発表概要

世代管理を保守的に行う世代別GCアルゴリズムの提案およびRubyへの実装と評価

五百蔵 重典† 西尾 孝典† 野木 兼六†

ヒープに確保された使用されていないオブジェクトを自動的に回収するガーベジコレクション機能（以下，GC）は，プログラマのメモリ管理の負担を軽減するための重要な機能である．GC アルゴリズムの中には，GC で生き残った古いオブジェクトは若いオブジェクトよりも長く生き残るという経験則を利用して，新しく作成されたオブジェクトのみを GC の対象とすることで，処理速度を向上させる世代別 GC アルゴリズムがある．しかし世代別 GC アルゴリズムは，古い領域から新しい領域へのリンクを検出する処理（以下，ライトバリア）が必要である．そして，そのライトバリアは実行時間のオーバヘッドになること，処理系を実装するために必要な箇所にライトバリアを配置することは煩雑であることから，世代別 GC アルゴリズムを効率良く実装することは難しいのが現状である．そこで本発表では，先頭側の領域を old 領域，末尾側の領域を new 領域に分断し，old 領域に属しているオブジェクトはすべて古いオブジェクトと見なす新しい世代別 GC アルゴリズムを提案する．本発表のアルゴリズムでは，old 領域では new 領域へのポインタが存在するかを検査し，new 領域では GC を行う．本発表のアルゴリズムの特徴として，ライトバリアが必要ない，メジャーコレクションとマイナーコレクションが一体化している，および生きているオブジェクトの移動を必要としないなどがあげられる．本発表では，提案アルゴリズムをオブジェクト指向スクリプト言語であり，マーク&スイープ型の保守的 GC を備える Ruby 上に実装した結果，全体の処理時間は最高 90.8%に短縮でき，1 回の GC 時間では最高 70.8%に短縮することができたことを示す．

Proposal of Generation GC Algorithm Managing Generation Conservatively, and Implementation and Evaluation in Ruby

Shigenori Ioroi,† Takanori Nishio† and Kenroku Nogi†

Garbage collection (GC) algorithms which collect unused objects in heap memory automatically are important technology, because there free programmer from memory management. There are Generation GC algorithms which try to collect unused object in only heap area are which contain new objects (new-area), using the experience that many younger objects tend to be unused object soon after allocate, and used objects after GC tend to keep being used, therefore Generation GC algorithms improvement execution time. But, Generation GC algorithms need program code which search for link from old-area to new-area (hereafter, write-barrier code). Then write-barrier code has over-head at program execution time and needs to set many write-barriers appropriately in language processor. Therefore we are difficult to implement Generation GC algorithm. We propose new Generation GC algorithm which we assume that head-side of heap area is old-area, and tail-side of heap area is new-area. The framework of this algorithm searches pointers from old-object to new-object in old-area, and applies to GC in new-are. The character of this algorithm don't need write-barrier, a distinction of combines major collection and minor collection is a little, and don't move old-objects in old-area. We implement this algorithm in object-oriented script language Ruby which have conservative mark & sweep GC algorithm. We show that execution time is 90.8% at a maximum, and one GC time is 70.8% than original ruby.

（平成 19 年 1 月 18 日発表）

† 神奈川工科大学情報学部情報工学科
Faculty of Information Technology, Department of Information and Computer Science, Kanagawa Institute of Technology