2P－1

分散メモリ型並列計算機モデルに対応した言語処理系の開発

舟津 佳永　北海道大学工学部
山本 強　北海道大学大型計算機センター

1　はじめに

近年、並列計算機には、プロセッサ数が２～１６個のワークステーションクラスのものから、プロセッサ数が６４を越える大型計算機クラスのものまでがある。また、PVM の様にネットワークに接続された計算機群を並列計算機の各ノードとみなすタイプの実装も見られる。

ここで、現実的な利用環境を考えた際、高価な大型計算機クラスのハードウェアを使用することなく、並列プログラミングを実験することのできるプラットフォームがあると便利であると思われる。

本研究では、並列計算プログラミングモデルの実験台としての用途を念頭におき、ネットワークに接続された計算機群において各ノード間のオブジェクト管理を考慮した Lisp 型言語処理系の実装に関して報告する。

2　本研究の目標

本研究では、言語処理系として Lisp をインプリメントしている。一般に並列プロセスを言語処理系で扱う場合、宣言による明示的な並列化と、コンパイラ等による自動的な並列化の二種類が考えられるが、ここでは前者を採用している。

また、並列計算機モデルには、共有メモリ型アーキテクチャと分散メモリ型アーキテクチャがあるが、後述するオブジェクト管理方法によってどちらのタイプも表現可能とすることにした。

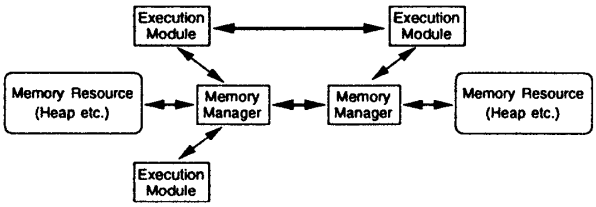

図 1: システム概念図

3　処理系の概要

処理系は、図１のような構造をしている。

実行部は Lisp 式の実行を行なう部分であり、１並行プロセスに対して１つが割り当てられる。実行部はそれぞれ他の計算機に置くことも可能である。

メモリマネージャは、ヒープなどのメモリ管理を行なう部分であり、実行部からのヒープ等へのアクセスは、このメモリマネージャを通しておこなわれる。メモリマネージャは、ヒープ等の Lisp オブジェクト資源群にそれぞれ対応するように置かれ、共有メモリモデルを表現する際には複数の実行部が一つのメモリマネージャにアクセスする形になり、分散メモリモデルの場合にはそれぞれの実行部が別々のメモリマネージャにアクセスする形となる。このことで、どちらのメモリアーキテクチャに対しても対応することができるようになっている。

Implementation of programming language for parallel machine with local memory.
Yoshihisa Funatsu, Tsuyoshi Yamamoto, Hokkaido University

並列プロセスの実行は、(make-process function ...) のように明示的に宣言することによって行なう。この時、同時にメモリマネージャへの接続指示や、プロセス間の関係の指示を行なう。プロセス間の関係には、従属関係もしくは独立プロセスかを指定し、従属関係を指定した場合は、派生させたプロセスが全て終了するまで実行を止める wait 機能を有効にする。並列プロセス実行中は、生成したプロセスの管理とオブジェクトの管理を行なうために、各実行部とメモリマネージャは必要に応じて通信を行なう。

4 オブジェクト表現方法

分散メモリ型並列計算機アーキテクチャにおいて Lisp の様な言語を動作させる場合、各オブジェクトの管理は非常に重要な問題である。そこで本研究では、オブジェクトを図 2の様に表現する。

各オブジェクトは、型情報を収める 8bit の Tag field と、24bit の Obejct ID を持つ。Object ID は、上位 6bit をメモリマネージャ判別用に使用し、下位 18bit をそれぞれのメモリマネージャ内でのオブジェクト識別用として使用する。これら Object ID は、オブジェクト生成時に与えられ、生成される各オブジェクトに 1 対 1 で対応する。これによって、異なるヒープ上に置かれたオブジェクトに関しても同一オブジェクトであるかどうかの判定を行なうことができる。

メモリマネージャは、オブジェクトの格納される実アドレスとの対応をとるために、Object ID と実アドレスとの対応表を持っており、実行部からのアクセス要求に対して排他制御などの処理を含めて対処する。分散ヒープ型の並列プロセス作成時には、必要なオブジェクトのコピーを行ない、各メモリマネージャの輸入・輸出オブジェクト表を変更・参照することによって、異なるヒープに存在する同一オブジェクトの整合性を管理している。

また、Lisp の場合にはガーベージコレクションの問題を扱わなければならないが、本手法においては、通常、オブジェクト ID と実アドレスの対応表を書き換える形で GC を行ない、Object ID の空きがなくなってきた場合には、GC 時に Object ID の付け直しを同時に行なうことにしている。

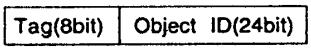

図 2: オブジェクト表現

5 まとめと今後の課題

本研究では、ネットワークで結ばれた計算機群上で各ノード間のオブジェクト管理を考慮した Lisp 処理系を実装した。

今回の手法では、オブジェクトの整合性の管理と GC を比較的容易におこなうために、オブジェクトにアクセスする際には、メモリマネージャを介して行なわなければならない。この部分のオーバーヘッドを軽減することが処理系としての性能に大きく響いてくるので、オブジェクトのローカル性とグローバル性に応じてオブジェクトの表現方法とアクセス方法を変更するなどの方法で最適化を行なうと良いと思われる。

また、実験用プラットフォームとしての使用を考慮すると、動作時にノード間の通信量等の情報を記録できるようにし、処理実行における細かい情報をわかりやすく提供することができる様にすると、より現実的なものにすることができるであろう。

参考文献

[1] Guy L.Steel Jr. et al., 井田昌之訳, “Common LISP,” , 共立出版株式会社

[2] SunSoft, Inc.,“Beyond Multiprocessing... Multithreading the SunOS Kernel,” *Summer '92 USENIX* June 8-June 12, 1992