Lispマシン SYNAPSEにおける並列Lispの実現

4Q-1

山内 雅彦

慶應義塾大学

1 はじめに

近年、マルチプロセッサ構成によるマシンの研究が盛んである。本研究室でも数年前にLisp処理系のためのマルチプロセッサ構成のマシンを作成し、そこで、様々な研究が成されてきた。本稿は、そのマシン上に並列処理Lispインタプリタをインプリメントした報告である。

2 SYNAPSE

SYNAPSEは、本研究室で作成されたマルチプロセッサ構成で密結合型のLispマシンである。その当時、リスト処理を高速化するため実記憶を16Mbyte(共有メモリ8Mbyte)実装した大規模なものであった。そのSYNAPSE上でのLisp処理系はSYNAPSE Lisp (v 1.0)と呼ばれ、インタプリタ、コンパイラ、エディタを備えたCOMMON Lispのサブセットである。本処理系は、SYNAPSE Lisp (v 1.0)を拡張して、並列処理を可能にしたものである。SYNAPSEのアーキテクチャは、リストプロセッシング専用のユニット(Lisp Processing Unit, L.P.U.)とガーベッジコレクション専用のユニット(Garbage Collection Unit, G.C.U.)が、バス・アービタを通して共有メモリに接続した形となっている。各々のユニットが共有メモリのバスに対して起こす競合は、バス・アービタによって調定されている。また、ガーベッジコレクションのためにG.C.U間で非可分なアクセスを実現したハードウェアスタック、印付け専用のタグブロックが存在する。

3 並列性の導入

3.1 プロセス

プロセスは、SYNAPSE上の最小の処理単位かつ資源単位である。SYNAPSE上でマルチプロセスを実現するためにMC6840を用いてタイマー割り込みをかけてタイムシェアリングを行なっている。全てのプロセスは、実行中、実行可、待ちのいずれかの状態にいる、実行可のプロセスは、ユニットの数だけ存在する。それ以外のプロセスは、それぞれ実行可のプロセスのための待ち行列か待ちのための待ち行列に存在している。プロセスコントロールのために以下の基本命令を用意した。

1 (suspend f)
この式を評価したプロセスを停止し、新しいプロセスを作り、関数f を呼び出す。その時、停止したプロセスのプロセス記述子が、関数fに引数として渡される。

2 (activate process-descriptor)
プロセス記述子を引数にとってそれを起動する。

3 (terminate)
この関数を評価したプロセスを消滅させる。

4 (replace-if-eq l v x)
lとxが等しい（eq）ならば、それをvで置き換える。置き換えられたらt、それ以外は、nilが返る。

3.2 future

本Lisp処理系では、上で述べた4つの基本命令の他にfutureと呼ばれる基本命令がある。futureはMultiLisp[Halstead 85]で初めて導入された構文である。futureとは、値がまだ確定していないS式のための仮の場所を意味する。従って(future X)を評価するとその仮の場所が返って来るとともに、新しいプロセスが生まれ、Xを並列に評価する。もし、あるプロセスがXの値を必要とし、かつXの値が確定していないとき、そのプロセスは待ちの状態になると共にfutureの持つ待ち行列に入る。futureの構造はfuture評価するS式、futureが確定したかどうかを示すフラグ、そのfutureのための待ち行列という3つ組からなっている。以後、その構造をfuture構造と呼ぶ。futureを評価する具体的な手順は、次のようになっている。

1 (future X)を評価。*future*（future構造）を作成。

2 新たにプロセスを作り、そのプロセスに(eval-future *future*) を評価させる。

3 eval-futureはfuture構造内の評価すべきS式を評価した後、確定フラグを立て、求まった値をfuture構造内に代入する。

4 もし、future構造内の待ち行列に待っているプロセスがあればそれらに起動信号を送る。

An Implementation of Multiprocessing-Lisp by the Lisp Machine SYNAPSE

Masahiko YAMAUCHI
Keio University

5 eval-futureのプロセスは、消滅する。

ここで、

futureの値が求まった後、futureに対して値をアクセスする時普通のデータ構造よりも一回間接が多い。このため、そのようなアクセスが初めて起きた時、リスト構造を書き換えるようにして、リスト処理の速度低化を防いでいる。

4 ガーベッジコレクション

4.1 L.P.U. 1個とG.C.U. 1個のとき

SYNAPSEでは、ガーベッジコレクションに並列G.C.のアルゴリズムを採用した。このアルゴリズムはmark&sweep方式で、次のような特徴を持っている。

- 印付けはリストたどり法により高速である
- L.P.Uのオーバーヘッドがほとんどない
- 印は、黒、白、灰白色の3色で表される
- 自由リストは印付けされない

G.C.はルート挿入フェーズ、印付けフェーズ、回収フェーズの3つのフェーズを繰り返す。ルート挿入フェーズでは、全てのルートが挿入されるまでL.P.U.はリスト処理を中断する。すなわちこの操作は非可分に行なわれる。詳しいアルゴリズムについては[寺村 88]を参考のこと。

4.2 L.P.U. n個 とG.C.U. n個のとき

拡張に当たって、問題になることは以下のことである。

- マーキングスタックに対するアクセス競合
- 自由リストに対するアクセス競合
- G.C.U.の仕事の効果的な分配
- 通信路の競合
- G.C.U.間の同期

[寺村 88]は、これらの問題について明確な解法を示して実際のインプリメントも行なった。

5 今後の展望

密結合型のマルチプロセッサマシンの持つ弱点は、共有メモリと各ユニットを結ぶバスの競合である。この問題を解決すると共に実記憶装置のアクセス速度を仮想的に速めるためにキャッシュを行なう必要がある。また、プロセスの増加によるデッドロックが有り得るので、プロセスコントロールをもっと強力にする必要がある。さらに、並列性を明示的に表示されていなくともそれを発見できる並列Lispコンパイラの開発が望まれる。

参考文献

[Halstead 85] Robert H. Halstead. Jr. : Multilisp: A Language for Concurrent Symbolic Computation. ACM Transactions on Programming Languages and Systems, Vol.7,No.4 ,October 1985, pp.501-538

[寺村 88] 寺村 信介 : 並列ガーベッジコレクションに関する研究,博士論文,慶應義塾大学院理工学研究科(1988)