# ネットワーク型分散プロダクションシステム

3W−2

高原[1]　利之　　，吉田[2]　広行
インテリジェントシステム研究所，東京工芸大学

## 1．はじめに

近年，人工知能におけるプロダクションシステムの応用において，リアルタイムに問題を解く必要が増大している．また一方では，密結合型マルチプロセッサ・ワークステーションが，ネットワークを介して他のワークステーションと結合された疎・密結合混在型のマルチプロセッサシステムが一般的になりつつある．しかし，従来のプロダクションシステムの高速化技法において，このような疎・密結合混在型の分散環境を前提としているものは少ない．そこで本研究では，疎・密結合混在型の分散環境下で，個々のエージェントが負荷を均一化しつつ，互いに協力しあいながら推論する方式を提案する．

## 2．動的負荷分散

前述の，密・疎結合混在型のマルチプロセッサ・システムで効率的に処理を実行するためには，動的な負荷分散，すなわち各ホストの負荷状況に応じて負荷の移動をダイナミックに行う機構が必要である[2]．

ここでは，分散単位として，生成・消滅のオーバーヘッドが UNIX のプロセスに比較して非常に少なく，実行の切り換えも高速で行えるスレッドを用いた．しかし，スレッドはタスク内に存在しなければならないため，ネットワーク上でスレッドのみを移動させることはできない．従って，疎結合型マルチプロセッサシステムでスレッドを移動させながら負荷を分散させる特別な機構を次に示す．

## 3．スレッド移動

まず，前処理として，LANで結合された複数台のホストの各々に，問題解決を行うタスクのコピーを生成する．このタスクには，あらかじめ問題解決に必要な知識と解決機構が含まれているものとする．

次に，ネットワークを介した複数のタスク間でポート（Machカーネルが管理する通信路）を共有するために，各タスクが１つのポートを生成し，そのポートをネットワーク・ネームサーバーに登録する．ここで，ネットワーク・ネームサーバーとは，ネットワーク上でのポートの共有を行うために，各ホスト毎にポートとそれに対するグローバルな名前のペアを管理するサーバーで，Mach が提供しているものである[3]．

これにより，各ホスト上に相互の通信用ポートを持った同一環境のタスクをつくることができるので，それらを合わせて１つの均一なタスク環境と見なすことができる．従って，処理のモデルとしては，１つのタスク環境の中で，複数のタスクが動きまわるという，通常のタスクとスレッドの関係と同一である．ただし

---

Distributed Production System for Network
Toshiyuki Takahara,Hiroyuki Yoshida
1 Intelligent System Lab.,2 Tokyo Institute of Polytechnics

1つのタスク内にある複数のスレッド間での切り替えは，Machが行うが，1つのホスト上のタスク内にあるスレッドが，他のNeXT上のタスク内に移動する機構はMachのメッセージ機構を用いて実現した．

## 4．分散プロダクションシステム

分散プロダクションシステムは，個々にプロダクションシステムを実行するエージェントの集合が，協調して問題解決を行うシステムである．前述の負荷分散機構のもとで，以下のようにして分散プロダクションシステムを実現した．

まず，エージェントとしてスレッドを用いる．系全体でのエージェントの数は不変とする．各エージェントが参照するルールベースは均一タスク環境内にグローバルな共有情報として置かれる．但し，各ルールはあらかじめエージェントの数に応じて重複なしにセグメントに分割され，各セグメントがエージェントに割り付けられているものとする．各エージェントは，個別にワーキングメモリを持ち，それは，スレッドの持つ局所スタック内に取られる．更に，各スレッドは他のスレッドと協調動作をするために，自分の関係するルールとデータ依存関係があるルールに関係する全てのエージェントとの間に，ポートを持ち，ワーキングメモリの更新データの送受信を行う．

負荷に応じてスレッドの移動が起こるが，これは拡張された認知－実行サイクルのどの段階で行われてもよい．

## 7．処理例

本稿における環境としては，密結合のマルチプロセッサシステムがLANで結合された，疎・密結合混在型マルチプロセッサシステムを想定しているが，実際には，Machが稼働しているNeXTコンピュータ（シングルプロセッサ）2台をイーサネットを介して接続したシステムでプロトタイプ．例題として，医療診断システム（約50ルール）を実行し，1台に負荷をかけところ，時間の経過とともに2台の負荷を平均化する方向にスレッドの移動が適度に起こることが観察された．

## 8．まとめ

ポートを共有する複数のタスクからなる均一なタスク環境のもとでのスレッド移動の機構を，拡張されたforkを用いて実現し，それを用いて負荷分散プロダクションシステムを実現した．これにより，単にプロダクションシステムを高速化するのみならず，負荷集中時に負荷を分散しつつ，応答速度を一定に保ちリアルタイム性を確保すること，および分散されたハードウェア資源を有効に活用することができる．

## 参考文献

[1] 増井庄一，田野俊一，「プロダクションシステムの高速化」，人口知能学会誌 Vol. 6, No. 1, pp38-45, 1991.

[2] Filman, R. E. and Friedman, P. "Coordinated Computing," McGraw-Hill, 1984.

[3] "Preliminary 1.0 System Reference Manual," NeXT Inc., 1989.

[4] Rashid, R. F. Threads of a New System, UNIX Review, August, 1986.