Committed-Choice 型言語の分散処理系における参照経路の圧縮*

1T－5

田口 景介　青柳龍也　有山正孝†
電気通信大学‡

1 目的

fleng処理系の基本的な動作は、ゴールプールから適当なゴールを1つ選び、このゴールとのユニファイに成功した定義節によってリダクションを行うというものである。本研究で扱っている処理系は分散処理を行っているため、ユニファイの対象となるオブジェクトが遠隔ノード上にあれば、遠隔ユニファイを行わなければならない。遠隔ユニファイを行うためには、引き数の個数とそれぞれの値をすべて他方のノードへ送らなければならない。したがって、このときに送る必要のあるオブジェクトを減らすことが、メッセージ数の減少、ひいては処理系の処理速度の改善につながると考えられる。

このために、飛び石のように、複数のノードを無駄に遠隔参照している状態をなくすことを考える。これを参照経路の圧縮と呼ぶ。

2 Committed-Choice 型言語 Fleng

2.1 Fleng

Flengは GHC に良く似た言語ではあり、その構文はガードなしのホーン節を採用している。ただし、Flengは失敗の概念を持たないために GHC とは異なるし、論理型言語でさえない。

我々が実装した fleng[1] は、ゴールごとの分散処理を可能にしている。ノード間の同期通信は、遠隔参照した変数の具体化を待つことによって行われる。また遠隔オブジェクトとのユニファイは、局所オブジェクトの複製を遠隔オブジェクトの存在するノードに作成することによって行う。

2.2 分散 Fleng の実装

遠隔ノードのオブジェクトへの参照は、ERT(External Reference Table) と ODT(Object Directory Table) を介して行われる。局所参照オブジェクトはまず局所 ERT を指し示す。次に局所 ERT は目的の遠隔ノードにある ODT を指し示し、この ODT は遠隔ノード上のオブジェクトを指し示す。

ERT のエントリ1つは、参照先の遠隔ノードを表す値とシステム全体で唯一の値を持つ GID(Global ID) を持つ。ODT のエントリ1つは、参照元の遠隔ノードを表す値と対応する ERT のエントリと同じ値の GID、それと局所オブジェクトを指し示すポインタを持つ。GID の値は、GID を発行したノードを表す値と、単調増加する関数から得られた値の組からなっている。GID は ERT と ODT のエントリの1対1の組で使われ、それ以外で同じ値が使われることはない。

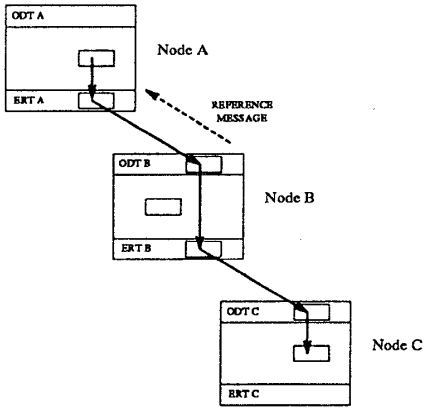

図 1: 無駄な遠隔参照

3 参照経路の圧縮

3.1 通常の REFERENCE メッセージの処理

遠隔参照を作るときには REFERENCE メッセージを送信するが、このときの処理は次のようになる。まず局所ノードの ODT に新たにエントリを作り、参照される局所オブジェクトへのポインタをそのエントリに書き込む。次に REFERENCE メッセージを遠隔ノードへ送信し、これを受信した遠隔ノードは ERT に新たにエントリを作り、REFERENCE メッセージを送信したノードへのポインタをそのエントリに書き込む。

以上の処理を行って遠隔参照が作られるが、遠隔参照オブジェクトとのユニファイに成功した場合、同一ノード内で ODT のエントリが直接 ERT のエントリを指している状態が作られる。図1では、ノードBは単に通過するだけであり、まったく無駄な参照であると言える。

3.2 COMPRESS PATH メッセージの導入

そこで、飛び石状態の参照を、参照元から最終的な参照先への参照に書き換えることを考える。これには、その後の参照処理を単純化することと、ODT, ERT エントリを節約するというメリットがある。

参照経路の圧縮を行うために、COMPRESS PATH メッセージを導入する。REFERENCE メッセージを送信する前に参照経路の圧縮が可能か調べ、可能ならば COMPRESS PATH メッセージを送信する。

図2は、ノードB上の処理系がノードAからノードBへの遠隔参照を要求した状態を表している。ただし、ERT 上のオブジェクトを遠隔参照しようとしたので、ノードBの ODT にはエントリを作らず、代わりに COMPRESS PATH メッセージをノードCに対して送信する。ノードCに送信したのは、y が

*Compression of Reference Path on Distributed Committed-Choice Language

†Keisuke TAGUCHI, Tatsuya AOYAGI, Masataka ARIYAMA
‡The University of Electro-Communications

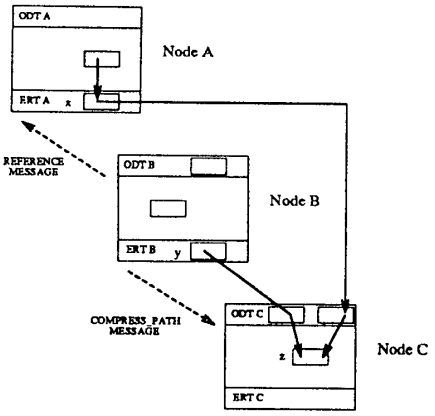

図 2: COMPRESS PATH

参照しているオブジェクトがノードC上にあるためである。また、ノードAに対しては通常通りREFERENCEメッセージを送信する。

COMPRESS PATHメッセージを受信したノードCは、ODTに新しくエントリを作り、ノードAからの参照を受け入れる準備をする。このとき作成したエントリは、yが参照しているオブジェクトzを指し示すようにしておく。

4 評価

本研究で実装を行ったfleng処理系が利用しているメッセージのうち、REFERENCEメッセージとOBJECTメッセージが、参照経路の圧縮を導入することによって、増減するものと考えられる。REFERENCEメッセージは、1つの遠隔参照を作成するメッセージである。OBJECTメッセージは、遠隔ノードにオブジェクトの複製を作成するメッセージである。1つのOBJECTメッセージで複数のオブジェクトを伝えることができるので、OBJECTメッセージが運ぶ情報量は一定ではない。

4.1 行列の計算

3行3列の行列の掛け算を行うサンプルプログラムである。2つのノードで分散して処理を行う。このサンプルプログラムでは、わずかのCOMPRESS PATHメッセージしか発行されない。結果として、後述する理由によりOBJECTメッセージが増加し、付随してREFERENCEメッセージも増加してしまっている。

	REF	OBJECT	COMPRESS	合計
圧縮なし	79	153	0	232
圧縮あり	82	165	6	253

4.2 リストの逆転

リストの要素の順序を逆にするサンプルプログラムである。2つのノードで分散して処理を行う。このサンプルプログラムでは、非常に多くのCOMPRESS PATHメッセージが発行される。しかし、参照経路の圧縮が行われた参照オブジェクトのうち有効に使われるのはほんの一部であるため、多くのCOMPRESS PATHメッセージが無駄にメッセージ数を増やすことになっている。

	REF	OBJECT	COMPRESS	合計
圧縮なし	205	270	0	475
圧縮あり	201	266	77	544

4.3 リストの並べ変え

リストの要素を昇順に並べ変えるサンプルプログラムである。3つのノードで分散して処理を行う。REFERENCEメッセージ、OBJECTメッセージ共に減少しているが、COMPRESS PATHメッセージによるメッセージ量の増加によって、その効果は打ち消されてしまっている。

	REF	OBJECT	COMPRESS	合計
圧縮なし	109	205	0	314
圧縮あり	105	189	20	314

5 考察

参照経路の圧縮が行われると、その後不必要なREFERENCEメッセージやOBJECTメッセージが送信されなくなることが期待できる。しかし、参照経路の圧縮を行った参照オブジェクトに対して、その後一度も遠隔ユニファイが行われなければ、COMPRESS PATHメッセージの分だけ余分にメッセージが送信されることになる。したがって、参照経路の圧縮が必ずしもメッセージ量の減少につながるとは限らない。評価の結果を見ても、REFERENCEメッセージやOBJECTメッセージを減らすことに成功しているものの、COMPRESS PATHメッセージによるコストが高いため、全体として改善とはなっていない。

また、場合によってはOBJECTメッセージが増加することもあり得る。参照経路の圧縮を行わない場合は、REFERENCEメッセージによって作成される遠隔参照の参照先ノードは、すべてREFERENCEメッセージを送信したノードとなる。そのため、複数のオブジェクトをまとめて1つのOBJECTメッセージで送ることができる。しかし、参照経路の圧縮を行うと、REFERENCEメッセージを送信したノードとは異なるノードへの参照が作成されるので、OBJECTメッセージを参照先のノードごとに送信しなければならないためである。したがって、全体として作成されるオブジェクトの量は一定だが、OBJECTメッセージの個数は増加する。

そこで今後の方針としては、圧縮可能な参照はすべて圧縮している現在の実装を、有効と思われる参照経路だけを圧縮するように変更することが考えられる。参照される回数が多く、参照経路の長い参照オブジェクト程効果が大きいので、これを予想する手段が必要となる。長生きしているオブジェクト程参照される回数が多いと予想できるので、これに対して圧縮を行うことが考えられる。また、一度参照された参照経路についてだけ圧縮を行うなどの手段が考えられる。

また、図2の例でノードAとノードCが仮に同じノードであれば、遠隔参照を圧縮するだけではなく、局所参照に変更できる。これは少ないノードで実行するときには特に有効であると考えられる。

参考文献

[1] 藤原克則“分散fleng処理系における大域的ゴミ集め”電気通信大学修士論文, 1993