並列処理における重みつき参照カウントを用いた実時間GC

市吉伸行 六沢一昭 近山隆 中島克人 宮崎敏彦 杉野栄二
(財) 新世代コンピュータ技術開発機構 (ICOT)

1 はじめに

ICOTで開発中のマルチPSIは、最大64台のPSIをネットワークで結合した並列マシンである[1]。このマシン上の並列論理型言語KL1の処理系における PE間アドレス管理の単一参照方式については[2]で述べたが、今回は多重外部参照の管理における PE間実時間GC方式として採用した、重みつき輸出カウント方式(Weighted Export Count(WEC)方式) を説明する。これは参照側にも参照カウントを持たせたのが特徴で、被参照側にのみ参照カウントがある通常の参照カウント方式に比べて並列環境に適しており、メッセージ数を増やさずにPE間の実時間GCを可能にしている。この方式は複数プロセッサでデータを共有するようなシステム一般において有効である。

当稿では、WECの原理とその管理、PE間メッセージプロトコル等について述べる。

2 計算モデル

次のような計算モデルを仮定する。

- 有限個のプロセッサ(PE)があり、それらは局所メモリを持つ。(共有メモリは存在しない。)
- 全てのPEを結ぶネットワークがあり、各PEは任意のPEと非同期なメッセージ通信ができる。
- PEは自分の局所メモリへの参照を他のPEに輸出できる。他PEへの参照(外部参照)を輸入したPEは、%readメッセージまたは%unifyメッセージを輸出元PEに送信することにより外部参照先の読出しまたは書込み(ユニファイ)ができる。

外部参照アドレスは、PE番号とPE内IDの対で表すものとする。各PEは外部からの参照を一括管理するための輸出表を持っている。PE内IDは輸出表のエントリ番号で、対応する輸出表エントリは局所メモリを指す。(図1) このような間接参照方式にすると、輸出されたセルの局所アドレスが局所GCによって動いても、輸出表エントリを更新すれば外部参照アドレスを変えずに済む。また、PEは同一の外部参照に対しては、ただ一つの外部参照セルを対応させる。

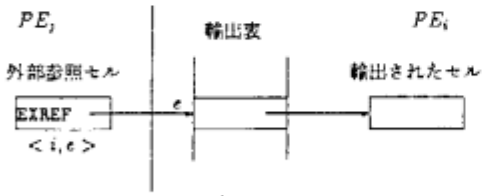

図1. 外部参照の表現

3 PE間GCとその困難

上記のような計算モデルにおけるPE間GCとは、もはや外部から参照されなくなった輸出表エントリの解放のことである。PE間GC方式として、並列版マーク・アンド・スイープ方式と参照カウント方式が考えられるが、それぞれ以下のような問題がある。

並列版マーク・アンド・スイープ方式では、マーキングメッセージを用いてPE間構造のマーキングを行なうが、最悪の場合、多くのPE間にまたがる構造体がボトルネックとなって、処理時間が局所GC時間の2桁増し(PE内アクセスとPE間アクセスの比)程度にもなり得る。

参照カウント方式では、輸出表エントリは参照カウントを持ち、それがゼロになったときにエントリを解放できるというものである。しかし、メッセージの到達時間が予測できないようなシステムにおいてはインクリメントおよびデクリメント用メッセージだけでは参照カウントの正しい管理ができない。メッセージ着信確認(ACK)メッセージを導入すれば順序性が保証できるが、オーバヘッドが大きい。

4 WEC方式

4.1 原理

WEC方式では、輸出表エントリだけでなく、外部参照ポインタにも参照カウント (常に正の値) を持たせる。 (外部参照ポインタは外部参照セルだけでなく、メッセージ中のものも含む。) この参照カウントをWeighted Export Count(WEC)と呼ぶ。通常の参照カウント方式では、

(輸出表エントリXの参照カウント) =
(Xへの外部参照ポインタの数)

が成り立ったが、WEC方式では、

(輸出表エントリXのWEC) =
(Xへの外部参照ポインタのWECの和)

を保つようにする。従って、輸出表エントリXに関して、 Xへの参照ポインタが存在しないことと、XのWECの値がゼロであること、は同値になる。通常の参照カウント方式は、WEC方式においてポインタのWECの値を1に制限したものと見なせる。

WEC方式が通常の参照カウントより優れている点は、参照ポインタを分割する際に被参照側のカウントを更新する必要がないことである。これにより参照先へのアクセス回数が減るとともに、非同期通信による更新の遅れの問題もなくなっている。

なお、重みつき参照カウント方式は、共有メモリ型並列マシン上の関数型言語処理系に用いられている[3,4]

A Real-time garbage collection for parallel processing using weighted reference counting
N.Ichiyoshi, K.Rokusawa, T.Chikayama, K.Nakajima, T.Miyazaki, E.Sugino (ICOT)

が、それらにおいては以下に述べる外部参照の輸出入時の処理に対応するものはない。

4.2 WEC の管理

WEC の管理は次のように行なう。

輸出: 新たに生成する外部参照に適当な重みwを持たせ、輸出表エントリの WEC にw を加える。あるセルの最初の輸出の際は、新たな輸出表エントリを割り付ける。

輸入: PE が外部参照を輸入した場合は、その WEC を対応する外部参照セルの WEC に加え入れる。最初の輸入の際は、新たな外部参照セルを割り付ける。輸入には、輸出表からの場合と他の外部参照の分与の場合の2通りがあるが、それらは区別しない。

分割: 1つの外部参照を複数に分割するときには、分割後の参照ポインタの WEC の和が元の WEC に等しくなるようにする。(図2) %read および %unify による参照も外部参照の分割 (PE 内の外部参照セルとメッセージ中の外部参照への分割) と考える。

消費: 外部参照を消費して2度と使わない場合は、その外部参照の WEC を %return_WEC メッセージにより参照先の輸出表エントリに返す。 %read または %unify による参照で外部参照を消費してしまう場合は、WEC の全てをそのメッセージにつける。局所 GC により外部参照セルが解放されたときも、%return_WEC メッセージを送る。 WEC を返却された輸出表エントリは、それを自分の WEC から引く。その結果がゼロならば、エントリを解放する。

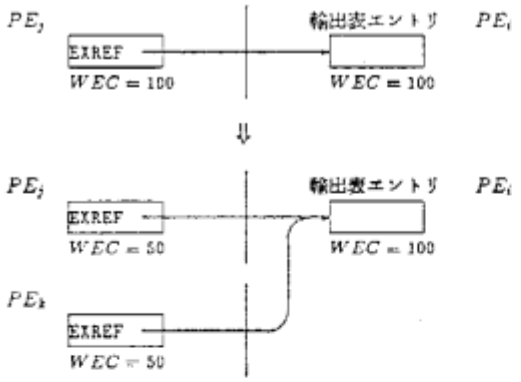

図 2. WEC の分割

5 WEC 方式の問題点とその対処

5.1 循環構造の問題

WEC 方式は、通常の参照カウント方式と同様、(PE をまたがる) 循環構造を回収できない。このようなゴミが溜まると、全域 GC が必要になる。

KL1 の場合、プロセス間の因果連鎖がサイクリックになることがあるが、正常終了すれば、因果リンクであるストリームが閉じられるので循環構造は残らない。しかし、プログラムがデッドロックしたり、アボートされると循環構造が残る。また、ユーザが故意にゴミになる循環データを多量に作る可能性もある。

5.2 分割できない WEC の問題

実際の計算機で WEC の値を正整数とし WEC の分割を続けて行くと、いずれは WEC がこれ以上は分割できない状況に達する。その場合の対処の仕方として2通りが考えられる。

(1) 間接輸出法: 外部参照セルそのものを輸出する。(これによって参照チェーンが長くなってしまうのは問題である。)

(2) WEC 補給法: WEC の分割を中断し、%request_WEC メッセージを輸出表エントリに送る。メッセージを受け取った輸出表エントリは、%supply_WEC メッセージによって WEC を補給してやる。

[3,4] においては前者を採用しているが、我々の処理系では両者を併用することにした。これは、関数型言語と違って KL1 では不用意に外部参照セルを輸出するとユニファイによって外部および内部参照ポインタのみからなるループができてしまうからである。

6 マルチ PSI 上処理系における実装

輸出表エントリの WEC は 64 ビット符号なし整数で表わし、これは絶対にあふれないと仮定する。外部参照セルの WEC は 32 ビット符号なし整数で表わす。輸出の際の WEC としては、固定値2^{24}を与える。輸入により WEC があふれることがあるが、その場合は 2^{24}を残して全てを輸出表エントリに返す。 WEC の分割については、他の PE に外部参照を分けるときには、WEC の現在値の半分を与え、また外部参照の参照先へのメッセージ中の WEC は1とする。(ただし、それが外部参照の消費である場合は WEC の全てを与える。)

上記実装によれば、輸出された外部参照は最低 24 回は WEC の補給なしに分割できる。このため、 WEC 補給のオーバヘッドは十分に小さいと期待される。

7 おわりに

マルチ PSI 上 KL1 処理系で採用した PE 間実時間 GC 方式である WEC を説明した。これは、非同期メッセージ通信を行なう並列環境一般で有効であり、 GC を行なわない場合と比べてメッセージ数のオーバヘッドが小さい点で優れている。

前述した WEC 方式の問題点については、実装後のテストにおいて評価したい。

参考文献

[1] 淵和男. Multi-PSI システムの概要. 情報処理学会第35回全国大会 5Q-8, 1986.

[2] 六沢一昭 他. マルチ PSI 第2版の PE 間アドレス管理方式. 情報処理学会第35回全国大会 3C-4, 1987.

[3] D.I.Bevan. Distributed garbage collection using reference counting. In *Parallel Architectures and Languages Europe*, pages 176–187, 1987.

[4] P.Watson and I.Watson. An efficient garbage collection scheme for parallel computer architectures. *ibid.*, pages 432–443.