汎用計算機上の KL1 処理系におけるメモリ管理とデッドロックの検出

平野喜芳, 中越靖行, 宮崎芳枝, 西崎慎一郎, 宮崎敏彦
(富士通ソーシアルサイエンスラボラトリ)　(ICOT)

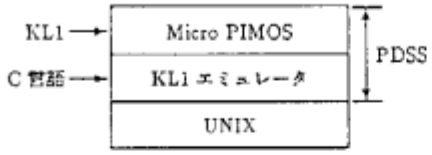
1　はじめに

本論文で取り上げる KL1 処理系は PDSS — PIMOS Development Support System[1] と呼ばれ、 PIMOS[2] の開発支援と KL1 アプリケーションの開発促進を目的として汎用計算機 (OS は UNIX) 上に開発された。

PIMOS は第5世代コンピュータの研究開発プロジェクトにおいて開発中の並列推論マシン用のオペレーティング・システムであり、ユーザーが消費するメモリ等の資源を管理し、入出力機能等をユーザーに提供する。

PDSS はエミュレータと Micro PIMOS の2つの部分から構成されている。エミュレータはコンパイラによりソースプログラムから変換された抽象機械命令を実行する部分で、C 言語により記述されている。抽象機械命令は KL1-B と呼ばれ、 WAM をベースとした命令系である。 Micro PIMOS は KL1 自身で記述されたシングルユーザー, シングルタスクの簡易 OS であり、会話型の実行環境を提供する。

Micro PIMOS に関しては本大会予稿 7H-6 の『KL1 による簡易 OS — Micro PIMOS』を参照していただくとして、本論文では KL1 エミュレータについてメモリ管理を中心にして述べる。



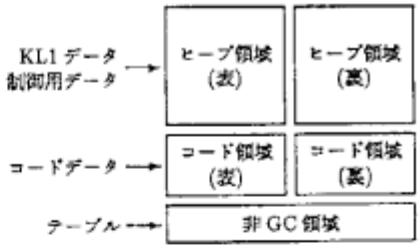
2　扱うデータの種類

PDSS のエミュレータでは大別して次に示す3種類のデータを扱っている。

- KL1 データ
  これは言語から見えるデータであり、アトム, 整数, リスト, ベクタ, ストリング, 変数がある。狭い意味のデータはこれを指す。
- コードデータ
  これは KL1 の抽象機械命令 (KL1-B) であり、モジュール単位になっている。
- 制御用データ
  これはエミュレータの実現方式上で作られるものであり、言語からは直接見えないデータである。制御用のデータにはゴールレコード, 冠雷レコード, サスペンションレコードがある。
- テーブル
  これにはアトムテーブルやモジュールテーブル等がある。このテーブルは名前と実体を対応させるものであり、ソフトで管理するほうが融通性が高いと思われるが、処理速度などの都合により処理系レベルで扱うようにした。

3　メモリマップ

PDSS のエミュレータのメモリ領域はヒープ領域, コード領域, 非 GC 領域の3つの部分に分けられている。このうちヒープ領域とコード領域はコピー方式の GC の為に表と裏の2つに分かれている。ヒープ領域には KL1 データと制御用データが、コード領域にはコードデータが、非 GC 領域にはテーブルが格納される。



コードデータはヒープ領域に入れることができるが、 GC の効率化のためにコード領域を別に作った。コードデータは一般の KL1 データと違いあまり変更されるものでなく、その寿命はかなり長いと言える。また、そのサイズはかなり大きなものとなる。従って、同じ領域に入れてしまうと GC を行う度に同じコードデータをコピーすることになる。別の領域に入れ、通常の GC の時にはコピーしないことにすれば GC の効率は良くなる。

一般にコード領域やヒープ領域の裏側は GC のとき以外には使われていないが、デバッグ情報の表示などで使用する一時的なデータを格納するために使用されることもある。

Memory Management and Deadlock Detection in a KL1 System

Kiyoshi HIRANO[1], Yasuyuki NAKAGOSHI[1], Yoshie MIYAZAKI[1], Shin'ichirou NISHIZAKI[1], Toshihiko MIYAZAKI[2]
1: Fujitsu SSL, 2: Institute for New Generation Compuler Technology

4 データへのメモリの割り付け

データへのメモリの割り付けは種類により次のように行う。

- KL1 データ, コードデータ
  これらのデータのための領域は必要に応じてヒープやコードの未使用領域の先頭から切り出すことで確保する。
- 制御用データ
  これはエミュレータの実現方式上作成しているデータであり、開放されるタイミングが分かっている。また、大きさも一定である。従って、フリーリストによる管理を行っている。フリーリストが空になったときには一定個数分の領域をヒープの未使用領域の先頭から切り出し、フリーリストに補充しておく。
- テーブル
  この領域は開放されることがないので、C 言語の malloc() で確保する。

5 ガベージ・コレクタ

PDSS のエミュレータでは GC としてコピー方式 GC を用いている。コピー方式 GC は使用可能なメモリの量が半分に減ってしまうが、GC の所用時間が生きているデータの量に比例し、アルゴリズムが簡単であるため採用した。PDSS でのコピー方式 GC の特徴としては、デレファレンスと MRB メンテナンスが挙げられる。デレファレンスを行うのは無駄なコピーを避けメモリの消費量を少なくすることと GC 後のプログラム実行中に行うデレファレンスを減らすことを目的とする。MRB — Multiple Refereuce Bit [3] は単一参照と多重参照を区別するためにポインタに付けられる 1bit の情報である。GC を行うと多重参照であったものが単一参照になる場合があり、MRB メンテナンス [4] ではこのようなときに MRB を OFF に戻す。これらの処理は全データを 1 回調べるだけで行うことができるので、通常の (これらを行わない) コピー方式 GC とあまり変わらない手間で実現することができる。

コピー方式 GC はリダクションの区切りにおいてのみ起動する。これは 1 リダクションで消費するメモリの量は一部の組込み述語を除いてそれほど多くないと考えられ、領域にある程度の余裕があるうちに GC を起動するようにしておけばオーバーフローの可能性は少なく、実用上は問題がないと思われるからである。メモリを大量に消費する可能性のある組込み述語はボディ部にだけ書くことができ、これらは各述語が自分で残りの領域の大きさを調べ、オーバーフローしないようにしている。これは、残りの領域が少ない時には組込み述語をすぐに実行しないで、一旦ゴールスタックに入れ、GC 後に実行することで実現できる。

6 デッドロックの検出

デッドロックを検出する目的はデッドロックに陥っている荘園を検知することとデバッグにある。PDSS では生成される全てのゴールレコードを 1 本のリストに繋ぐことによって、GC の際にデッドロックしたゴール及び荘園を検出している。

具体的には、GC の前後でゴールレコードの数が変化したか検査し、減っていればデッドロックしたゴールが有る可能性があるので、旧領域にある全ゴールレコードを調べ、そのゴールがデッドロックしているか調べる。ゴールがデッドロックしているといえる条件は以下で定義される。

- GC 時にコピーされない。
- そのゴールが所属する荘園がコピーされている。
- そのゴールが所属する荘園の状態が Aborted でない。

ここで荘園とは KL1 で OS を記述するために導入された言語機能であり、まとまったゴール群の実行制御を行う単位である。[5]

デッドロックが検出されると、そのゴールが属する荘園のレポートストリームにデッドロックのメッセージが流される。OS はこのメッセージを受け取ると、それに対処するための処理 (荘園の強制終了など) を行う。

7 今後の方向

今後の課題として GC の効率化が挙げられ、ジェネレーション GC の採用等が考えられる。KL1 で記述した本格的な OS のように大きなプログラムを動かすようになるとあまり変更されないデータが多く作られると考えられる。例えば Micro PIMOS のパーザがオペレータのテーブルを持っているようにこれらはほとんど変更されることがなく、寿命もかなり長いと言える。このようなデータもコピー方式 GC では GC の度にコピーされ、効率を落とす原因となる。そこで、コードデータを別の領域に分けたように、データを格納する領域をその寿命ごとに複数個用意し、通常は寿命の短いものについてだけ GC を行うようにする方式である。

謝辞

日頃御指導頂いている ICOT 第 4 研究室の方々、特に GC の実装に関して助言して頂いた木村康則 (ICOT), 宮内信仁 (三菱電機) の各氏に感謝します。

参考文献

[1] PDSS — 言語仕様と使用手引 — ,ICOT TM-437,1988

[2] 佐藤他: PIMOS の概要 — 並列推論マシン用オペレーティング・システムの概要 —, 情報処理学会第 34 回全国大会 2P-8,1987-3

[3] T.Chikayama, Y.Kimura: Multiple Reference Management in Flat GHC, Proc. of ICLP '87

[4] 宮内, 木村他: MRB による多重参照管理方式 —KL1 処理系における一括型 GC の特性評価 —, 情報処理学会第 35 回全国大会 2Q-7,1987-9

[5] 佐藤他: 並列論理型 OS PIMOS(1) — 資源管理方式 —, 情報処理学会第 35 回全国大会 4D-3,1987-9