦軸がヒープとゴールレコードの変化を、プロセッサ台数1のときとの比で示している。ここで、ヒープとはサスペンションレコードによるアクティブなセルをも含め、ゴールレコード以外のアクティブなセル量を示している。図5は、1ゴールあたりのアクティブセル数の変化を示したものである。

図5: ゴールレコードとアクティブセルの比

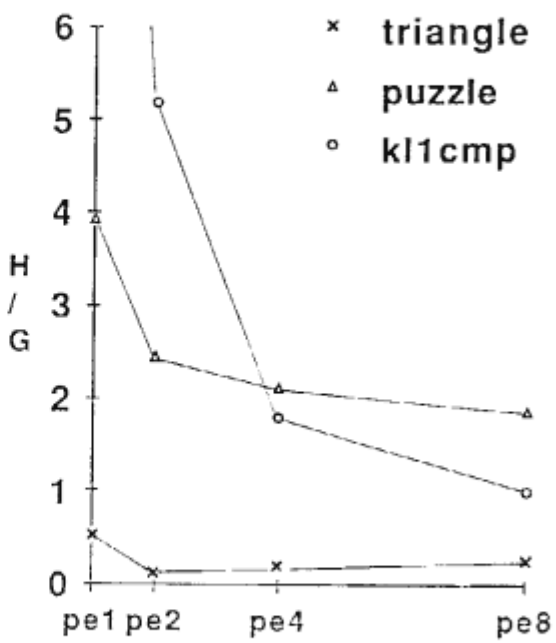


図4と図5より、ヒープの増加率がゴールレコードに比べ低いことが分かる。ヒープの増加率が低い理由は、プロセッサ数が増加するに伴い計算木をbreadth-firstに実行することになり、ゴール間のデータの共有率が増加しているためである。

第二は、アクティブセルの推移のパターンはプロセッサ台数の増加に関わらず、ほぼ一定であることである。プロセッサ台数が増加してもアクティブセルの推移パターンが変わらない理由は、各プロセッサが単一プロセッサの場合と同様に、各々の枝をdepth-firstに行っており、全体はその総和であるためである。

第三は、測定のバラツキはプロセッサ台数が増加するに従って大きくなっていく傾向にあることである。プロセッサ台数が増加するに従ってバラツキが大きくなる理由は、プロセッサ台数が増加するに従って負荷バランスが難しくなっていく様子を示している。つまり、アクティブセル量はプロセッサ台数の増加に従い増加し、負荷のバランスが旨くいっているときは増加する。負荷のバランスが良くない場合は、実質上プロセッサ台数が減少していることになり、アクティブセル量は減少する。

5 並列コピーイングGCの評価

5.1 GC性能と台数効果

GCの影響は、与えるメモリ量により異なる。ここでは、メモリ管理で述べたように、メモリ量を320Kワードとし、並列コピーイングGCの評価を行った。

図 3: アクティブセルの時間変化

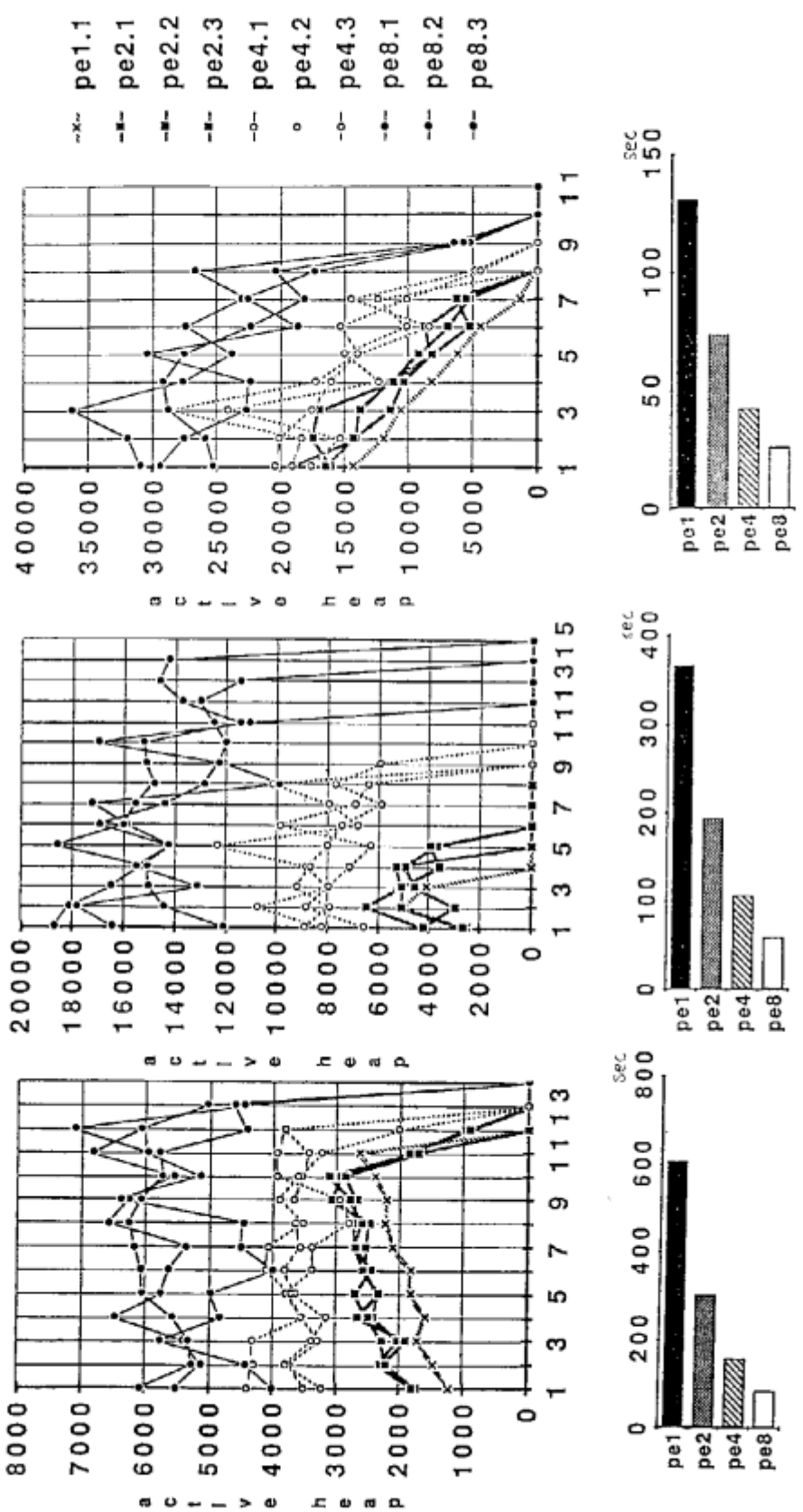


図 4: アクティブセルの台数変化

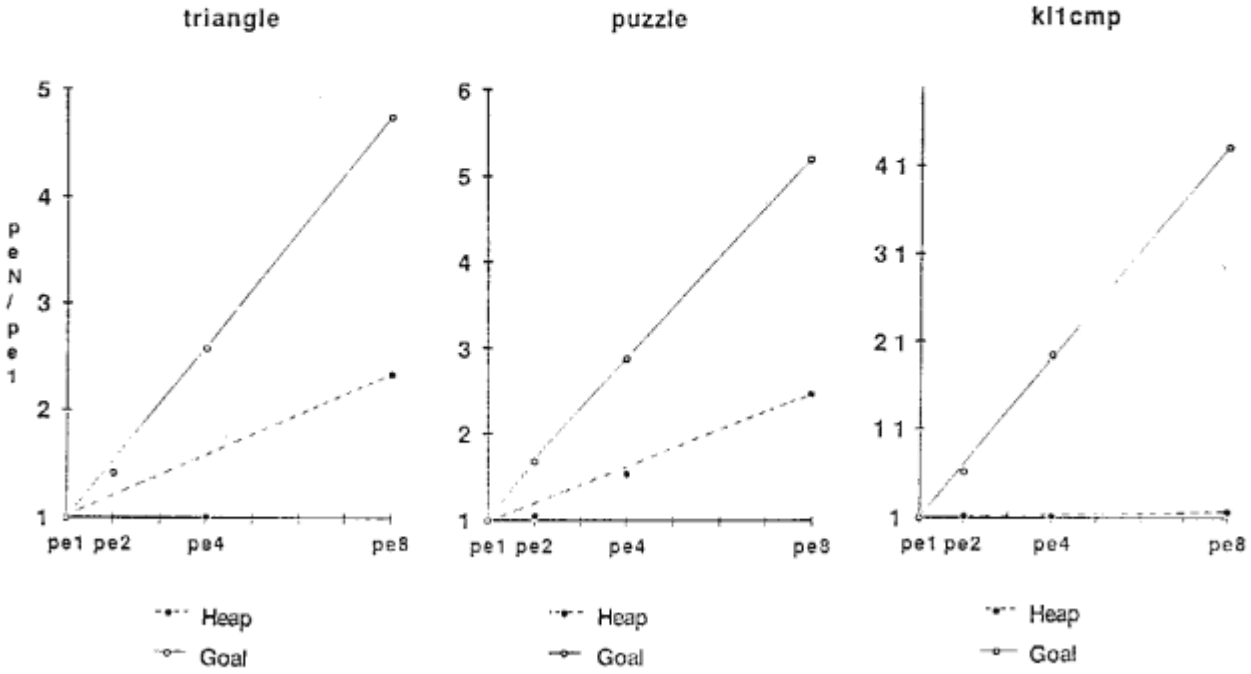


表 4: 単体 GC 性能 (copyword/sec)

	kl1cmp	puzzle	triangle
単体 GC 性能	14.8K	14.4K	15.1K

このメモリ量では、各ベンチマークとも10回程度GCが発生する。表4にプロセッサ1台の場合のGC性能を示す。ここで、GC性能は、プロセッサ1台の時のアクティブセルのコピースピード(copyword/sec)を示している。表より、GC性能はベンチマークに依存せず、ほぼ一定であることがわかる。

図6に各ベンチマークのGC時間を含めない場合と含めた場合との台数効果の変化を示す。ここで、GC時間を含めない場合とは、KL1処理系のGC以外の要因で決まる性能であり、メモリ量を十分大きく与えた場合の台数効果の変化に対応する。

図6より、puzzleやtriangleではGCによる影響はほとんど無いことがわかる。しかし、kl1cmpは、プロセッサ台数が増すに従ってGCの影響が大きく現れている。GCコストが増加する理由は、2つ考えられる。第一は、プロセッサ台数が増加するに従いアクティブセルが増加し、コピーすべきセル量自身が増加しているためで、この増加の様子は前節のメモリ消費特性で示した。第二は、GC自身の性能の並列化効果が、処理系の並列化効果に比較して小さいためと考えられる。以下では、後者の評価として、コピーイングGCの並列実行方式の評価を行う。

図7に上記のベンチマークのGC時のコピースピード(GC性能)の台数変化を示す。ここで、triangleの台数効果がプロセッサ台数以上である理由は、triangleのメモリ消費特性にある。つまり、前節で示したように、triangleはゴールレコード当たりのヒープの使用率が少ないため、アクティブセルのコピーはほとんどがゴールレコードとなる。一方、ゴールレコードのコピーでは引数個数分のコピーしか行わないため、見掛けのコピー量(コピー後のアクティブセル数)が増大する。triangle以外のベンチマーク(kl1cmp,puzzle)では、GC性能の台数効果は、リダクション性能の台数効果に比較して半分近くである。以下で、GC性能の台数効果を妨げているオーバヘッドについて考察する。

5.2 オーバヘッド

並列コピーイングGCの実現において、オーバヘッドとなる要因は、(1) リダクション実行フェーズとGCフェーズ間で、全プロセッサを同期し、切り替える操作、(2) コピーイングGCを並列に行うためのコ

図 6: speedup

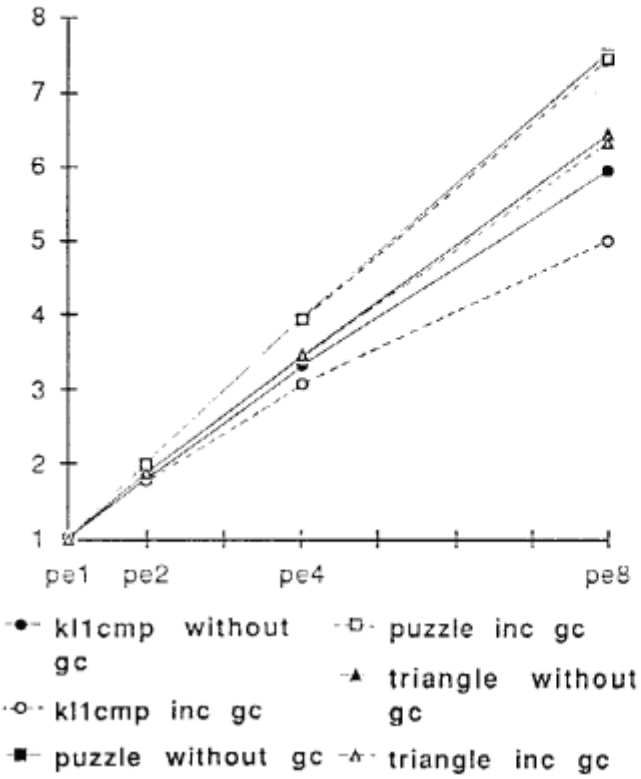


図 7: speedup-GC

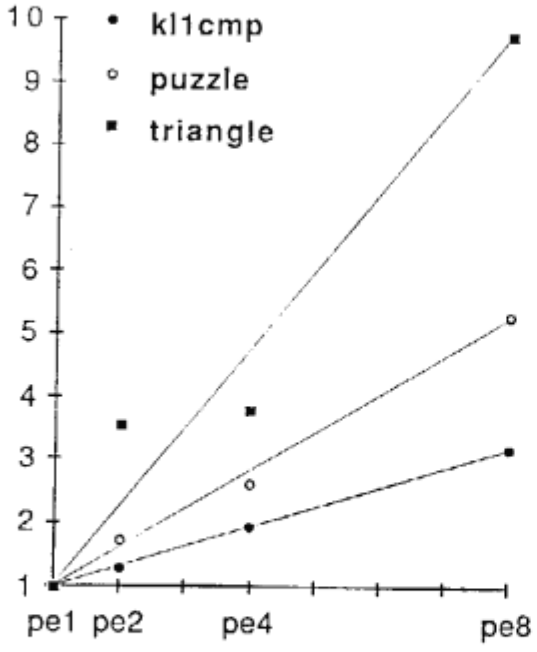


表 5: 排他操作のオーバヘッド

		pe2	pe4	pe8
puzzle	(gc/red)	0.4%	0.7%	1.2%
	(lock in red)	6.8%	6.5%	6.2%
	(lock in gc)	35.3%	29.1%	19.2%
kl1cmp	(gc/red)	5.6%	10.3%	18.2%
	(lock in red)	6.2%	6.3%	5.8%
	(lock in gc)	4.8%	3.6%	3.0%
triangle	(gc/red)	0.2%	1.2%	2.0%
	(lock in red)	9.2%	9.5%	8.9%
	(lock in gc)	19.7%	23.4%	13.7%

ビーするメモリに対しての排他操作、(3) コピーイング GC 時の負荷バランス等がある。

1. 同期

フェーズの切り替え時間は、プロセッサ間通信のレスポンスに対応し、プロセッサ間通信はリダクションの切れ目で通信要求をチェックするため、1 リダクション時間程度の誤差で実現できると予想される。実際、1Krps(reduction per sec) のベンチマークで 1msec 程度であった。この時間は、1 回の GC 時間において無視できるぐらい小さい。

2. 排他制御

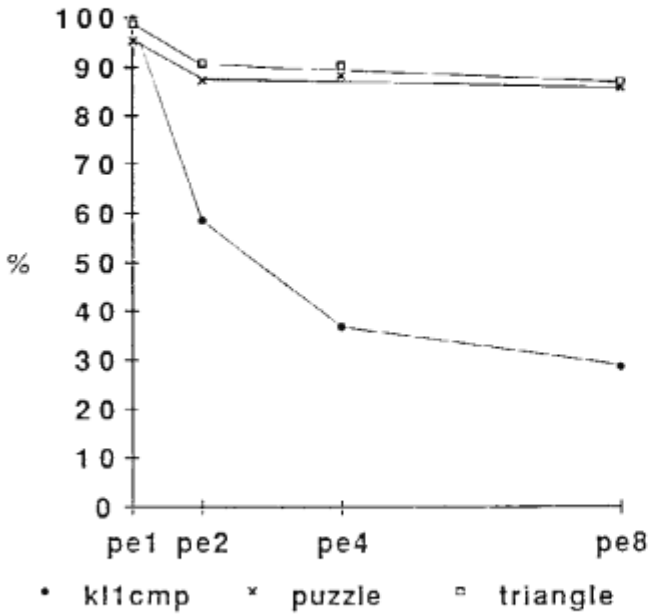
リダクション及び GC における排他操作のオーバヘッドを表5 に示す。表では、GC 時間がリダクション全体でどのくらいのオーバヘッドかを示すために、gc/red で GC 時間比を示す。また、lock in red はリダクション全体の排他操作の割合を示し、lock in gc は GC 中の排他操作の割合を示している。この表より、kl1cmp を除くベンチマークでは、排他操作のオーバヘッドはリダクション時に比較して GC 時のほうが大きいことがわかる。kl1cmp で、排他操作のオーバヘッドが少なく現れている理由は、GC 稼働率の影響である。(GC 稼働率は事項で述べる。)

しかし、GC 自身のオーバヘッドが小さいので全体からみるとそのオーバヘッドは小さいと言える。また、排他操作のオーバヘッドは、マーキングするアクティブセル量に比例し、プロセッサ台数が増加したときのアクティブセル量の増加はプロセッサ台数の増加より小さいので 1 プロセッサ当たりのマーキングするアクティブセル量は減少し、そのオーバヘッドは減少する傾向にある。

3. 負荷バランス

GC 時の稼働率を図8に示す。この図より、kl1cmp を除くベンチマークでは、GC 時の稼働率は、十分高いと言える。ただし、kl1cmp では、稼働率

図 8: GC の稼働率



が非常に低い。この理由は、ベンチマークである kl1cmp は、コンパイル対象であるプログラムをデータとして静的に与えており、このサイズが非常に大きく、GC 時にはこのデータが 1 つのゴール (つまり、1 プロセッサ) からコピーされるためである。つまり、GC の負荷分散の粒度をゴール単位としているため、大きなデータを抱えるゴールが存在すると負荷のバランスが困難になる。特に、GC は実行時間が非常に短く、ゴール当たりのアクティブセル量が大きく異なる場合、その影響を受け易い。

6 おわりに

実際の密結合マルチプロセッサシステム上の KL1 並列処理系を用いて、KL1 の並列実行時のメモリ消費特性を評価するとともに、並列コピーイング GC の評価を行った。

メモリ消費特性の評価では、性能向上とメモリ消費スピードの比は、ほぼ 1:1 であることがわかった。ただし、探索空間を爆発的に広げるようなプログラムでは、ゴールレコード再利用の効果が減少してしまいメモリ消費スピードが増加する。また、アクティブセルは、プロセッサ台数の増加に伴い増加する傾向にあった。しかし、その要因は主にゴールの増加であり、共有データは共有率が高まる傾向にあった。

並列コピーイング GC の評価では、GC の台数効果は、性能の台数効果に比べ低いことが分かった。また、GC のオーバヘッドは、同期に関しては小さく、排他制御に関しては大きかった。負荷バランスは、分散の粒度 (ゴール) が大きく影響した結果となった。今後は、排他制御のオーバヘッドを削減することと負荷バランスの粒度の検討を行って行く予定である。

謝辞

日頃御指導頂く、ICOT4 研、内田俊一室長に感謝します。また、貴重なコメントを頂いた ICOT、PIM グループの方々や沖電気 PIM グループの方々に感謝します。

参考文献

[1] A.Goto et al, Overview of the Parallel Inference Machine Architecture (PIM), Proc. of the International Conference On Fifth Generation Computing Systems 1988, pp.208-229, Tokyo, Japan, (1988-11).

[2] T.Chikayama et al, Overview of the Parallel Inference Machine Operating System (PIMOS), Proc. of the International Conference On Fifth Generation Computing Systems 1988, pp.230-251, Tokyo, Japan, (1988-11).

[3] Y.Kimura and T.Chikayama, An Abstract KL1 Machine and its Instruction Set, Proceedings of the 1987 Symposium on Logic Programming, pp.468-477, (1987).

[4] M.Sato et al, KL1 Execution Model for PIM Cluster with Shared Memory, Proceedings of the Fourth International Conference on Logic Programming, pp.338-355, (1987).

[5] M.Sato and A. Goto, Evaluation of the KL1 Parallel System on a Shared Memory Multiprocessor, Proceedings of IFIP Working Conference on Parallel Processing, Pisa, Italy, (1988-4).

[6] 佐藤、後藤, 密結合マルチプロセッサでの KL1 並列処理系の評価. 信学データフローワークショップ 1987, pp.95-102, (1987-10).

[7] K.Ueda, Guarded Horn Clauses, ICOT TR-103.

[8] T. Chikayama and Y. Kimura, Multiple Reference Management in Flat GHC, Proceedings of the Fourth International Conference on Logic Programming, pp.276-293, (1987).

[9] 木村他, KL1 の多重参照ビットによる GC 方式について, 信学データフローワークショップ 1987, pp.215-222, (1987-10).

[10] Symmetry, Guide To Parallel Programming.

[11] E.Tick, Performance of Parallel Logic Programming Architectures, ICOT TR-421.