4Q−12 NeXT上の分散画像生成方式について

吉田広行*　高原利之**
*東京工芸大学/MeCインテリジェントシステム研究所　**東京家政大学

1．はじめに

近年，密結合型マルチプロセッサ・ワークステーションにより並列処理を行い，高速に画像を生成することが盛んに行われている[1]．一方では，これらのワークステーションは，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ていることが多い．しかも，最近では，1つの高速ネットワークに結合されたワークステーションが100台近いものも珍しくなくなってきている．即ち，現在は，密・粗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[2]．

本稿では，このような密・粗結合混在型のマルチプロセッサシステム上で効率的に負荷分散を行い，画像を高速生成する1方式について報告する．また，Machオペレーティングシステム搭載のNeXTコンピュータでの実装例についても述べる．

2．動的負荷分散[2]

前述の，密・粗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．

UNIXでは，資源の保護単位とCPUの利用単位がどちらもプロセスであるため，動的な負荷分散をプロセス単位で行うことが多い．すなわち，実行中のプロセスをあるホストから他のホストへ移動させることが多い．しかし，この方式では，多数のプロセスの生成・消滅を繰り返すアプリケーションにおいては，プロセス生成・消滅のコストが大きいこと，および負荷分散を実現するためにプロセスをネットワーク経由で転送すると通信コストが高いことなどの欠点があり，高い効率は望めない．

一方，Mach分散オペレーティングシステム[3]では，資源の保護単位とCPUの利用単位を切り離して，前者をタスクに，後者をスレッドに割り当てている[4]．スレッドは計算に必要最小限の情報しか持たないため，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，また複数スレッド間の切り換えも高速で行える．さらに，1つのタスクの中で複数のスレッドを同時に走らせることができ，それらはタスク内の全ての資源を共有する．そのため，共有メモリを持つ密結合型マルチプロセッサシステムでの並列処理に有効である．

上述のように，スレッドは，少ないオーバーヘッドで生成・消滅・切り換えができるため，動的負荷分散における分散の単位として適当と考えられる．しかし，スレッドは必ず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．したがって，粗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際に，なんらかの特別な機構が必要となる．

そこで次節では，均一なタスク環境のもとでプロセス分岐(fork)の拡張としてスレッド移動を実現する方法を示す．以下において，各ホストでの処理は，原則としてシングルタスク・マルチスレッドで行われるものとする．

3．スレッド移動

まず，前処理で，LANで結合された複数台のホストの各々に，画像生成処理を行うタスクのコピーを生成する．このタスクには，あらかじめ画像生成処理に必要なプログラムとデータ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 OSが提供しているものである[3]．

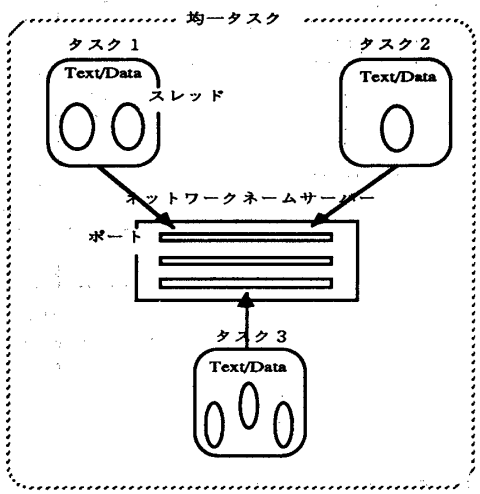

図1　均一なタスク

On a Distributed Image Synthesis Method based on the NeXT Computer
Hiroyuki Yoshida*, Toshiyuki Takhara**
*Tokyo Institute of Polytechniques, **Tokyo Kasei University

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（図1）．したがって，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，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 OS 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，以下のように実現した（図2）．

（１）負荷の少ないホストを選定
（２）新たなスレッド(スレッド１)をそのホスト上に生成
（３）移動するスレッド(スレッド０)の状態をメッセージを用いてポート経由でスレッド１へ転送
（４）スレッド１の状態にスレッド０の状態をコピーする．
（５）スレッド０を消滅させる．

なお，上記(1)(2)の処理は，１つのホスト上の特別なタスクが行うものとする．また，(1)の処理に際するグローバルな負荷情報は既に解っているものとする．

この処理は，通常のforkでローカルな子スレッドを生成する代りに遠隔子スレッドを生成することとに相当する．ただし，子スレッドを生成するのは親スレッドではない点およびメッセージを用いて状態のコピーを行う点が異なる．

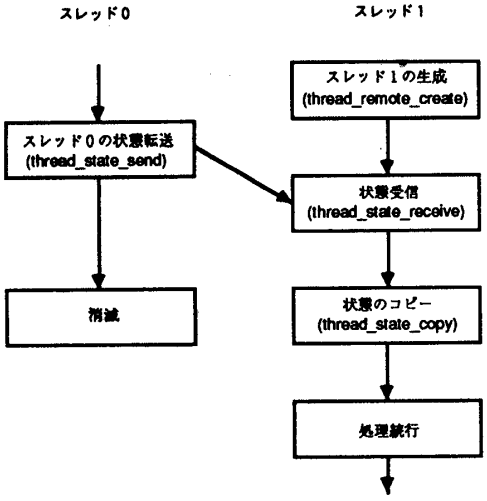

図2　スレッド移動

4．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，密・粗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ethernetを介して接続したシステムで実装を行った．

例題として，疑似３次元表示による顔画像の生成において，ポリゴンのシェーディング部分に上記手法を適用した[5]．この処理において，顔画像を構成するポリゴンはスプライン曲面として生成されるが，１個１個のポリゴンは他のポリゴンとは独立にシェーディングされるため，分散並列処理に適している．また，処理中に，最初に与えられた顔のポリゴンデータには変更が起こらないため，前述の均一なタスク環境を実現することができる．

今回の実装では，インプレメンテーションを容易にするために，ネットワークを介した並列処理部分は明示的にプログラムで指定している．2台のNeXTコンピュータにそれぞれ適当な負荷をかけてスレッドの移動を観測したところ，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．

5．おわりに

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，それを用いて分散並列な画像生成をNeXT上で行った．

今後の課題としては，以下の点が挙げられる．

（１）本稿では，システム全体の負荷情報はあらかじめ分かっているものと仮定した．実際には，リアルタイムで正確な負荷情報を軽い負荷で得ることができるような機構が必要である．

（２）今回はNeXTコンピュータ2台でのみ実験を行ったが，今後はマルチプロセッサ対応のNeXTコンピュータを多数接続したシステムで定量的な実験を試みる予定である．

参考文献

[1] McCormick, B. H., DeFanti, M. D. and Brown, M. D. (ed.), *Visualization in Scientific Computing*, Compu. Graph., Vol. 21, No. 6, 1987.

[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 news system, UNIX Review, August, 1986.

[5] Thalmann, N.M. and Thalmann, D., "Image Synthesis," Springer-Verlag, 1987.