

3H-2

分散メモリ上の並列Lispへの ストリーム通信の導入

吉池 久夫 中西 正和

慶應義塾大学大学院 理工学研究科 計算機科学専攻

1. はじめに

Lispにおいて、並列性を導入する研究は数多く行われ、共有メモリ上においては数多くの並列Lispが開発されてきた[1][2]。しかし、分散メモリにおいては、データの一貫性の保持が困難である、サイズの大きいリストの通信にかかるコストが大きいなど問題点は多い。

本研究ではリストの通信にかかるオーバーヘッドに着目した。プロセッサの無駄な待ち時間を短くすることによる処理効率向上を目標として、ストリーム通信によってリストを通信する並列Lispを富士通AP1000上に実装し、その性能の評価を行った[4]。

2. ストリーム通信

ここで、ストリーム通信について説明する。

本システムでストリーム通信は以下のように働く。

- 受信側は送信側の送った順にデータを受信して評価を行う
- 送信側はリストを先頭の要素から部分評価して送信していく

このような通信方式により、リスト全体が評価されてから通信が起きる方式（今後、一括通信方式と呼ぶ）と比較して、受信側の無駄な待ち時間が短くなり、受信側と送信側は、ある時点ではリストの違う部分を評価していることになるのでパイプライン処理的な高い並列性を抽出することが可能となる。

3. 実装方法

ここで、本システムの実装方法について述べる。まず、本システムの特徴は以下のようになっている。

- データフロー式な並列処理

Parallel Lisp with stream communication on distributed memory

Hisao YOSHIKE Masakazu NAKANISHI

Department of Computer Science, Faculty of Science and Technology, Keio University 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223, Japan

- データの通信路は一方方向
- データの送受信を行う関数とプロセスの依存関係を記述する構文により、並列処理を実現
本研究では以下の構文を用意した。

`connect` : 指定した相手に通信要求を行なう。

`listen` : 指定した相手から通信要求を受けて、受信路を解放する。

`mcons` : 実際にデータの送信を行なう。詳細は後に述べる。

`switch-cid` : セル¹にプロセスを割り当てる。

3.1 ストリーム通信の実現

並列推論言語KL1では、ノード間のリストの通信においてはコンスというデータ構造を用いることで、ストリーム通信を実現し、高い並列性を引き出す事に成功している[3]。

本システムでも既存のcons関数を改良したmcons関数によって、car部は確定しているがcdr部はまだ評価中である（未確定である）リストを実現することにより、リストのトップレベルでストリーム通信を実現している。

送信側は上に述べたmcons関数により評価の完了したcar部を送信して、その後cdr部を評価していく。受信側では受け取ったデータがリストであれば、リストはトップレベルで部分的に送信されてくるので、そのcdr部にストリームオブジェクトという仮の値を繋げて、オブジェクトが参照された時点で、残りの部分を受信する。本システムでは、このような方式でストリーム通信を実現している。

4. 実験結果

本研究では以下にあげるプログラムをセル1台による逐次処理、一括通信方式での並列処理、ストリー

¹LispのコンセルではなくAP1000のプロセッシングエレメントを指す

ム通信方式での並列処理による3通りの方式で実行し、それぞれ実行時間を測定した。

実験1 1から n までの自然数の2乗と3乗の和をリストにして返すプログラム。ここでは、1から n までの自然数を生成する関数(`naturals`)、リストの要素を2乗する関数(`square`)、リストの要素を3乗する関数(`cube`)、2つのリストの要素同士を足しあわせる関数(`cube`)を用意した。そして、並列処理で実行する際は、以上、述べた関数につきセルを1台ずつ割り当てた。

実験2 1から500までのリストの各要素に9を加えたものを6倍するプログラム。ただし、ここでは普通に計算するのではなく、自然数を生成する関数(`naturals`)、リストの要素に1を加える関数(`add1`)、6つのリストの要素を足しあわせマージする関数(`add1`)を用いて計算をすすめた。従って並列処理の場合は関数1つにつきセル1台を割り当て、合計61台(`naturals`6台、`add1`154台、`merger`1台)のセルを使用した。

実験1の結果を図1に示す。

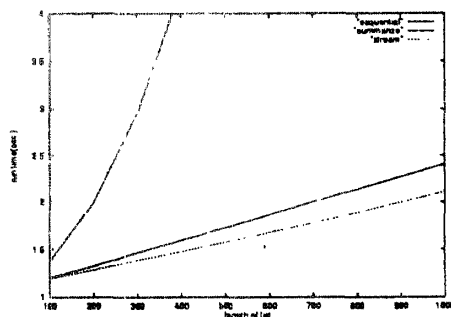


図1: 実験1の結果

図1より、ストリーム通信方式は一括通信方式と比較して処理速度が大幅に速くなっていることがわかる。これは受信側が、送信側でリストの既に評価された部分を受けとって評価を行い、それにより送信側と受信側で高度な並列処理が実現されているためである。

実験2の結果は表1に示す通りになった。なお実行時間は5回測定した平均値を取っている。

表1: 実験2の結果

ストリーム通信方式	一括通信方式	逐次処理
2.35710(秒)	24.28317(秒)	9.25376(秒)

この結果より、数十台のセルを協調させる場合、ストリーム通信方式では一括通信方式よりも大幅に効率が向上していることがわかる。

5. 結論

本研究では、分散メモリ上でリストの通信にストリーム通信を行なうことにより、パイプライン処理的な高い並列性を得られることが確認できた。

実験の結果から以下のことが示された。

- 通信するリストの長さに関わらず、ストリーム通信では一括通信方式よりも並列処理の効率は高い。
- セルを数十台使うような大規模なアプリケーションを実行する場合、ストリーム通信はセル1台の負荷に関わらず、並列処理の効率は高い。

以上のことより、大量のセルを使って、サイズの大きいリストを処理するような大規模なアプリケーションにおいて、ストリーム通信方式による並列処理は有効であるといえる。

6. 展望

本研究では、高い並列性を獲得できたが、常に理想的な並列処理が得られているわけではない。どんなに細かい粒度でも、より効率の良い並列処理を得るためには、通信単位の動的な切替えなど、改善すべき点は多い。

このような改善を行なうことにより、ストリーム通信を取り入れた並列度の高い並列処理言語の開発が今後、期待される。

参考文献

- [1] R. H. Harstead, Jr. *Multilisp: A Language for Concurrent Symbolic Computation*. ACM Transaction on Programming Languages and Systems, 7(4):pp501-538, Oct. 1985
- [2] R. Goldman, R. P. Gabriel. *Qlisp: Experience and Directions*. ACM, 1988
- [3] 瀧 和男 編, *bit 別冊, 第5世代コンピュータの並列処理*. 共立出版, 1993
- [4] 吉池 久夫. 分散メモリ上の並列 Lisp へのストリーム通信の導入. 慶應義塾大学 学士論文, 1997