参照カウンタ法を用いた並列ゴミ集め処理

4N－3

荻原拓也　君島有紀　田中良夫　中西正和
慶応義塾大学大学院理工学研究科

1. はじめに

計算機の性質は過去のものとは大きく変わり，大容量のメモリをのせ，かつ複数のCPUを積んだ計算機が多く出回るようになった．

しかし，そのような状況においてもLispなどのリスト処理言語においてゴミ集め処理 (Garbage Collection以下GC) は，必須のものである．計算機の性質が変わるにつれて，GCの手法にも新たな思考が必要になってくる．近年，逐次計算機上で研究されているmark&sweepなどの手法を改良し分散環境で研究されるようにもなった．その中でも，脚光を浴びているのが，参照カウンタ法である．

本論文では，汎用機では不向きであるとされる参照カウンタ法を用いたゴミ集め処理を実装し，様々な角度から検討・考察を行なう．また，複数のゴミ集め処理プロセスを起動しその効率化をはかる．

2. GC

- Mark and Sweep法
- Copying法
- 参照カウンタ法

3. 参照カウンタ法

参照カウンタ法はリスト処理言語が考案された当初から存在する基本的なGCアルゴリズムの一つである。現段階では、Mark and Sweep法やCopying法を応用したものがGCアルゴリズムの主流であり参照カウンタ法は実用的ではないとされている。

3.1 アルゴリズム

参照カウンタ法

cellごとに自分を指しているリンクの数を設け、リスト構造を変更するたびに参照カウンタの増減を行なう。参照カウンタが0のものが死んでいるセルであり、参照カウンタが0になった時点で即座にそのcellがフリーセルとして回収される。図1にその様子を示す．

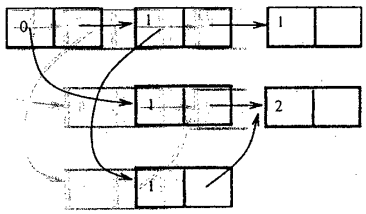

図1: カウンタの様子

3.2 参照カウンタ法の利点・欠点

- 利点
  - 即時回収性に優れている
  - リスト処理の中断時間がほとんどない
- 欠点
  - リスト処理による負荷が大きい
  - 循環リストを回収できない
  - 領域の確保をしなければならない
  - 長く連続したcellの回収による負荷

以上のことが，主な利点及び欠点として挙げられる．

4. Deffered Reference Counting GC

Deffered Reference Counting GC[2] は 1976 年にDeutschとBobrowによって発表された参照カウンタ法を改良したGCである．

4.1 改良点

本来の参照カウンタ法ではリスト処理が行なわれる度にカウンタの操作をした．また、スタックなどの

Parallel Garbage Collection Using Refference Counting Technique
Takuya OGIHARA
Yuki KIMISHIMA
Yoshio TANAKA
Masakazu NAKANISI
Department of Computer Science, Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kanagawa Pref., 223, Japan

操作についても同様に行なった. 関数の呼び出しなどが繰り返し起こっているような処理では、スタックの書き換えなどの度に頻繁にカウンタを操作しなければならず、この操作によるオーバーヘッドは無視できない. よって、ルートの管理は別にすることによってオーバーヘッドはかなり軽減されることになる. この点を改良し、提案されたのが Deffered Reference Counting GC(遅延参照カウンタ法) である。

4.2 手法

Deffered Reference Counting GC で用いるカウンタ操作は **allocate, create, destroy** の 3 種である.

4.3 テーブル管理

- **ZCT(Zero Count Table)**
- **MRT(Multi Reference Table)**
- **VRT(Variable Refference Table)**

4.4 euzak lisp 上での実現

この手法を用いて euzak リスプ上に実装し, 同様に通常の参照カウンタ法も実装した. これらを用いて実験を行なった結果は以下のとおりであった. Bench Mark として,boyer・ackerman・ハノイの塔を選んだ.

	boyer	ackerman	ハノイの塔
参照カウンタ法	356.51	302.13	16.29
遅延参照カウンタ法	326.85	285.10	15.64
Mark and Sweep 法	117.83	115.72	5.10
Copying 法	116.79	107.94	5.13
世代別 GC[5]	99.72	91.55	4.58
並列 GC	117.39	120.42	5.70
Partial Marking[6]	114.41	115.31	5.47
Incremental GC	173.76	153.41	10.41

表 1: **実験::他の GC 手法との全処理時間の比較**

5. Deffered Reference Counting GC の並列化

4.4 で示したとうり,Deffered Reference Counting GC をそのまま使うことはかえって,大きなオーバーヘッドを生むことになる. この負荷により, リスト処理のプロセスとテーブル操作とは全くべつの処理であるにもかかわらず,参照カウンタ法の利点であるリスト処理プロセスが止まらないという特性を,無くしてしまう.

- 改良点
  - allocate・create・destroy の三つの操作から生み出されるテーブル操作を mutator と切り離すことにより, リスト処理プロセスのオーバーヘッドを軽減する.

そこで, Deffered Reference Counting GC の並列化をおこなった [4].

	boyer	ackerman	ハノイの塔
参照カウンタ法	356.51	302.13	16.29
遅延参照カウンタ法	326.85	285.10	15.64
並列遅延参照カウンタ法	284.51	251.31	15.49

表 2: **参照カウンタを用いた GC の全処理時間の比較**

これらの結果からもわかるようにある程度の時間短縮は可能ではある. しかし, 生きているセルが多くなってくるとテーブル操作が多くなるので, リスト処理プロセスにかかる時間よりも GC 処理プロセスにかかる時間の方が多くなってしまい, リスト処理が超時間停止する自体が起こる. この解決策として複数の GC 処理プロセスを動かすことが考えられる.

6. 今後の課題と問題点

リスト処理プロセスにかかる負荷は,非常におおきく, このオーバーヘッドを削減しない限り, 参照カウンタ法を用いた GC は実用することは難しい. しかし, ネットワーク分散処理を行なうような処理系では通信のオーバーヘッドが大きいため,実用の可能性はある. 現在ネットワーク分散 LISP 上に実装している途中であり, 実装後分散環境における GC に対しての追実験を行なう必要がある.

参考文献

[1] J. Cohen. Garbage Collection of Linked Data Structures. In *ACM Computing Surveys No.3*, Vol. 13, September 1981.

[2] L. Peter Deutsch and Daniel G. Bobrow. An efficient, incremental, automatic garbage collector. In *Communication of the ACM*, Vol. 19, September 1976.

[3] 中西正和. Lisp 入門 ーシステムとプログラミングー. コンピュータサイエンス大学講座 1. 近代科学社, 1985.

[4] 荻原拓也. 参照カウンタ法を用いた並列ゴミ集め処理. 情報処理学会全国大会, September 1995.

[5] 高岡詠子. Adaptive garbage collection の提案及び実験. 電子情報通信学会論文誌, Vol. 9, No. J77-D-I, September 1994.

[6] Y. Tanaka, S. Matsui, A. Maeda, and M. Nakanishi. Partial Marking GC. In *Proceedings of International Conference on CONPAR 94 - VAPP VI*, September 1994.