2D－1

動的ネットワーク負荷分散における相互優先順位法の効用

野中秀俊　伊達惇　佐藤展章
北海道大学工学部情報工学科

1　はじめに

近年コンピュータネットワークを利用した分散処理が一般的になりつつある。特に従来の集中処理に対して分散処理には、1) 負荷分散、機能分散による処理効率の向上、2) 複数の自立したコンピュータによる信頼性、3) 機種によらない拡張性・柔軟性などの利点がある。

一方分散処理において重要な概念に、ネットワーク透過性 (network transparency) がある。この透過性を高めることにより、ユーザの利用環境をより単純化することができる。上で述べた分散処理の利点のうち、負過分散による処理効率の向上を、高いネットワーク透過性のもとで実現するためには、自律的な負荷分散 (autonomous load sharing) の手法が必要であるが、現状では効果的な手法が提案されるには至っていない。

本報告では、既存のあらゆるネットワークシステムにおいて自律負荷分散の実現を可能とする方式として、相互優先順位プロトコル (mutual priority protocol) を提案し、シミュレーションによる処理効率の評価について述べる。特にこの方式は、タスクのプロセッサへの割り当ての決定に、相互優先順位アルゴリズム [1] を採用しており、またシステム全体としてだけでなく、個々のタスクに対して処理効率を向上させることを目的としている。

2　相互優先順位プロトコル

本論文では、プロセッサおよびタスクをそれぞれ $P_i(i = 1,\cdots,n)$, $T_i^k(k = 1,2,\cdots)$ で表す。また P_j において T_i^k, を実行する際の推定実行時間を $E_{ij}(T_i^k)(i,j = 1,\cdots,n;k = 1,2,\cdots)$ 、P_i から P_j への推定転送時間を $c_{ij}(i,j = 1,\cdots,n)$ と表す。

T-P 順位とは、タスクからプロセッサへの優先順位であり、$w_{ij}(c_{ij} + E_{ij}(T_i^k))(j = 1,\cdots,n)$ の値により順位づけられるものとする。また P-T 順位とは、プロセッサからタスクへの優先順位であり、到着時刻によって順位づけられるものとする。T-P 順位と P-T 順位は、相互優先順位アルゴリズムにおける二種類の優先順位に対応している [1]。

以下に相互優先順位プロトコルの概要を示す。システムは n 個のプロセッサから成り、それぞれに臨界負荷 $C_i(i = 1,\cdots,n)$ が決められているものとする。

1. **タスクの生成**：プロセッサ P_i においてタスク $T_i^k(k = 1,\cdots)$ が発生したものとする。

   プロセッサ P_i は、T_i^k に関して $P_j(j = 1,\cdots,n)$ を $w_{ij}(c_{ij} + E_{ij}(T_i^k))$ の値に応じて順序づける。この順位を タスク T_k の T-P 順位と呼ぶ。

2. **タスク割当の依頼**：P_i は、T-P 順位に従って他のプロセッサに対して依頼メッセージを送る。

   なお T-P 順位が最上位の場合 T-P 順位最上位フラグ (Flag of the leader)$F_j(T_i^k)$ は 1 、それ以外の場合は 0 とする。

3. **依頼の受信**：P_j は P_i からの T_i^k の依頼メッセージを受信し、タスク $T_i^k(k = 1,2,\cdots,i = 1,\cdots,n)$ を到着順に従って順序付ける（ $T^\kappa,\kappa = 1,2,\cdots$ ）この順位を P_j の P-T 順位と呼ぶ。

4. **タスクの棄却**：受理基準 (acceptance criterion) を以下のように定義する

$$A_j(T^\kappa) = C_j(E_{ij}(T^\kappa)/(E_{jj}(T^\kappa)) \qquad (1)$$

   ここで、以下の不等式が成り立つ時、P_j は T^κ を棄却し、棄却メッセージを P_i に送る。

$$\sum_{1\le\lambda\le\kappa,F_j(T^\lambda)=1} E_{jj}(T^\lambda) > A_j(T^\kappa) \qquad (2)$$

Mutual Priority Protocol for Dynamic Autonomous Load Sharing
Hidetoshi Nonaka, Tsutomu Da-te, and Nobuaki Sato
Faculty of Engineering, Hokkaido University
Nishi 8, Kita 13, Kita-ku, Sapporo 060, Japan

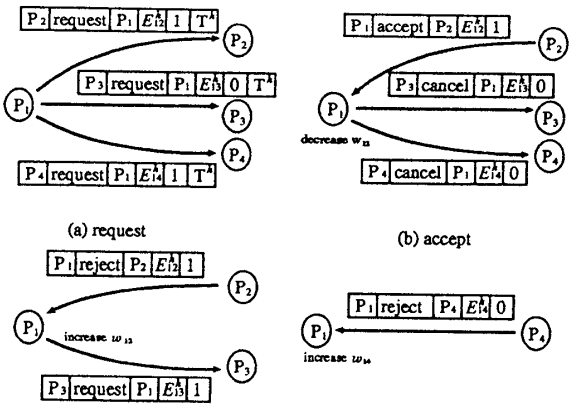


図 1: 依頼・受理・棄却の手順

5. **タスクの受理**：T-P 順位最上位フラグが 1 であり、かつ

$$\sum_{\lambda=1}^{\kappa} E_{jj}(\mathrm{T}^{\lambda}) < A_j(\mathrm{T}^{\kappa}) \qquad (3)$$

が成立するとき P_j は T^{κ} を受理し実行する。また受理メッセージを T^{κ} の発生元である P_i に送信する。

6. **タスクの実行**：タスク T^{κ} が実行されている間、推定実行時間 $E_{jj}(\mathrm{T}^{\kappa})$ を減少させ, タスクが終了したとき、$E_{jj}(\mathrm{T}^{\kappa}) = 0$ とする。

7. **棄却メッセージの受信**：T_i^k の棄却メッセージを P_j から受信すると、P_i は T-P 順位から P_j を削除し、もしそれが T-P order において最上位の場合は、T-P order における次のプロセッサを最上位とし、T-P 順位最上位フラグを変更するための依頼メッセージを送信する。さらに補正係数 w_{ij} を一時的に増加させる。

8. **受理メッセージの受信**：T_i^k の受理メッセージを P_j から受信すると、P_j は取消メッセージを T-P 順位に含まれる P_j 以外の全てのプロセッサに送信し、補正係数 w_{ij} を一時的に減少させる。

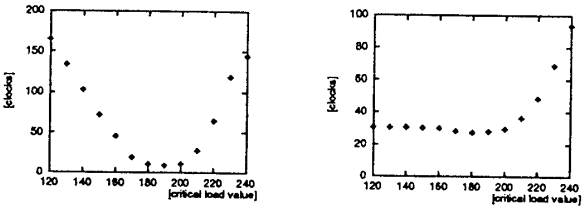
3 シミュレーションによる評価

本報告で提案した相互優先順位プロトコルによる処理効率をシミュレーションにより評価した。負荷および時間の単位として "[s]" を用い、負荷 t[s] と処理性能 $\xi(t)$[s] との間に以下の関係を仮定した。

$$\xi(t) = \xi_0 \frac{1}{1 + e^{(t-t_c)/C}} \qquad (4)$$

また $\xi_0 = 2$[s], $t_c 200$[s], $C = 10$[s] とし、タスクの発生はポアソン分布に従うものとした。

図 2 にシミュレーション結果の一例を示す。相互優先順位プロトコルにおいて、臨界負荷を変化させた時の平均待ち時間および平均実行時間をプロットしている。臨界負荷が大きい場合は、処理能力の高いプロセッサにタスクが集中した場合に相当している。この例においては、臨界負荷として 約 190[s] が最適値となっている。実験結果より、臨界負荷を適当に設定することにより待ち時間および実行時間を最小化することが可能であることが例示された。



(a) 平均待ち時間　(b) 平均実行時間

図 2: 臨界負荷の設定に伴うタスク処理時間の変化

4 まとめ

分散処理システムにおける自律負荷分散の方式として、相互優先順位プロトコルを提案し、その定式化を行った。特にタスクのプロセッサへの割当ての決定に優先順位割当てアルゴリズムを応用した。本報告ではワークステーション上でシミュレーションシステムを構成し、実験による評価によりその有効性を検証した。

参考文献

[1] Da-te, T. and Nonaka, H., "On the Assignment Algorithm by Means of Multi-Linked Data Structure," *Bulletin of the Faculty of Eng. Hokkaido Univ.*, No.128, pp. 95-102, 1985.

[2] Nonaka, H. and Da-te, T., "Efficiency of the Algorithms for Assignment Problem with Mutual Priorities," *Proc. of the 2nd JSSS Int. Conf. on Comp. Appl.*, pp. 5-8, 1986.

[3] Nonaka, H. and Da-te, T., "Autonomous Load Sharing and Mutual Priority Protocol Using Fuzzy Numbers," *Proc. IEEE Int. Conf. on Fuzzy Systems*, June 1994.