推論向き高並列計算機システムのアクティビティ制御機構

4N-5

丸山　勉，相田　仁，後藤　厚宏，田中　英彦，元岡　達

（東京大学　工学部）

1．はじめに

本稿では高並列推論エンジンＰＩＥ［１］の第一次モデルにおけるアクティビティ制御機構の概略について述べる。

2．ゴールフレーム間並列処理におけるアクティビティ制御

アクティビティ制御の目的は、①並列処理を効率化してより早く解を求める、②資源の枯渇を防止する、③論理型言語の拡張機能を実現する、の３つである。

これらの目的に沿ったアクティビティ制御機構の役割は以下の通りである。

ａ）多数のゴールフレームの中からＵＰにおいて実行するものを選択する。

ｂ）選択されたゴールフレームをシステム内の負荷が均一になるように物理的なＵＰへ割り付ける。

ｃ）割り付けられたゴールフレームの中からユニフィケーションを実行するリテラルを選択する。

ｄ）現実的な問題解決プログラムの記述に必要な言語上の拡張機能を実行する。

ｅ）ユニフィケーションの失敗等によって不要になったデータを消去し資源を回収する。

ここでａ）は、探索ストラテジ（プロセッサ数の幅を持った縦型探索／横型探索）に基づく探索の深度による制御である。一方、ｄ）はif～then～elseに類する制御構造や履歴依存性の記述等、逐次的な実行順序の制御である。この両者を実現する場合は各ゴールフレームの相互関係を明確に把握しておく必要がある。また、資源の回収（ｅ）において、逐次システムにおける知的後戻りのように失敗経験を積極的に利用する場合も、ゴールフレーム間の関係が重要になる。ｃ）は、前述の目的①，②の点において大きな意味を持つが、ゴールフレーム間の並列処理においては、ＵＰにおける付加機能として考慮することにする。ただし、リテラル間の並列処理を考える場合はアクティビティ制御の役割りとなる。

3．ＰＩＥのアクティビティ制御機構

前節において、ゴールフレーム間の相互関係を把握することが、アクティビティ制御の基礎となることを述べた。

論理型言語によるプログラムの実行は、探索木におけるノードとして捉えることによって各親子関係が明確になる。また、ゴールフレーム間の並列処理では探索木の各ノードがＯＲノードとなるが、ノードの属性を追加することによって探索ストラテジや言語上の拡張に対応することができる。そこで、ＰＩＥではゴールフレームの実体を保持するMMに付随してActivity Controller（ＡＣ）を設け、多数台のＡＣによって探索木情報を保持し、ＡＣ相互間のコマンドによってアクティビティ制御を行なう。

4．探索木情報によるアクティビティ制御

（１）探索木のノード情報とコマンド

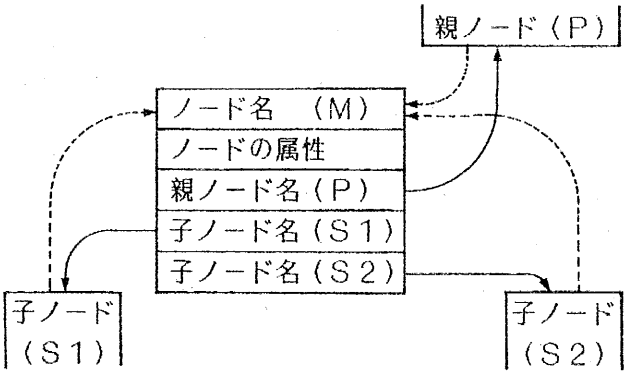
実行途中の探索木は、根ノード・中継ノード（基本的にＯＲノード）・末端ノード（成功／失敗ノード）・ゴールノード（ゴールフレームに対応するノード）によって構成されていると考えてよい。

ＰＩＥでは分散した多数のＡＣによって探索木情報が保持できるように、図１ようなノード情報を管理する。

各ＡＣは、ゴールノードの伸長・失敗ノードの刈込み・言語上の制御構造に応じた実行管理を、各ノード情報が有する属性とノード間のコマンドを用いて実現する。各コマンドは

＜宛先ノード名、コマンド名、引数…＞

という型式を持つ。



ノード名＝＜ノードを保持するＡＣ名：識別名＞

ノードの属性＝＜基本的にはＯＲ＞

［図１］　ノード情報

（２）ゴールノードにおける探索木の伸長

ゴールフレームに対応したゴールノードはＵＰによって実行が進められ、新たなゴールフレームが生成される。この一連の操作はＵＰとＡＣ間で送受される以下のメッセージによって表現できる。（図２）

① ＲＥＳＯＬＶＥ（Ｍ）…ノードＭを実行せよ

② ＥＮＤ－ＲＥＳＯＬＶＥ（Ｍ，ｎ）…ノードＭの実行結果、ｎ個のゴールノードが生成された

③ ＮＥＷＧＦ（Ｍ）…ノードＭの実行結果

以上のような探索木の伸長の操作では、コマンド

＜Ｍ，son ，Ｓ＞

を新ゴールノードＳから旧ゴールノードＭへ送ることによってＭ－Ｓ間に親子関係を結ぶ。旧ゴールノードは、すべての子ノードとの関係を結ぶと中継ノードとなる。ＵＰによる実行がすべて失敗した時は末端ノード（失敗ノード）となる。

（３）末端ノードにおける探索木の刈込み

末端ノード（特に失敗ノード）の多くは探索木情報として不要となり、資源の節約のために刈込まれる。失敗ノードＦは親ノードＰに対して、コマンド

＜Ｐ，fail，Ｆ＞

を送り自分自身（ゴールフレームの実体を含めたノード情報）を消滅させる。＜fail＞を受けとった親ノードは対応する子ノードをノード情報から削除する。すべての子ノードから＜fail＞が来た場合は自分自身を失敗ノードとして回収する。

（４）中継ノードにおける不要ノードの削除

子ノードがただひとつになった中継ノードは多くの場合不要ノードとして削除が可能である。不要中継ノードの削除はノード間の高速なコマンド伝搬を行なう上で重要であり、また再帰を用いた無限ループ等では無数のノードが生成されるため、資源の枯渇を避けるためにも必要となる。

ノード間のコマンド授受が非同期的に実行される環境のため、以下の３種のコマンドとそのノードが削除中であることを示す属性＜削除ノード＞を導入する。

①親がＭからＰに変更されたことを子Ｓに知らせる

＜Ｓ，parent－change，Ｍ，Ｐ＞

②子がＰ２からＭに変更したことをＰ１に知らせる

＜Ｐ１，son-change，Ｐ２，Ｍ＞

③親子の変更操作の終了を削除ノードに知らせる

＜Ｍ，changed ＞

不要となった中継ノードは＜parent－change＞を子に送り自分自身は削除ノードとなる。＜parent－change＞を受けとったノードはノード情報を変更し、新しい親に対して＜son-change＞を送る。＜son-change＞を受けとったノードはノード情報を変更した後、＜changed ＞を削除ノードへ送る。削除ノードは＜changed ＞を受けとった後、消滅する。

この一連の操作の間、削除ノードは親からのコマンドのうち＜changed ＞以外はすべて子ノードへパスする。以上の手続きによって非同期的な不要中継ノードの削除が可能である。（図３）

（５）制御操作とActivity Controller

アクティビティ制御は（２）～（４）で述べた他に、子ノードの実行を一時停止するコマンド・実行を再開するコマンド・子孫のゴールノードを実行せずに強制的に失敗ノードとするコマンド等と、数種のノードの属性を用意することによって実現する。

また上記において、各コマンドはノード間で送受信するとして述べたが、実際のシステムではノード情報を管理する多数台のＡＣ間で送受される。

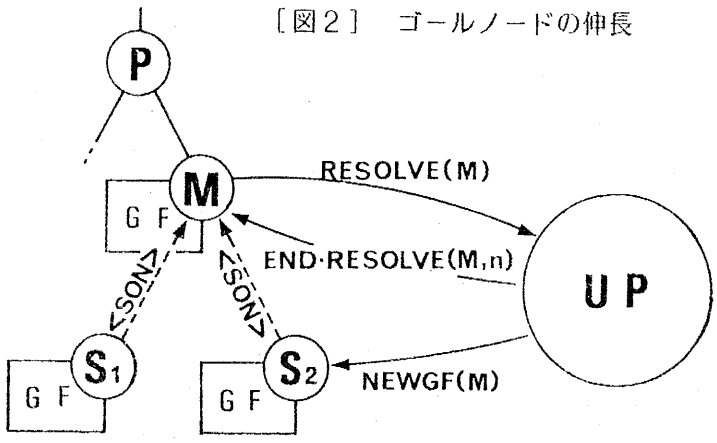
５．おわりに

今後は、ＰＩＥの機械語の機能にあわせてノードの属性とノード間のコマンドの詳細を決定していく予定である。

＜参考文献＞

［１］ 後藤 他：“推論向き高並列計算機システムのアーキテクチャ”，本予稿集 ４Ｎ－４

［図２］ ゴールノードの伸長



［図３］ 不要中継ノードの削除