5K−2

記憶構成方式 MOLDS の動特性

實藤 隆則、前川博俊
ソニー(株) 情報通信研究所

1. はじめに

ポインタでリンクされた構造を持つデータ (Linked Data Structures, 以下リストデータ)は記号処理の分野においては重要で不可欠である。リストデータは構造の柔軟性からアクセスされる際のアドレスに局所性が少ないため、記憶空間をページング方式などを用いて仮想化すると二次記憶アクセスが大量に発生し、処理効率がかなり低下してしまう。我々はこのような問題点を解決する記憶構成方式 MOLDS (Memory Organization for Linked Data Structures) を提案している[1][2]。

MOLDS では互いに論理的つながりの深いデータの集まり(二次記憶ストラクチャと呼ぶ)を記憶階層間の転送(ストラクチャ・イン／ストラクチャ・アウト) の単位とする。このためガーベジや当面必要無いデータなどを無駄に転送することがない。MOLDS においては主記憶上のデータから二次記憶に転送すべきデータをストラクチャとして抽出する処理がオーバヘッドとなり得るが、二次記憶へのアクセス回数が十分に少なくなれば全体としての効率は向上する。

今回我々は MOLDS の記憶管理アルゴリズムを検討するためのシミュレータを作成した。同時に従来行なわれてきたページング方式を用いたシミュレータも作成し、比較を行なった。

2. シミュレータ

シミュレータは図1に示すような構成をとっている。シミュレータの言語処理系部分には CLOS [3]のサブセット (インタプリタ) を実装した。ベクトル型などはサポートしておらず、データ空間は実質的にコンスセルのみからなる。

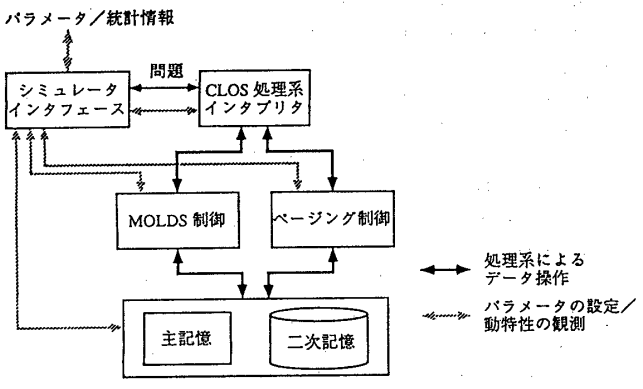

図1. シミュレータの構成

実行時間の測定はシミュレータ内部で管理する仮想時間によって行なう。CPU からの主記憶アクセス時間を1単位時間とし、それに対して CPU 内部での処理時間は無視できるものとした。二次記憶アクセス時間としては、現在の主記憶装置と二次記憶装置の平均的な性能を反映した値を設定した。

MOLDS におけるストラクチャ・アウトは GC を行なっても主記憶上に十分な量のフリー・セルが得られなかった場合に起動する。GC のアルゴリズムはリファレンスカウントを用いたインクリメンタル GC にマーク&スイープ方式を併用したものである。このとき二次記憶上のデータについてはストラクチャ単位で管理することで、直接二次記憶上のデータセルをアクセスすることなく GC の処理が行なえる。

ストラクチャの抽出は以下のようにして行なう。まず、プログラム実行中に生成される論理的に意味のあるデータのまとまり(関数の実行環境やシンボルの実体)に注目し、これらをストラクチャの候補とする。これらの候補を各時点での必要度を反映させたリストで管理し、ストラクチャ・アウトの際にはリスト中の順位の低いものを抽出の対象とする。

今回作成したシミュレータではアルゴリズム検討の必要性などから、関数の環境について、全ての関数呼び出しで順位リストを書き換えるようにしたため、これによるオーバヘッドが生じている。実用的な実装では関数の環境についてはストラクチャ・アウトの必要が生じた時点で呼びだし履歴を調べるなどの方法を用いることで、このオーバヘッドを少なくすることができる。

ページング方式におけるページ置き換えアルゴリズムは LRU 方式を用い、これを実現するためのオーバヘッドは無視できるものとした。GC にはコピー方式を用い、仮想空間上でのデータの局所性をある程度向上させるようにした。

3. シミュレーション結果

シミュレーションに用いたテストプログラムは Boyer ベンチマーク[4]である。Boyerは大量のコンスセルを消費し、データ空間におけるワーキングセットも比較的大きいことから、実際の記号処理システムにおけるアプリケーションに近い振舞いをするものと考えられる。

図2〜5にシミュレーション結果を示す。図2は両方式によるテストプログラムの実行時間の比較である。ストラクチャ・サイズ、ページ・サイズによって差はあるが、MOLDS によって処理速度が10〜40倍向上している。これは図3でもわかるように、MOLDS では二次記憶へのアクセス回数がページング方式に比べてはるかに少ないことによるものである。

図3においてページング方式が主記憶容量に関わらずほぼ一定の二次記憶アクセスを行なっているのに対し、MOLDS では主記憶容量に応じてアクセス量が変化してお

Dynamic Characteristics of the Memory Organization Scheme, MOLDS
Takanori SANETO and Hirotoshi MAEGAWA
Telecommunication & Information Systems Research Laboratory, Sony Corporation

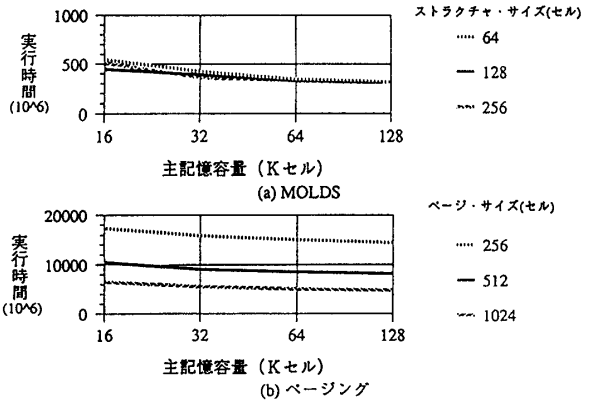

図2. 実行時間

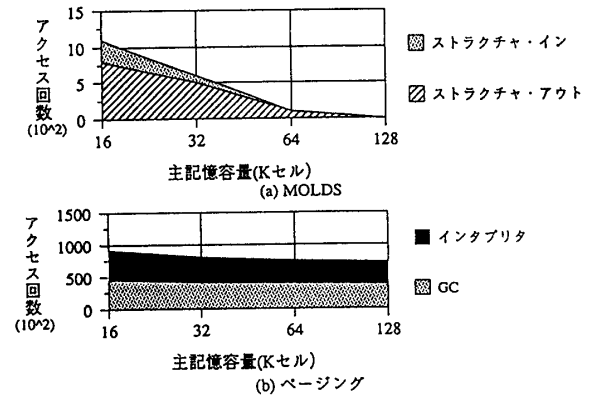

図3. 二次記憶アクセス回数

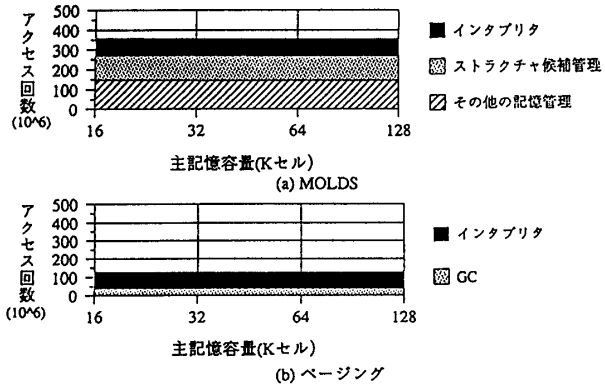

図4. 主記憶アクセス回数

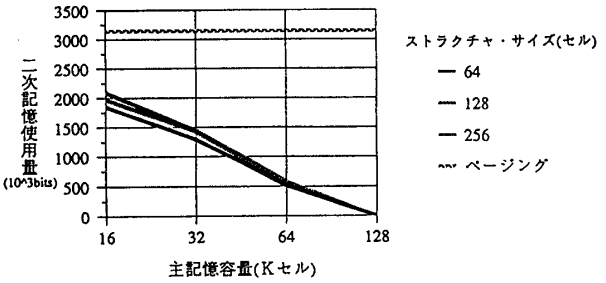

図5. 二次記憶使用量

り、無駄なアクセスが少ないことがわかる。

一方図4からわかるように、 MOLDS ではストラクチャ管理・抽出のためのオーバヘッドが比較的大きい。このオーバヘッドにはストラクチャ・イン／アウトの処理を行なう上で避けられないものもあるが、前節で述べたようにある程度は削減可能であり、さらに効率が良くなることが期待できる。

図5に MOLDS の二次記憶使用量を示す。アクセス回数と同様に、必要量のみが使用されている。さらにその絶対量も、仮想空間の大きさを64Kセルとした場合にページング方式で必要とされる二次記憶容量(図5の"ページング"の線)にも満たない程度であり、 MOLDS では二次記憶の使用効率もよいことがわかる。

4. おわりに

我々の提案する記号処理向き記憶構成方式 MOLDS について、その基本性能をシミュレーションによってページング方式と比較し、実行効率、空間効率ともにかなりよい結果が得られることを確認した。

ページング方式の枠組の中でも処理効率向上のために様々な工夫が行なわれている([5][6][7]など)が、 MOLDS では、記憶管理の基本的な部分で効率の良い処理が可能であり、データ構造や処理系の構成ををより素直に実現できる。

今後、より詳細なシミュレーションによってストラクチャ抽出アルゴリズムや管理用のデータ構造の検討などを行なっていく。また、ページング方式上での様々な手法との比較も検討する。さらに現在我々が開発を進めている記号処理計算機 Lilac [8]上で、実機への実装による有用性の検証も行なっていく。

謝辞

本研究の遂行にあたり有益な助言を頂いた福田譲治部長を始めとする研究部のメンバーに、また、本研究の機会を与えて頂いた当社総合研究所宮岡所長、情報通信研究所松田所長に感謝する。

シミュレータの作成及びデータの採取にあたっては、(有)アクセスの村田典幸氏に御協力を頂いた。

【参考文献】

[1] 前川博俊 他: Linked-Data のその構造に基づく記憶空間の構成, 情報処理学会研究会報告, ARC80-5 (1990).

[2] Maegawa, Hirotoshi: Memory Organization and Management for Linked Data Structurs, ACM 1991 Computer Science Conference (1991).

[3] Bobrow, D., et al.: Common Lisp Object System Specification, X3J13 88-002R(1988).

[4] Gabriel, R. P.: Performance and Evaluation of Lisp Systems, MIT Press (1985).

[5] Cohen, J.: Garbage Collection of Linked Data Structures, Computing Surveys, Vol.13, No.3, pp.341-367(1981).

[6] Moon, D. A.: Garbage Collection in a Large Lisp System, Conference Record of the 1984 ACM Symposium on Lisp and Functional Programming, pp.235-246(1984).

[7] Bishop, P. B.: Computer Systems with a Very Large Address Space and Garbage Collection, MIT/LCS/TR-178, Laboratory for Computer Science, MIT (1977).

[8] 安田弘幸 他: 計算資源指向型並列分散処理システム — Lilac —, 情報処理学会研究会報告, SYM55-2 (1990).