5U-8

PIE64のゴール間同期／分散ガベージコレクション支援機構

清水 剛, 島田 健太郎, 許 魯, 小池 汎平, 田中 英彦
東京大学 工学部 *

1 はじめに

現在、我々の研究室では、並列推論マシン PIE64[1] の開発をすすめている。PIE64 は 64 台の推論ユニット (IU) が 2 系統の自動負荷分散機構付き多段ネットワークによって結合された構造を持ち、並列論理型言語 FLENG を実行する。

各 IU には、IU 内部と相互結合網とのインタフェスを行なうネットワーク・インタフェス・プロセッサ (Network Interface Processor: NIP) が用意されるが、NIP は 2 つの IU 間のデータ転送以外に、FLENG のプロセス間同期や PIE64 の一括ガベージ・コレクションをハードウエア・レベルで支援する機能をもち、PIE64 で行なわれる並列推論処理のうち、並列処理機能を支援する。

本稿では、NIP が提供する、プロセス間同期及び PIE64 の分散メモリにおける一括ガベージ・コレクション支援の機能について述べる。

2 コマンドの形式

IU 内部の各プロセッサ (管理プロセッサ SPARC 、推論プロセッサ UNIRED、NIP) は高速双方向のコマンド・バスを通じてコマンドのやり取りをし、協調動作をするが、この通信は、N ワードのコマンドに対して M ワードのリプライが返る形でおこなわれる ($N \geq 1, M \geq 0$)。

コマンドのタイプは、コマンドの第 1 ワードの上位 2 bit と下位 2 bit を使用して区別される。この 4 ビットは、PIE64 のポインタ型データでのタグ及びガベージ・コレクション時のマークビット用の領域にあたる。特に、転送コマンドでは、タグのビットパターンがそのまま転送サイズに対応している。

コマンド / リプライ・フォーマットの例として、後に述べる bind コマンドを図 1 に示す。

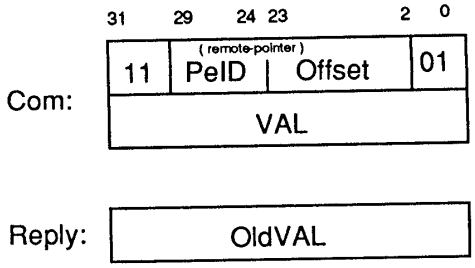

図 1: bind コマンドのコマンド / リプライ・フォーマット

3 プロセス間同期のコマンド

PIE64 では、並列論理型言語 FLENG が基本言語として用いられる。FLENG の持つプロセス間同期機構を以下の 3 つの処理が支援する。

- `oldval = bind(var, val)`
- `suspend(var, suspendID)`
- `activate(suspendID, val)`

各 IU には、1 系統のネットワークにつき、ネットワークへの接続要求を出すマスタ NIP と、ネットワークからのコマンドにより動作するスレーブ NIP の 2 つの NIP が用意されるが、これらの動作は未束縛論理変数の実体が置かれている側の IU のスレーブ NIP によって主たる処理が行われる。これらのコマンドを受けたマスタ NIP は受け取ったコマンドをネットワークを介してリモート変数のある IU に送信し、bind 処理の場合には結果を受け取る。

`bind` は 2 ワードのコマンドに対して、1 ワードのリプライが返る処理で、未束縛論理変数の `var` に値 `val` の束縛を試み、成功すれば、`var` によってサスペンドされていたゴールをアクティベイトする。リプライは変数 `var` から読みだした値で、これが UNDEF であるかどうかを調べることで、`bind` の成否を確認できる。`bind` が成功した場合には、サスペンション・レコードの解放をするが、これはそのレコードに登録されていたゴールが自 IU にある場合は SPARC に、他 IU にある場合は同一 IU のマスタ NIP に `activate` コマンドの発行をしつつ、サスペンション・リストに使われていたリスト・セルを回収することで行われる。この動作をするスレーブ NIP においては、activate 動作がメインのシーケンサではなく、別系統のシーケンサによって行われるため、activate 動作の実行中に別のコマンドを受け付けることが可能である。

`suspend` では、サスペンドを起こしたゴールの識別子 (`suspendID`) を、その原因となった未束縛論理変数のサスペンション・レコードに登録する。この処理では、サスペンション・レコード・リストの消滅を防ぐためにまず変数をロックし、フリー・リストからセルを取り出し、`suspendID` の登録をし、変数のロックを解除する。

上記のコマンドでは、`var` がリモート変数の場合にはマスタ NIP に対してコマンドを送信する。自 IU 内にある未束縛論理変数に対して同様の処理をする場合は、自 IU 内のスレーブ

*Support Mechanism for Inter Goal Syncronization/Distributed Garbage Collection of PIE64
Takeshi SHIMIZU, Kentaro SHIMADA, Lu XU, Hanpei KOIKE, Hidehiko TANAKA, the University of Tokyo

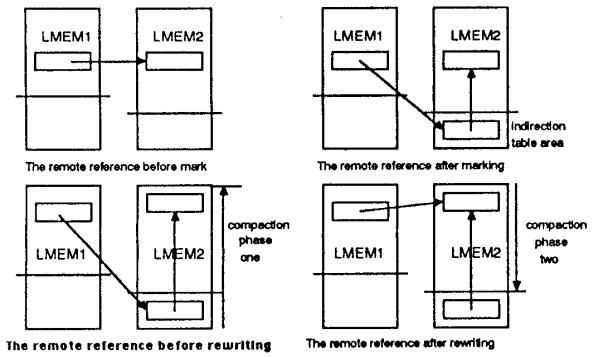

図 2: mark / restore 動作

NIP にこのコマンドを発行することで、スレーブ NIP の持つサスペンション・レコードの管理機能を利用できる。

activate コマンドをマスタ NIP が受け取ると、アクティベイトすべき suspendID のある IU のスレーブ NIP と接続し、suspendID と論理変数への束縛値をネットワークを通じて転送し、さらにそのコマンドをその IU の SPARC に発行する。SPARC は指定されたゴールを自 IU のアクティブ・ゴール・キューに登録する。

任意の IU において、その IU 内のサスペンション・レコードは、リストの形でスレーブ NIP によって管理される。スレーブ NIP は使用可能なフリーリストを指すポインタ・レジスタを持ち、suspend 動作時には、フリーリストからの切り出しをし、bind-activate 動作時には、セルの回収を行う。

このサスペンション・レコードのためのフリー・リストは、システムの初期化時に SPARC によって用意され、newlist コマンドによってスレーブ NIP に与えられる。

4 ガベージ・コレクション支援のコマンド

PIE64 では、ページ単位のローカルなメモリ管理と、64 台全体で行う一括型のガベージ・コレクションを併用している [2]。

NIP は PIE64 の分散メモリの一括ガベージ・コレクションを支援するため、2 つの機能を有する。1 つは、他 IU 内のオブジェクトをマーキングするリモート・マークの機能であり、他の 1 つは、コンパクション・フェイズにおいてのリモート・ポインタの書き戻しである。

マスタ NIP が 受け付ける GC 支援のコマンドは以下の通りである。

- マーキング・フェイズ用: mark1, mark2, markn
- コンパクション・フェイズ用: restore

4.1 マーキング・フェイズのコマンド

マーキング・フェイズでは、NIP はリモート・ポインタのマーキングを支援する。

マーキングのためのコマンドは、2 ワードで構成される。

- mark[1,2,n](remote-ptr, location)

このコマンドでは、リモート・ポインタを相手側 IU の間接参照テーブルに登録する 2。1、2、n はリモート・ポインタの指すオブジェクトのデータタイプをしめす。この動作手順は以下の通りである。

- mark コマンドを受け取ったマスタ NIP はネットワークに接続要求を出し、相手側 IU のスレーブ NIP と接続する。
- マスタ NIP は要求の元となったマークの対象となるリモート・ポインタを送出する。
- スレーブ NIP は受け取ったリモート・ポインタを間接参照テーブルに登録しマークを付け、このテーブルのエントリ・アドレスをマスタ NIP に送り返す。
- さらにスレーブ側では、新しいエントリが増加したことを自 IU の SPARC へ送信し、マーキングの継続を依頼する。
- ネットワークを通して、間接参照テーブルのエントリアドレスを得たマスタ NIP は、もともとのリモート・ポインタをこのエントリ・アドレスで置き換え、マークを付けておく。

間接参照テーブル用の領域は、マーキング・フェイズを実行する前に、SPARC によって用意され、スレーブ NIP に対して newpage コマンドにより与えられる。

4.2 コンパクション・フェイズのコマンド

コンパクション・フェイズでは、リモート・ポインタの書き戻しを、相手側の間接参照テーブルを参照して行う (図 2)。この書き戻しは、コンパクション・フェイズの第 2 フェイズに行われる。この処理のためにマスタ NIP の受け取るコマンドは、

- restore(remotepointer, location)

である。このコマンドでは、マスタ NIP は、remotepointer で指された、相手側 IU の間接参照テーブルのエントリを読みだし、その内容をローカル・メモリのアドレス location に書き込む。

5 おわりに

本稿では、PIE64 において NIP が提供するプロセス間同期、及び、分散ガベージ・コレクション支援の機能について述べた。今後の課題としては、ゲートアレイ完成後の実際の性能評価、PIE64 のネットワークとの接続試験があげられる。

参考文献

[1] 小池, 田中, “並列推論マシン PIE64 の概要”, 情報処理学会第 37 回全国大会5N-4, Sep. 1988.

[2] Lu Xu and Hidehiko Tanaka: *Distributed Garbage Collection for the Parallel Inference Machine: PIE64*, 本大会.