論文の内容の要旨

論文題目
世代別ごみ集めのための
オブジェクトの配置法

氏名　小林 広和

本論文は、プログラム言語処理系における自動記憶管理 (ごみ集め) において、ハードウェア支援を行なう際に問題となる「ルート検査のコスト」を低減するために、オブジェクトの配置を工夫することにより性能の向上をはかれることを提案、実証したものである。

古くから用いられていたマークスイープ方式やコピー方式などの従来型ごみ集めを利用した場合には、長時間にわたって通常処理が停止するという問題が発生していた。現在ではこの問題を解決するための方法として、世代別ごみ集めを利用することが多い。世代別ごみ集めは、誕生してからの年齢によってオブジェクトを分類し、若いオブジェクトだけを通常のごみ集めの対象とすることによって、ごみ集めがひき起こす通常処理の停止時間を短縮する。しかし、世代別ごみ集めでは、古いオブジェクトに対する書き換えによって発生する世代間参照を検出する必要がある。これを行うための方法として、ソフトウェアだけによる方法と、ハードウェアを利用する方法の二通りの方法がある。一般的なマシンでハードウェアを利用する方法を用いる場合には、ページングハードウェアを利用することができる。従来の OS では、ページングハードウェアで管理している情報に直接アクセスできなかったために、ページトラップ機構を利用して必要な情報を得ていた。しかし、最近では、ペー

ジングハードウェアが管理する必要な情報を呼び出すためのシステムコールを提供するOSが登場しているため、これを世代間参照の検出のために利用することが可能になっている。

以上のような背景から、本研究では、ハードウェア支援による世代別ごみ集めの性能向上が重要であると考え、ページングハードウェアを利用して世代間参照を行う世代別ごみ集めをGNU Emacsに実装した。そして性能計測を行い、ページングハードウェアを利用した世代別ごみ集めでは、ルート検査の際の局所性が重要であることを発見した。

ここで言語処理系の協力によって、その局所性を向上することができるとの仮説をたてた。その方法として、書き換えの頻度によって、配置領域を分割する手法を提案した。

そしてその仮説を実証するために、提案した手法についてダイナミックバインド言語のシンボルオブジェクトに対して実験とシミュレーションを行い、ルート検査の際の局所性を向上でき、世代別ごみ集めの性能を向上できることを示した。

このことから、他のデータタイプに付いても、書き換えの頻度によって配置領域を分割する手法が有効であると推定できる。

以下で、本論文の内容を具体的な結果を交えて説明する。

本研究では、ページングハードウェアの機能を利用して世代間参照の検出を行う世代別ごみ集めをGNU Emacsに実装した。GNU Emacsは、Emacs LispというLisp処理系が組み込まれたテキストエディタである。Emacs Lispは、GNU Emacsの基本的な機能を定義するのに用いられるだけでなく、機能の拡張のためにも用いられ、多くのアプリケーションがEmacs Lispで作成されている。本研究では、実装した世代別ごみ集めの性能を評価するために、実際の利用時の状況を疑似的に作り出すことにした。こうすることで、実使用環境に近い状況での性能評価を行った。その結果、世代別ごみ集めを実装することによってごみ集めに必要な時間を短縮することができ、200msec以下におさえることが可能であると分かった。

ページングハードウェアを利用して世代間参照を検出する場合、世代間参照を直接検出するのではなく、間接的に検出する。つまり、前回のごみ集め終了時から今回のごみ集めの開始時までにページの内容が書き換えられたページに世代間参照が含まれる可能性があるので、これらのページをページングハードウェアを利用して検出する。そしてこれらをすべて走査し、実際の世代間参照を検出する。このような処理は、ごみ集めの処理を行う場合の起点となるルートを求める処理であるので、ルート検査と呼ぶ。一般的なハードウェアでは検出の単位となるページのサイズが4096バイトとなり、一般的なオブジェクトの大きさに比べ大きい。計測の結果、1ページ内に存在する世代間参照の数が最大でも10個であり、1ページに存在可能な世代間参照の数(1024個)に比べ少ないことが分かった。つまり、ページあたりの世代間参照

の密度を上げることにより、書き換えが行われるページ数を減らすことができると考えられ、これによりページの走査に必要な時間の短縮が行える。つまり、ルート検査のコストが少なくなると考えられる。

このように、ページングハードウェアを世代間参照の検出に利用する場合には、従来は指摘されていなかったルート検査の際のオブジェクトの配置の局所性が問題となることが分かった。

ここで、言語処理系の協力によって書き換えの局所性を向上することができるという仮説をたてる。そして、局所性を向上するための手法として、書き換えの頻度によってオブジェクトを違う領域に分けて配置するという方法を提案する。この方法によって書き換えの局所性が向上すれば、世代別ごみ集めを行なう場合のルート検査のコストを削減できる。

この仮説を実証するために実際の言語処理系において実験を行った。実験は、Emacs Lispのシンボルオブジェクトに対し、バイトコードコンパイラを用いた静的解析の結果、書き換えが行われると分かったシンボルオブジェクトを他のシンボルオブジェクトとは別のページに配置するという手法を用いた。このときの世代別ごみ集めの性能を計測し、この手法の効果を判定した。

Emacs Lispにおけるオブジェクトの書き換えのトレースの結果では、シンボルオブジェクトはconsオブジェクトに比べて書き換えの頻度の差が大きい。これは、Emacs Lispではダイナミックバインドであるので、書き換えが何度も行われるシンボルオブジェクトが存在するためである。実験の結果、書き換えのあるシンボルを他のシンボルオブジェクトとは別のページに配置することで、プログラムの実行時間およびごみ集めの実行時間を短縮することができることが分かった。ごみ集めの実行時間は、本方法を利用することによりCPUのユーザ時間で約10%短縮できた。ここで、短縮できたごみ集めの実行時間が、書き換えの局所性が向上したことによって、ルート検査に必要な時間が短縮されたことによるものであることを確かめるためにシミュレーションを行った。その結果、短縮された時間の50%はルート検査の時間の短縮によるものであることを確認した。プログラムの実行時間の短縮は、書き換えの多いオブジェクトが同じページに配置されることになったことによるキャッシュのヒット率向上によるものであると推察される。

さらに、ハードウェアによるライトバリアとソフトウェアを用いたライトバリアの性能の比較をシミュレーションによって行った。この結果、ハードウェアによるライトバリアがソフトウェアを用いたライトバリアに対して同程度の性能であると分かった。

これらの結果から、他のデータタイプにおいても書き換えの頻度の推定の方法があれば、言語処理系の協力によって書き換えの局所性を向上することができることが分かった。この方法を利用して、ページングハードウェアをライトバリアとして利用した世代別ごみ集めにおいて、ルート検査に必要な時間を短縮することができ、ごみ集めの高速化が可能である。

本研究の寄与として以下の点があげられる。

1. ページングハードウェアを用いた世代別ごみ集めにおいて、従来指摘されなかったルート検査の際の局所性の問題が存在することを発見し、これがごみ集めの性能向上の鍵となっていることを指摘したこと。
2. 書き換えの局所性が通常実行だけでなくごみ集めの性能にも大きく影響を与えるという仮説を提示したこと。
3. 具体的な局所性の向上法として、オブジェクトの書き換えの頻度によって配置する領域を分割する手法を提案したこと。
4. テストケースとして Emacs Lisp のシンボルオブジェクトに対して実験を行い、この手法が効果的であることを明らかにしたこと。
5. 書き換えの頻度推定ができる場合は、それがごみ集めの性能向上に利用できることが明らかになったこと。