発表概要

回収を遅延して行うガーベッジコレクションの実時間化について

山室弥久† 森田綾子† 寺島元章†

使用済みになったデータオブジェクトの回収を遅延させて行う圧縮型ガーベッジコレクション（mark-and-compact GC）の実時間化とその評価について述べる．回収の遅延とは最新に作られたオブジェクトが存在する領域の回収処理を次回に延ばすことであり，比較的短い生存期間のオブジェクトを圧縮処理から除くことができる．さらに，世代別 GC に見られるように GC を経たオブジェクトを殿堂入りさせると，ある上下限で規定される生存期間を持つオブジェクトのみが圧縮処理の対象となり，圧縮処理の軽減が図られることが停止回収型の実装から示されている．この GC の実時間化では，write-barrier を使用する snapshot-at-the-beginning とオブジェクトの複製圧縮の各技法が使用されている．前者は，read 操作の多い純計算側の負荷を確実に減らすとともに，最新のオブジェクトが未処理で残るような遅延 GC にはむしろ効果的に機能する．後者は一度に圧縮するオブジェクトを移動先にその複製が作られる範囲に限定することで，圧縮処理中での GC の中断を可能にする．GC 回数の増加につれて，圧縮済オブジェクト群の領域と最新のオブジェクトを含む未処理のオブジェクト群の領域の間には 1 つの空き領域が生まれる．これが飢餓状態の回避に役立つ．評価実験は，GC と純計算側の mutator をコルーティン化して行った．そして，数種の Lisp プログラムの実行結果に基づく本 GC の評価を述べる．

An Incremental Garbage Collector Based on a Lazy Garbage Collection Scheme

Mitsuhisa Yamamuro,† Ayako Morita† and Motoaki Terashima†

The implementation and evaluation of an incremental garbage collector based on mark-and-compact garbage collection (GC) are presented that put collection of unused data objects off. Such lazy GC does not scavenge a space where new objects are allocated and put collection of unused objects of the space off for a one GC cycle, so that relatively short-lived objects are omitted from GC processing. In addition to this, the promotion that long-lived objects with GC experiences are omitted from future GC, as seen in generational GC, makes objects with a certain lifetime processed. The decrement of objects being compacted is shown by a prototype of stop-and-collect version. The incremental version uses two techniques, namely, snapshot-at-the-beginning using write-barrier and duplication of objects. The former is less costly in time for coordination with mutator that performs many read operations, and has good effects on the lazy GC where new objects remain unprocessed. The latter is to relocate objects in use stepwise so that both the relocated and original objects exists till the relocated objects are updated, and this enables the GC to break it process at any stage. According as the GC proceeds, a continuous empty space appears between the space of compacted objects and the space of unprocessed objects. This space is useful to avoid starvation. The analysis of the GC on its performance is done by implementing co-routine made of the GC and the mutator routines, and obtained from the execution of typical Lisp programs.

（平成 14 年 1 月 29 日発表）

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications