ＰＩＥにおける並列論理型言語ＦＬＥＮＧの実行方式

5B-4

垂井　俊明、　丸山　勉、　田中　英彦

（　東京大学　工学部　）

１．はじめに

高並列推論エンジンＰＩＥ[1] はゴール書き換えモデルに基づき、論理型言語を高並列に実行する推論マシンである。現在迄のところ、ＰＩＥにおける並列実行は、OR並列を主体に検討が進められて来た。従って、現在のＰＩＥのアーキテクチャそのままでは、ＧＨＣを始めとする並列 AND論理型プログラミング言語を効率良く実行することは難しい。ＰＩＥにおいてＧＨＣ等を実行する場合、問題になるのは次の２点である。

① 共有変数への効率的アクセスと負荷分散

② ゴール間の実行制御

ここではまず上記の①についてソフトウエアシミュレーションにより評価、検討を行なうことにする。共有変数については、現在のＰＩＥで用いられている集中型の構造メモリ（ＳＭ）[2] を利用することも可能であるが、アクセスの集中等の問題が生じる可能性が大きいので、集中型に代わり分散型のＳＭを導入する。各ＳＭは、単一化プロセッサ（ＵＰ）と対になっており、ＵＰは対となっているＳＭに高速にアクセスすることができる。シミュレーションの対象とする言語については、ＧＨＣでは実行制御等の問題のために、①のみに対する評価が明確でなくなるので、ここでは実行制御を持たない並列言語ＦＬＥＮＧを対象とすることにする。

２．論理型言語ＦＬＥＮＧ

ＦＬＥＮＧ[3] はＧＨＣと良く似た言語であり、以下の点がＧＨＣとの主な相違点である。

① ガードが存在しない。従って定義節のヘッドリテラルの単一化が成功するとすぐにコミットされる。

② ボディ部の各リテラルは論理的 AND関係にない。従って、全てのボディ部のリテラルは必ず実行され、あるゴールが失敗しても他のゴールには影響を与えない。論理的な AND関係が必要な場合は、プログラム中に明示的に書く必要がある。

ＦＬＥＮＧにおいてもＧＨＣと同様に、ヘッドリテラルの単一化を行なう場合親ゴールの変数を具体化することはできず、この単一化はサスペンドする。ＦＬＥＮＧのより詳しい説明は文献[3] に譲る。

以上述べたように、ＦＬＥＮＧはＧＨＣと共有変数に対するアクセス等については同一であるが、実行制御の部分は大幅に簡略化されていることがわかる。また、ＦＬＥＮＧは低レベルな言語であるが、ほとんどのＧＨＣのプログラムはＦＬＥＮＧに容易にコンパイルすることができ、すでにコンパイラも完成している。

３．システム構成

図１に今回シミュレーションを行なうシステムの全体構成を示す。単一化等の処理を行なう単一化プロセッサ（ＵＰ）とＳＭの対が基本処理要素である推論ユニット（ＩＵ）を構成し、ＩＵが多数台並列に接続されてシステム全体を構成している。システムにはＵＰ間を結び、ゴールフレーム（ＧＦ）の分配等を行なう分配網（ＤＮ）、ＳＭ間を接続し共有変数の値や構造データ等をやりとりする構造メモリ網（ＳＭＮ）の２種類のネットワークがある。ＤＮについては、従来と同様多段結合網を用いる[4] 。ＳＭＮについては現在のところどのような構成が最適であるかわからないため、とりあえず理想的な回線交換網でシミュレーションを行ない、その結果により具体的な構成を決定する予定である。また同一

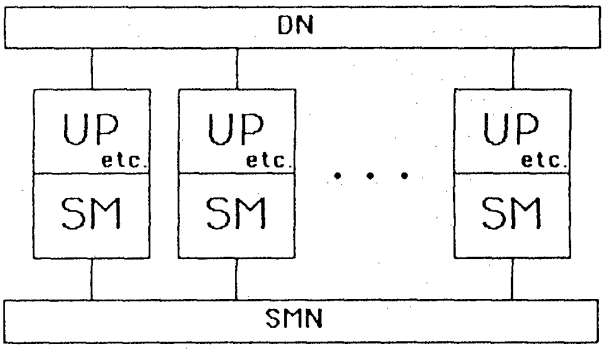

図１　システムの全体構成

Execution Method of FLENG Prolog in PIE

Toshiaki Tarui, Tsutomu Maruyama, Hidehiko Tanaka

University of Tokyo

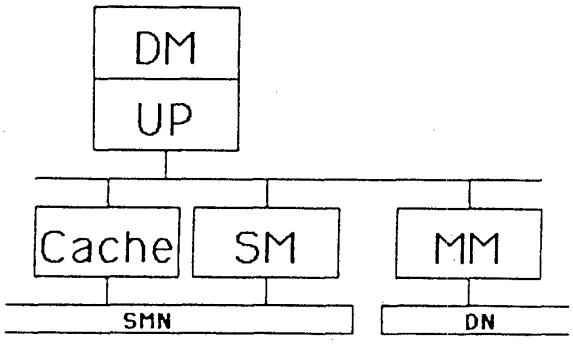

図2　IUの内部構成

IU内のUPとSMは密に結合されており、SMNを通さないで直接アクセスすることが可能である。これにより、負荷分散をうまく行なえばSMNへのアクセスの回数を減らすことが可能になる。

図2に推論ユニットの内部構成を示す。各部の機能は以下の通りである。

・DM（定義節メモリ）
　プログラムを保持する。
・UP（単一化プロセッサ）
　単一化を行なう。UPからはDM、MM、Cache、及び対となっているSMの内容を直接アクセスすることができる。また、他の全てのUPがbusyな場合は、新しいゴールの生成を行なわずに内部のスタックを使用して逐次処理を行なう。
・MM（メモリモジュール）
　PIEの基本処理単位であるゴールフレーム（GF）の記憶、管理、他のIUとの間のGFの転送処理等を行う。今回のモデルでは、GFはリテラル毎に分割されている。MM中には、active queueと suspended queueの2つのGFのキューがあり、単一化がサスペンドしたGFは suspended queueにつながれ、アクティベイトされるのを待つ。
・SM（構造メモリ）
　共有変数及び構造データを格納する。ここで、GFのサスペンド及びアクティベイトの処理は以下のように行なわれる。
① UPがヘッドリテラルの単一化中に親ゴールの undefの変数を具体化しようとした場合、GFはMM中でサスペンドする。
② それと同時にサスペンドしたGFのMM中のアドレスがSM中の undefのセルのところに記憶される。
③ その後 undefのセルに値がバインドされると、SMはサスペンドしていたGFにアクティベイトコマンドを送り、GFは再び実行可能になる。

システム述語unify において、もしUPが書き込もうとしたセルに、既に値が代入されていた場合は、SMはその値をUPに送り返し、UPは書き込もうとした値との単一化を行なう。

・Cache(キャッシュ)
　他のIUのSMの内容のキャッシュである。UPが他のSM中のデータをアクセスする場合は、先ずCache の内容を調べ、該当するエントリがない場合にはSMNを通じて読み出しが行なわれ、読み出された値はCache に登録される。また、他のSM中のundef のセルについても、1つのGFの単一化が終了するまでここに記憶される。

4．おわりに

現在以上のような方針でソフトウエアシミュレータをUnix上のC言語で製作中である。今後の予定としては

・SMNを経由した共有変数へのアクセスの頻度の測定
・SMNの構成の検討
・負荷分散方式の検討

等があげられる。また、以上の結果をもとに

・コントロールも含んだGHCの処理系のシミュレーション
・現在のPIEと合わせた新しいモデルの検討

等を行なっていきたい。

＜参考文献＞

[1] Moto-oka, T, Tanaka, H, et al , "The Architecture of a Parallel Inference Engine －PIE－", FGCS´84, ICOT.

[2] 平田、田中、元岡：　"PIEにおける構造メモリの構成について"、情報処理学会、アーキテクチャワークショップインジャパン '84、1984.

[3] Nilsson, M, Tanaka, H, "－FLENG Prolog－ The Language which turns Supercomputers into Parallel Prolog Machines", The Logic Programming Conference '86, ICOT.

[4] 坂井、田中、元岡：　"動的負荷分散を行なう相互結合網"、信学技報、1985年 8月.