3F－7

分散仮想記憶に基づくオペレーティングシステム DM-1 における タスク・スレッドモデル

藤川 賢治 †　篠原 拓嗣 †　大久保 英嗣 ††　津田 孝夫 †
† 京都大学工学部情報工学科　†† 立命館大学理工学部情報工学科

1 はじめに

我々は，数台から十数台程度の同一アーキテクチャの計算機を LAN によって接続した環境に，分散仮想記憶に基づくオペレーティングシステム DM-1 を設計・開発している．分散仮想記憶とは，システム上に唯一存在する仮想空間であり，ネットワーク上の全ての主記憶・2次記憶にマッピングされる．分散仮想記憶によりメモリ資源の位置透過性などが達成される．

DM-1 では資源管理を行なう単位としてタスク，実行の主体としてスレッドを定義している．タスクはネットワーク透過に存在し，ユーザはタスク中で起動されるスレッドがどのノードで実行されるかを意識する必要がない．このためユーザは，ネットワーク上のメモリ・プロセッサを効果的に使用するプログラムを容易に記述することができる．

本稿では，2章において分散仮想記憶の概念を述べ，3章において DM-1 におけるタスク・スレッドモデルについて述べる．

2 分散仮想記憶

分散仮想記憶は，ネットワーク上で唯一の仮想アドレスをノードの主記憶・2次記憶にマッピングする．仮想アドレスから実アドレスへのマッピングはプロセッサのページング機構等を利用することにより実現し，複数のノードに分散されるページの一貫性制御は write-invalidate 方式により行なう．

ユーザは仮想アドレス空間をメモリオブジェクトという単位で利用する．メモリオブジェクトは MULTICS[1] におけるセグメントに類似の概念であり，ファイルやデータ領域などに相当する．メモリオブジェクトはメモリオブジェクト ID により区別され，この ID は DM-1 システム中でメモリオブジェクトに対して一意に割り当てられる．さらにメモリオブジェクト内のアドレスの指定は，メモリオブジェクト内オフセットにより行なう．このように分散仮想記憶上の仮想アドレスの指定は，メモリオブジェクト ID とメモリオブジェクト内オフセットとの組により行なう．

3 タスク・スレッドモデル

Mach[2] などでは，プロセスの概念を，資源管理を行なう単位であるタスクと処理の流れであるスレッドに切り分けており，オーバヘッドの少ない並列処理を実現している．分散環境における並列 OS において，このようなタスク・スレッドモデルを採用しているものは多いが，通常 タスクはノードに固定されてしまっている．そのためタスクは，特定ノードのプロセッサ資源のみを利用することしかできない．

DM-1 におけるタスク・スレッドモデルでは，スレッドが実際に割り当てられるプロセッサを特定のノードに固定しておらず，タスクが複数のノードのプロセッサを利用することが可能となっている．ユーザはスレッドが実際にどのノードで実行されているかを意識する必要がない．このように，DM-1 におけるタスクは，複数のノードのプロセッサ・主記憶・2次記憶を使用することが可能であり，ネットワーク透過な存在であるといえる．DM-1 システムにおけるタスクとスレッドの構成を示す概念図を図 1 に示す．

3.1 タスクの構成

タスクは資源管理の単位であり，他のタスクからの不正なメモリへのアクセスが禁止される．タスクにメモリ資源を割り当てる場合，メモリオブジェクトを基本単位として行なう．一般的なタスクは，1つ以上のコード

Task Theread Model of the Operating System DM-1 Based on Distributed Virtual Memory
Kenji Fujikawa†, Yasuo Okabe†, Eiji Okubo††, Takao Tsuda†
† Department of Information Science, Kyoto University
Kyoto, 606-01, Japan
†† Department of Computer Science and Systems Engineering, Ritsumeikan University
Toujiin Kitamachi, Kita-ku, Kyoto, 603, Japan

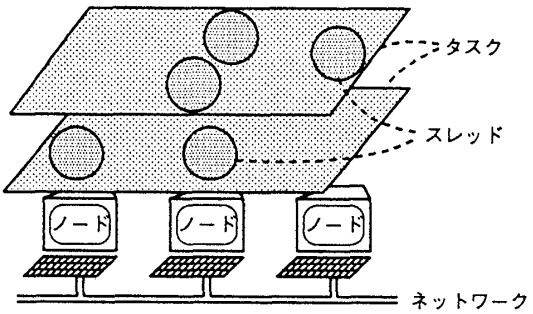

図 1: DM-1 システム上のタスクとスレッド

オブジェクト, 1 つ以上のデータオブジェクト, スレッドの個数分のスタックオブジェクトから構成される.

コードオブジェクトは, タスク間で共有されることを前提に記述されている. タスク間でコードを共有することにより, メモリ資源の有効利用が行なえる.

タスクは, アクセス可能なメモリオブジェクトをメモリオブジェクト ID によって識別する. DM-1 システムは, タスクごとに利用可能なメモリオブジェクト ID のリストを持ち, そのリストを基にハードウェアのメモリ管理機構を利用してタスク間のメモリ資源保護を行なう. タスクが必要とするメモリオブジェクトの主記憶への読み込みは, 要求時ページングにより 2 次記憶から主記憶に, ページ単位に区切られて行なわれる.

3.2 スレッドの構成

DM-1 により提供されるスレッドは, 仮想プロセッサと見ることができる. スレッドは fork システムコールを行なうことにより生成され, 必ず一つのタスクに所属し, スレッドが終了するまでそのタスク内に固定される.

DM-1 では, あるタスク中のスレッドを, ネットワークで接続されたどのノードのプロセッサにも割り当てることが可能である. これにより, システムによる動的な負荷分散を, またスレッド間の通信が頻繁に行なわれている場合にはスレッドを 1 つのノードに集めることによる通信量の抑制を行なうことができる.

3.3 タスク間手続き呼び出し

DM-1 ではタスク間通信方式としてタスク間手続き呼び出し (ITPC) と呼ぶ方式を採用している. ITPC は, ユーザインタフェースにおいては RPC と同様であるが, ライブラリで実現されているのではなく, マシンコードのレベルにおいても通常の手続き呼び出しと同じである. このため通信のためのオーバヘッドが少なくなり, またユーザは通信を行なうプログラムを容易に記述することができる.

具体的には, 他タスクの手続き呼び出しは, その手続きの開始アドレスを指定することにより実現される. 手続き呼び出しを行なったタスクはメモリ保護違反を起こすことになり, システムがメモリ保護違反を検出し, タスク間の通信を実現する.

3.4 プログラミング環境

DM-1 システムは, C 言語によるプログラミング環境を提供する. タスク・スレッドの操作に関するシステムコールは, システムタスクに対する ITPC により実現されている. タスク間通信としては, 共有メモリとしてのメモリオブジェクトを利用したものと, ITPC とが利用可能である. また UNIX システムコールとの互換性は, 互換ライブラリを提供することによって行なう.

DM-1 におけるタスクはネットワーク透過な存在であるため, ユーザは全くノードを意識することなく, ネットワーク上のメモリ資源・プロセッサ資源を有効に利用するプログラムを記述することができる. このように DM-1 は, 分散処理を容易に, かつ効果的に行なうプログラミング環境を提供する.

4 おわりに

本稿では, 我々が現在 研究・開発している分散オペレーティングシステム DM-1 のタスク・スレッドモデルに関して述べた. 現在 DM-1 の基本的な実装は i386 上で実現されており, さらに i486 への移植が進められている.

今後 システムの考察・実装を進めつつ, 応用例に関する考察を行なう予定である.

参考文献

[1] Elliott I. Organick, "マルティクスシステム: システムのアーキテクチャとソフトウエア", 共立出版 (1972).

[2] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, M. Young, "A New Kernel Foundation for UNIX Development", Proceedings of the Summer 1986 USENIX Technical Conference (1986).