4N－1

並列論理型言語処理系 KLIC への世代GC方式の導入

吉川 隆英, 近山 隆
東京大学 工学系研究科

1. はじめに

KLIC[1] は可搬性と効率性の両立を目標に構築された, 並列論理型言語 KL1 の並列処理系である.

KLIC では, 自動メモリ管理を行っておりガーベジコレクション(GC)の処理時間は, 通常, プログラム実行時間全体のうち無視できない時間をしめる. 従って, GCの性能は処理系の性能を左右する大きな要因の一つである.

現在のKLIC処理系は, 比較的短期間だけ必要とするような様々な制御用のデータ構造なども, プログラム中から明示的に利用する通常のデータと同じヒープ領域に割り付ける設計となっている. これは処理系の構造を単純にする効果を上げている反面, 短寿命のデータが非常に多くなり, GC機構に大きな負担となっている.

そこで, 本研究では, 様々なデータ構造を割り付けるヒープ領域を, 比較的長時間保持されているデータを格納する領域(旧世代領域)と, 最近割り付けたデータを格納する領域(新世代領域)とに分割する世代GC方式を KLIC に導入し, その効果を測定した.

2. 設計と実装.

KLIC 3.002 版に新世代2面, 旧世代1面のヒープ領域を持たせて世代GC方式を導入した.

旧世代のGCは行わず, 通常のデータ構造については, 旧世代から新世代へのポインタの箇所を覚えておいてGC時にそれを root set に含めるようにした.

制御用データ構造についても, 世代を跨ぐ箇所を表に登録することによってGCの手間の削減を図った.

An implementation and evaluation of generational garbage collection for KLIC
Takahide Yoshikawa and Takashi Chikayama
{taka, chikayama}@logos.t.u-tokyo.ac.jp
School of Engineering, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

3. 評価方法

効果の測定は, KLIC ランタイム付属のテストプログラムの実行時間を

(1) オリジナルのKLIC (Original)

(2) 世代GC方式を導入したKLIC (GGC)

(3) (2)で旧世代領域を使用しないもの(NO_GGC)

の3方式を比較することにより行った.

また, コンパイル後のコードサイズの比較も行った.

4. 測定結果

測定は JU-5 (Ultra SPARC IIi (270MHz), 128MB RAM, Solaris 2.6) を用いて行った.

4.1 実行時間の比較

KLIC ランタイムに含まれるテストプログラムのうち

hanoi.kl1	ハノイの塔, 17枚円盤を移す.
kkqueen.kl1	11個のクイーンを盤面に配置.
primes.kl1	10, 000以下の素数を生成.
life.kl1	ライフゲーム.

の実行時間のグラフを図1に示す.

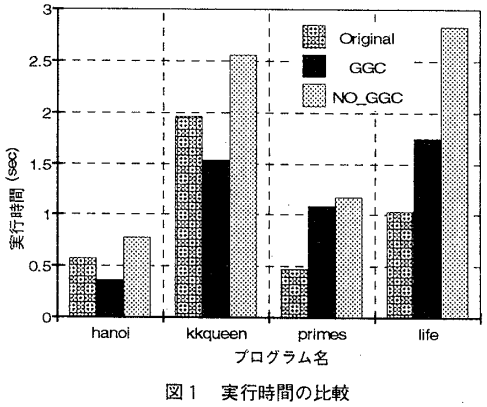

図1　実行時間の比較

Original と NO_GGC との差は, 世代GCを行うために必要な処理(世代を跨ぐポインタのチェックなど)に掛かる時間を示していると考えられる.

4.2 GCにかかる時間の比較

先ほどの4つのテストプログラムについて，GCに要する時間の比較を行った．その結果を図2に示す．

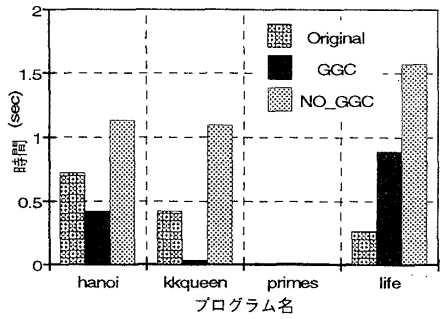

図2　GCに要する時間

4.3 コードサイズの比較

各テストプログラムをコンパイルした結果のオブジェクトのコードサイズは以下の図3のようになった．

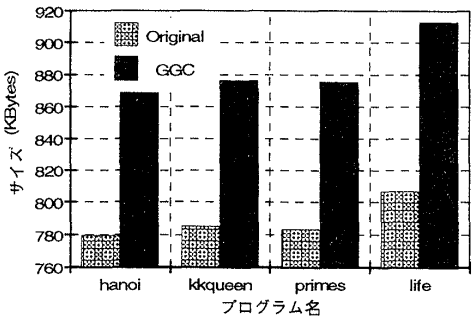

図3　オブジェクトコードサイズの比較

いずれの場合でも，10%程度大きくなってしまった，

4.4 評価

hanoi.kl1,kkqueen.kl1 は世代GCを導入することにより速度が向上していることが判る．この速度向上は主にGC時のコピーの手間削減によるものと考えられる．

primes.kl1 の実行時間は Original よりも劣っているが，旧世代を用いないものよりは良い結果がでている．これは，図2にあるように元々GCの処理が軽いプログラムであるために，世代GCによる効果は出ているが，世代を跨ぐポインタの検出などによるオーバーヘッドがその効果を上回ったためであると考えられる．

life.kl1 に関しては世代GCを導入することにより実行速度が低下している．これは life.kl1 で扱われるデータの中に一定時間経過後に消滅するというものが非常に多いためで，旧世代領域にコピーされてすぐに使われなくなるデータが非常に多くなってしまったためであると考えられる．

5. おわりに

並列論理型言語処理系 KLIC に世代GCを導入し，オリジナルの KLIC との間で，いくつかのテストプログラムについての実行時間の比較を行った．

その結果，一定時間おきに生成消滅を繰り返すデータを多量に作成したり，元々GC に時間を要さないようなプログラムを除けば，オリジナルの KLIC に対して数%～20%程度の速度向上が見込まれると言うことが示された．

今後は，旧世代領域のGC，ジェネリックオブジェクトの世代GCへの対応などを行っていく予定である．

参考文献

[1] T.Chikayama, T.Fujise,D.Sekita: "A Portable and Efficient Implementation of KL1", PLILP'94,1994.

[2] 関田大吾: "Inside KLIC", ftp://ftp.logos.t.u-tokyo.ac.jp/klic/contrib/tutorial/inside.tgz., 1998.

[3] Paul R Wilson: "Uniprocessor Garbage Collection Techniques", ftp://ftp.cs.utexas.edu/pub/garbage/bigsurv.ps , 1992.