発表概要

多世代管理 GC の並行化について

新田 寛[†] 藤田智子[†] 寺島元章[†]

多世代管理を行う並行ガーベッジコレクション（GC）とその実装法について述べる．データオブジェクトの再配置を行わない GC は，効率的な並行化が可能である．これに対し，従来の再配置型の並行 GC は，GC と純計算の双方がヒープとルートを書き換えることから，複雑な同期処理が必要となるため，汎用機上での効率的な実装は困難であった．本 GC は，再配置型 GC でありながら，ヒープの書き込みに対する同期のみで並行化を可能としている．このため，従来のアルゴリズムに比べ，並行化にともなうオーバヘッドが非常に小さくなっている．同時に，高速な圧縮型 GC である便宜的 GC を機能拡張し，データオブジェクトの多世代管理を効果的に実現している．便宜的 GC には使用中データの局所化にともなうワーキングセット・サイズの縮小により，純計算時間が短縮されるという利点がある反面，ヒープが単調に消費されるという問題点もある．本 GC では，1 回目の GC 処理で生き残ったデータオブジェクトを，その後も定期的に GC 処理の対象にすることで，長期間に渡るオブジェクトへのきめ細かな管理を行う．こうした多世代管理によって，便宜的 GC の問題点とされたヒープ消費に関する非効率性を緩和している．本 GC を Common Lisp 準拠の処理系である PHL に実装し，汎用的な環境下で数種の Lisp プログラムを実行して評価を行った．

Concurrent Generational Garbage Collection

Hiroshi Nitta,[†] Tomoko Fujita[†] and Motoaki Terashima[†]

The design and implementation of multi-generational garbage collection based on both incremental and concurrent features are presented. The garbage collection (GC for short) that performs its task without data object relocation can be effectively implemented as concurrent one. On the other hand, the GC with data object relocation may rewrite roots and heap that list processor (i.e., mutator) also manipulates. Therefore, it seems to be difficult to implement such concurrent GC on general purpose machines effectively due to complex exclusive and mutual processing. Our concurrent GC with data object relocation has high performance by applying the exclusive and mutual processing to the heap only. Our GC may be regarded as the refinement of the so-called occasional GC that is fast mark–and–compact one. The occasional GC has a merit of making the processors totally run faster by means of working set reduction that results from the localization of data objects in use, though it monotonously consumes the heap as a demerit. Our GC has good effects on multi-generational heuristics that the occasional GC has never done. The data objects being alive through the current GC processing are subjected to process again at the GC two or three times after, which results in space economy. The analysis of the GC on its performance is done by using our experimental data obtained from the execution of typical Lisp compiled programs running on PHL, a dialect of Common Lisp.

(平成 12 年 8 月 3 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, University of Electro-Communications