Adaptive Garbage Collection 実現のための実験および実装

4V－8

田中 詠子

慶応義塾大学理工学研究科

1. はじめに

リスト処理言語では動的なメモリ管理が不可欠であり様々な手法が提案されているが，その中にオブジェクトを寿命に応じて区別し，あらかじめ設定されたある「しきい値」に応じて不必要なオブジェクトの回収 (garbage collection, GC) を効率良く行う手法がある．このしきい値をいくつに設定すれば最も効率良くGCを行うことができるかという問題がある．最も効率の良くなるしきい値は実行するアプリケーションによって異なる．**Adaptive Garbage Collection(AGC)** はメモリの消費状態に応じてしきい値を動的に調節し，様々なアプリケーションに対し効率良く動作するGCである．AGC実現に向けて様々な実験を行った結果，効率良くAGCを実現するための有力なパラメータを発見することができた．本稿では，AGCの実装報告と，このパラメータの有効な使い方を検討する．

2. 世代別GC

1983年に，長い間生き続けたオブジェクトは半永久的であるということが発見された[1]．そして，1984年にはGCにオブジェクトの**寿命**の概念を取り入れた**generation scavenging GC**[4]が提案された．generation scavenging GCはオブジェクトをその寿命によっていくつかの世代に分ける効果的な手法である．この手法では，アドレススペースを**生成領域**，2つの**短寿命領域**，そして**長寿命領域**に分けている．オブジェクトの生成は生成領域に対して行い，生成領域がすべて消費されるとGCが始まる．生成領域および短寿命領域中の生きているオブジェクトはもう1つの短寿命領域にコピーされる．この時短寿命領域のGCを何回か生き残ったオブジェクトは長寿命であると判断され長寿命領域に移される．これを**tenuring**という．長寿命領域に移されたオブジェクトは通常のGCの対象外となる．objectが短寿命領域のGCを何回生き残れば tenuring されるかを表す数字を **advancement threshold(AT)** という．advancement threshold の値によってはその object が tenuring したすぐ後に死んでしまって長寿命領域を無駄にすることがある．このような object のことを **tenured garbage(TG)** という．tenured garbage を出さないためには advancement threshold をいくつに設定するかという問題が残される．1989年には，Wilson らによって generation scavenging GC を改良した **opportunistic garbage collector(OGC)**[5]が提案された．ここでは，最適なATは1から2の間であることを指摘している．generation scavenging GC がATを「通常のGCを生きのびた回数」と定義しているのに対し，ここでは生成領域中にある境界（**watermark**）を設け，その境界を移動することにより自由にATを平均的に1から2の範囲に設定することができる．

3. Adaptive Garbage Collection

ATを設定する際には，次のことを考慮に入れる必要がある．それは，アプリケーションによってセル消費のペースが違うということである[3]．セル消費のペースとはセルが生成されてからどれくらいの間隔で死ぬかということである．セル消費のペースが違うとセルが長寿命であるための条件が変わってくる．アプリケーションに応じて，またはセル消費のペースに応じて動的にATを決定することによって，より効率の良いGCを行うことができる．AGC[2]はセルの消費のペースに応じたGCである．AGCはOGC

Design and Experiment of Adaptive Garbage Collection

Eiko TANAKA

Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kanagawa, 223, Japan

のアルゴリズムをベースにしており，アプリケーション実行時にOGCにおけるwatermarkを動的に動かしてATを調整してGCを行うことにより，TGを最小限に抑え，長寿命領域を有効に使用する．

セルの消費状態に対してATをどのように調整するべきかを知るためにATを固定してさまざまな実験を行い，ATの最適値と生成領域から長寿命領域へ生き残るセルの割合とに密接な関係があることがわかった．この割合を，AGCを実現するための有力なパラメータとして，**tenuring parameter (TP)** と名付けた[3]

4. 実装

AGCはSunワークステーション上で実装された．AGCの基本的なアルゴリズムを以下に示す．

- 各GCの終了時にTPを計算し前回のGCでのTPの値との差を求める．
- TPの変化が小さい時にはATの値は動かさない．これは，TPの変化が極めて小さい場合はATを調節する必要がないということである．
- TPが大きく増えた時にはATの値を上げる．
  TPが前の値と比べて大きく増えるということはTGの増加につながるので，ATの値は上げる．
- TPが大きく減った時にはATの値を下げる．
  TPが前の値と比べて大きく減るということはTGの減少につながるので，ATの値は下げる．

このアルゴリズムを図1に示す．TPの変化がある値以上である場合にATを変化させる．図におけるchanging degreeとは，その時のしきい値のことである．また，TPがしきい値以上に変化していた場合にはATの値をその変化に応じて増やしたり減らしたりする．deltaとは，その際にどの程度ATの値を変化させるかを示す値のことである．初期値を色々な値に設定して実験を行った結果、どの初期値で始めても実行終了時のATはほぼ同じ数値になった。TGの割合は、AGCを適応する前の割合（ATを固定した場合）より減少した。

5. 今後の課題

初期値，changing degreeをはじめ，AGCの評価にはさまざまな要因がからむので何に注目して実験を

```
diff = current TP − last TP ;
if      ( | diff | < changing degree)
            do not move AT;
else if  ( diff ≧ changing degree)
            AT += delta;
else
            AT −= delta;
```

図 1: Algorithm of AGC

を行うかが重要となってくる．結果として期待されることは，AT固定時の最適値におけるTGの割合より，コピーコストが最適値により近い状態でTGが減少することである．現在の時点では長寿命領域を使い切ると実行をやめているので，今後は大きなアプリケーションを動かして，コピーコストをできるだけ抑えつつ長寿命領域のGCを減らすことができるかという視点で実験を行うことが必要であろう．

参考文献

[1] Henry Lieberman and Carl Hewitt: "A Real-Time Garbage Collector Based on the Lifetimes of Objects", Comm. ACM, Vol. 26, No. 6, pp.419–429(June 1983).

[2] 高岡 詠子, 中西 正和: "アダプティヴ・ガーベッジ・コレクションの実現のための実験報告", 情処学記号処理研報, 93-SYM-67-1, No. 8, pp.1–8(Jan. 1993).

[3] 田中 詠子, 田中 良夫, 中西 正和: "Adaptive Garbage Collectionの提案および実験", 電子情報通信学会論文誌 (Sep. 1994) .

[4] David Ungar: "Generation Scavenging:A Non-disruptive High Performance Storage Reclamation Algorithm", ACM SIGPLAN Notices, Vol. 19, No. 5, pp.157–167(May 1984).

[5] Paul R. Wilson and Thomas G. Moher: "Design of the Opportunistic Garbage Collector", OOPSLA '89 Proceedings, pp.23–35(Oct. 1989).