分散オペレーティングシステムDM-2における スレッドディストリビュータの実現

7L－1

伊藤ちひろ † 篠原拓嗣 † 藤川賢治 † 大久保英嗣 †† 津田孝夫 †
† 京都大学工学部情報工学教室 †† 立命館大学理工学部情報学科

1 はじめに

ネットワークを介して計算機を接続した分散環境の利点として、一つの仕事を複数の計算機で処理させることで応答時間の短縮が期待できることが挙げられる。しかし、これらの計算機間でネットワークを介した通信が頻繁に起こると、却って効率が悪化する。分散環境の効率よい利用のためには、負荷の分散のみならず通信量を抑制することが重要な課題である。

しかし、前もって通信量を予測し低通信コストの並列アプリケーションを作成することは一般に困難とされる。そこで、オペレーティングシステムレベルで動的に負荷分散・通信量の抑制を行うことが考えられる。

2 DM-2の概要

現在、我々は DM-2 [1] と呼ぶ分散OSを開発している。DM-2が対象とするのは、数台から十数台の計算機がネットワークで接続されたシステムである。それぞれの計算機をサイトと呼ぶ。現在は Intel i486 プロセッサを搭載した計算機をイーサネットで接続した環境に実装されている。

DM-2の採用している分散仮想記憶は、システム上の全サイトの二次記憶を単一の仮想アドレスによって統合している。各サイトの主記憶はキャッシュとして機能する。分散仮想記憶は、ハードウェアのページング機構を用いてページ単位で管理される。同一ページのコピーを複数のサイトが持つことを許しており、ページの整合性は現在 write-invalidate 方式にて制御している。

分散仮想記憶に基づき、ネットワーク透過なタスク・スレッドモデルが実現されている。ここでいうタスクは、プログラムを実行する際に資源の割り当てが行われる単位で、スレッドはプログラムの実行の流れである。一つのタスクに属するスレッドが複数生成されて、これに同時にプロセッサが与えられればプログラムは並列実行されることになる。

DM-2は、タスク・スレッドのネットワーク透過性を利用して動的な負荷分散及び通信量抑制を行う機構を有す。これをスレッドディストリビュータと呼ぶ。

The Implementation of Thread Distributor in DM-2
ITO Chihiro †, SHINOHARA Takuji †, FUJIKAWA Kenji †, OKUBO Eiji †† and TSUDA Takao †
† Dept.of Information Science, Kyoto University
†† Dept.of Computer Science, Ritsumeikan University

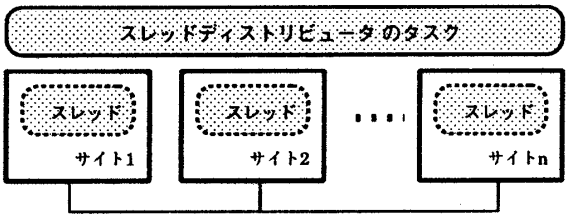

図 1: スレッドディストリビュータのスレッド配置

3 スレッドディストリビュータ

スレッドディストリビュータは、スレッドを各サイトへ定期的に再配置することでサイト間通信を抑制しつつ動的負荷分散を行う。各サイトの負荷やサイト間における通信量などの情報から、他サイトに移送すべきスレッドや移送先サイトを決定する。

スレッドディストリビュータはシステム立ち上げ時に、一つのタスクとして起動される。このタスクに属するスレッドはサイトの数だけ生成され、各サイトに一つずつ配置される (図1)。以降これらのスレッドは、それぞれ独立に自サイトに関する処理を行う。

スレッドディストリビュータが移送の対象とするのは、スレッドディストリビュータ以外のタスクに属するスレッドである。DM-2では、実際のスレッド移送のプリミティブはOSの機能として提供されている。スレッドディストリビュータが行う処理は、スレッド再配置の方針を決定しそれに従ってOSに移送を依頼することである。

3.1 構成

スレッドディストリビュータは、メイン部、情報取得部、負荷分散部、通信量抑制部の四つのモジュールにて構成される。各モジュール間の関係を、図2で示す。

3.1.1 メイン部

メイン部は、システム立ち上げ時やサイト追加時における各種の処理を行う。また、一定時間が経過する毎に情報取得部を呼びシステム情報をスレッドディストリビュータ内部に読み込む。そして負荷分散部や通信量抑制部を呼ぶ。処理が終了すれば、メイン部に制御が戻り待ち状態となる。スレッドの終了により各サイトの負荷に変動が起きた場合も、負荷分散のために同様な処理を行う。

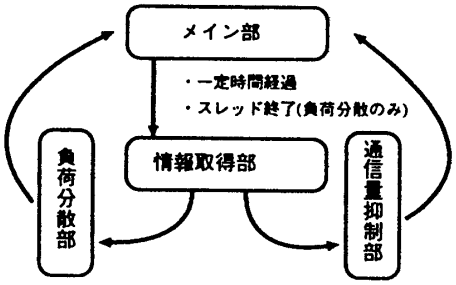

図 2: モジュール間の関係

3.1.2 情報取得部

情報取得部は、スレッド配置のために必要なシステム内の情報を得る。種々の負荷情報はOSが定期的に収集し、スレッドディストリビュータに読み込み権限を与えた記憶領域に書き込んでおく。

各サイトでは、これらシステム全体の情報を適宜読み込み、スレッドディストリビュータ内部のデータ構造体に格納する。この構造体はサイト毎に別のページに確保するので、データ格納の際に不要なページ転送を抑えている。

3.1.3 負荷分散部

負荷分散部は、負荷分散のためにスレッド再配置の計画を行う。負荷の指標には、サイトにある実行待ち状態のスレッドの数を用いる。二つの閾値によって、サイトの状態を低負荷、過負荷、どちらでもないの三通りに分類する。

負荷分散の目標は、システム内に実行待ち状態のスレッドがあるにも関わらず、実行すべきスレッドがないサイトが存在するような状態をつくらないことに置く。各サイトの負荷を均等にすることは考慮しない。なぜなら、サイト間で負荷に不均衡が生じても、全てのサイトに実行すべきスレッドが存在する限り総応答時間は変わらないからである。

負荷分散部に制御が移行すると、まず自サイトの負荷を調べて低負荷であれば以降の処理を行う。まず、ランダムに過負荷サイトの中から一つを選び、その中で移送対象とするスレッドを選択する。この際には、対象サイト内で同一タスクに属するスレッドが他に存在しないようなスレッドを選択する。該当するものが存在しなければ、CPU実行時間が最も長いスレッドを選ぶ。これは、負荷分散を行うことでサイト間の通信量を増大させないためである。同一タスクに属するスレッド間では通信が頻繁に行われている可能性があるので、一サイト内に存在するこのようなスレッドはなるべく移送対象から外すようにしている。

3.1.4 通信量抑制部

通信量抑制部は、通信量抑制のためのスレッド再配置を行う。スレッド移送を行うのは、あるタスク内のスレッドがサイト間通信を頻繁に行っていて、一つのサイトに全てのスレッドを集めた場合より応答時間が長くなると予想されるとき[2]である。

以下のような例を考える。

- あるタスクの処理量はn個のスレッドに均等に分割できる
- これらのスレッドは並列に実行可能である
- 各々のCPU実行時間はT_Eである
- システムに他のタスクがない

このとき、一サイトに全スレッドを配置して実行させた時の応答時間はnT_Eである。一方、n台のサイトに一スレッドずつ分散させた場合、各スレッドのうちサイト間通信にかかる時間の最大値をT_Cとすると、応答時間は$T_E + T_C$となる。このとき、分散することで応答時間を短縮できるT_Cの上限は$(n-1)T_E$となる。

そこで、スレッドの過去一定期間の通信時間ΔT_CとCPU実行時間ΔT_Eの比$\Delta T_C/\Delta T_E$を通信実行比とよび、スレッドの通信量が適度なものかどうかを判断する指標として用いる。

あるサイトにて通信量抑制部に制御が移ると、そのサイト内の全スレッドについて通信実行比を計算する。これが$n-1$(nはタスク内のスレッドの数)を超えていると、そのスレッドを移送する。移送先は、そのスレッドが過去一定期間に行った通信の回数が最も多いサイトとする。

ここで、負荷分散のために他サイトに移送されたスレッドが、通信量抑制の処理で元のサイトに戻ってくるということを繰り返す事態が考えられる。この影響を軽減するために、通信量抑制のために移送されたスレッドは、負荷分散のために移送することを一定時間禁止することにする。

4 おわりに

本稿で述べた、スレッドディストリビュータが用いるスレッド移送のアルゴリズムは、シミュレーションにてその有効性が示されている[2]。今後の課題は実際にDM-2システム上での評価を行うことである。ベンチマークアプリケーションを種々の条件にて実行し、応答時間を計測することによってスレッドディストリビュータの有効性を検証する予定である。

参考文献

[1] 篠原拓嗣: 分散仮想記憶に基づくオペレーティングシステムDM-2の設計と実現, 京都大学大学院工学研究科修士論文 (1995).

[2] 大西祐二: 分散オペレーティングシステムDM-1におけるスレッド分配機構, 情報処理学会第48回 (平成6年前期) 全国大会.