GHC のメッセージ指向の並列処理 — 概要 —

5M−9

上田 和紀
(財) 新世代コンピュータ技術開発機構

森田 正雄
(株) 三菱総合研究所

1. はじめに

我々はさきに，GHC (正確には Moded Flat GHC) のメッセージ指向処理方式を提案した [1]. これは，従来の方式と異なり，プロセスの実行の中断が頻繁に起きるプログラムを効率よく実行することを目的とした方式である．本稿では，このメッセージ指向処理において並列性を活かす方法を考察する．並列実行の効果についての評価結果は [2] を参照してほしい.

2. メッセージ指向処理

GHC のメッセージ指向処理とは，単一化によるストリームの要素の生成を，そのストリームの受信プロセスの (遠隔) 呼出しの形にコンパイルして実行する方式である．ストリームは本来，非有界バッファの役割を果たすが，メッセージ指向処理では，メッセージと制御を同時に受信先に渡すことにより，バッファリングのオーバヘッドを避けている．ただし，プロセス間通信の形態によっては，バッファリング操作が本質的に必要な場合もあり，そのときはメッセージのバッファリングを行なう.

従来の処理方式 (プロセス指向と呼ぶ) では，実行するゴールの切替えの頻度をできるだけ減らし，末尾再帰呼出しの最適化を活用することで効率化を図っていた．メッセージ指向処理では，実行するゴールの切替えと，通信データの授受の手間を低減することによって効率化を図っている．これにより，メッセージに対する応答速度が重視される用途で効率がかなり改善した．応答速度が重要なプログラムには，動的データ構造をプロセスとストリームで表現したプログラムや，要求駆動のプログラムなどがある.

メッセージ指向処理は，ストリーム通信の向きや受信者数等の静的解析が前提となるが，モード制約の概念に基づく解析技法がすでに提案されている [1].

3. 並列化

メッセージ指向処理では，どのようにしたら並列性を活かすことができるだろうか？ 二つの方式が考えられる.

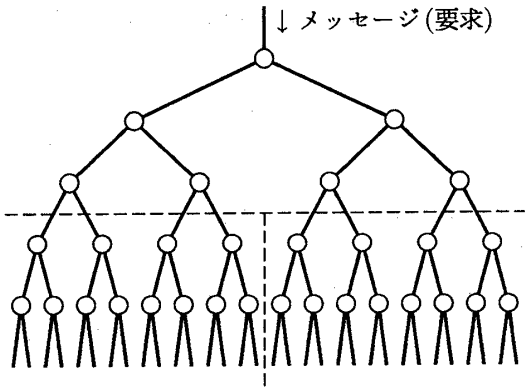

図1　プロセスの2分探索木

(1) (ゴール非共有方式) 実行すべきゴール群を，複数のプロセサが分担して受け持つ．各プロセサは，各自の担当するゴール群に来るメッセージの処理を行なう．例えば図1のようなプロセスの2分探索木の，破線で区切った三つの部分を3台のプロセサで分担することを考え よう．各プロセサは，自分の受け持つゴール群の中でメッセージ指向の処理を行なう．プロセサをまたがるメッセージ送受信は，バッファリングを伴うプロセサ間通信にコンパイルされる.

(2) (ゴール共有方式) 複数のプロセサがすべてのゴールを共有する．各プロセサは，未処理のメッセージ送信の仕事を共有のスタックから獲得し，それから始まるメッセージ送信の連鎖の処理を行なう．これによって，異なるメッセージ送信の連鎖をパイプライン的に並列処理できる．例えば図1の2分探索木では，根 (root) に送られてくる探索要求を一つずつ分担して処理する．一つの要求に答えるためには探索木の内部で0個以上のメッセージ送信をしなければならないが，それらは同一プロセサが担当する.

方式 (1) の利点は，ゴールを共有しないので，分散メモリ計算機にも適用できることである．しかし，あるメッセージ送信によって引き起こされるメッセージ送信の連鎖を分担して処理するので，全体のスループットは上がりうるものの，レスポンスの点で (2) より不利である.

方式 (2) は，共有メモリ計算機で現実的な方法である．大規模・高並列処理には適さないが，メッセージに対するレスポンスや負荷分散の点では有利である．以下では，このゴール共有方式をさらに検討する.

Messege-Oriented Parallel Implementation of GHC Programs: Overview
Kazunori Ueda[1], Masao Morita[2]
[1]Institute for New Generation Computer Technology
[2]Mitsubishi Research Institute

4. ゴール共有方式の並列実行

メッセージ指向方式では，実行中でないゴールをできるだけ中断(メッセージ待ち)状態におくことによって，送信したメッセージが直ちに受信できるようにする．簡単な例として，次のような節で定義される，メッセージを転送するだけのプロセスp(Xs,Ys)を考えよう．

```
p([T|Xs1],Ys) :- true |
        Ys=[T|Ys1], p(Xs1,Ys1).
```

第1引数 Xs にメッセージが到着して，この節を用いたリダクションが起きると，二つのボディ・ゴールのうちp(Xs1,Ys1) を先に実行して，まずプロセスを定常(メッセージ待ち)状態に戻す．その後，ゴール Ys=[T|Ys1] を，ストリーム Ys の受信者の呼出しの形で実行し，データ T の処理を行なう．

これをゴール共有方式で並列化すると，ストリーム Xs の中の複数個のメッセージの送信を，別々のプロセサが担当することになるかもしれない．このため，プロセスp(Xs,Ys) へのアクセス競合を防ぐための施錠機構が必要になる．一般に，メッセージを送るときはまず受信ゴールを施錠しなければならず，リダクションが済んでプロセスが定常状態に戻るまでは解錠してはならない．

さらに，メッセージ指向処理では，ストリームの要素の順序関係を，空間的に(つまりリスト構造として)ではなく，メッセージ送信の時間的順序として表現している．このため，ストリームの要素の順序が乱れないように配慮しなければならない．例えば上の例で，Xs にメッセージ α と β がこの順で送信されたら，Ys にも α と β をこの順で送信しなければならない．これを保証するためには，ゴールp(Xs,Ys) を解錠する前に，ストリーム Ys の受信先を施錠するようにしなければならない．

このように，ゴール共有方式の並列実行では，変数や共有スタックのほかに，ゴールにたいして施錠が必要になる．また，ここでは詳説しないが，すぐに処理できないメッセージをバッファリングしたり，そこからメッセージを取り出したりする場合，バッファにも施錠する必要がある．

これらの資源の解錠を待ち合わせる方法として，動的待合せ(busy wait)と中断待合せ(suspension)が考えられる．ゴール以外の資源については，施錠時間が非常に短いと期待されるので動的待ち合わせが適切である．ゴールについては，次のことを考慮しなければならない．

(1) リダクションにかかる時間，つまりガード部を実行し，選択した節のボディ・ゴールを定常状態に至るまで展開するのにかかる時間は，かなり長くなる場合もあるが，通常は短い(数十命令程度)．

(2) リダクションのときに，一時的に複数のゴールを施錠しなければならないことがある．したがって，動的待ち合わせでは，プロセスとストリームが環状構造をなす場合にデッドロックの可能性がある．

そこで，ゴールの解錠については，大多数のゴールがリダクションできる時間動的待合せを行ない，それで解錠されなければ中断するのが適当であろう．

なお，GHC のガード機構による同期，すなわち(ストリーム以外の)変数の値の待ち合わせは，中断によって行なうべきである．

5. プログラムの並列性

ゴール共有方式の並列性の源泉は，複数のメッセージ送信の連鎖が，相互に干渉し合わない限り並列処理できるところにある．例えば，図1のプロセスの2分探索木では，要求どうしの間に(次の探索のキーが前の探索の結果から決まるというような)依存性がないかぎり，それらをパイプライン処理できる．しかし依存性があるときは，複数台のプロセサを割り当てると，メッセージ送信の動的待合せや中断・再開始処理のためにかえって効率が低下する危険が大きい．

別の例として，素数の倍数を除去するフィルタ・プロセスをつないで素数列を生成するプログラムを考える．データ駆動計算では並列性を活かせるが，要求駆動計算では並列性を活かすことができない．前の要求に対する答 p が求まるまで，次の要求を処理する最上流のフィルタ(p の倍数を除去する)が，定常状態にならないからである．

このように，並列性はいつも活かせるとは限らない．効果のない並列実行を避けるためには，並列実行してほしいメッセージ送信とそうでないものの区別をプログラマに指定してもらうのが，原始的だが現実的な方法であろう．この場合，前者は共有のスタックに，後者は各プロセサの私有スタックに積み，私有スタックの仕事がなくなったときに共有スタックから仕事を獲得するようにすればよい．

参考文献

[1] K. Ueda and M. Morita, A New Implementation Technique for Flat GHC. In *Proc. Seventh Int. Conf. on Logic Programming*, MIT Press, 1990, pp. 3–17.

[2] 森田，上田：GHC のメッセージ指向の並列処理 — 初期評価 —．本予稿集，1991.