5Q－4　密結合型マルチプロセッサ上での自律的な動的負荷分散制御

Autonomatic and Dynamic Load Balancing Scheme on Tightly Coupled MultiProcessors

中山善太郎 (Yoshitaro Nakayama)　大川善邦 (Yoshikuni Okawa)
NEC 交換移動通信事業本部*　日本大学工学部情報工学科†

1　はじめに

本稿では，プロセッサ同士が局所的に結合する MNC 並列計算機において，各プロセッサが自身の状態に合わせて負荷分散と問題処理をバランス良く行なうための自律分散アルゴリズムを提案し，実験とシミュレータによる解析を行なった．

2　MNC 並列計算機

本稿ではプロセッサ (Processor Element : PE) が 4 ポート共有メモリ (Quad Port RAM : QPR) により局所的に結合する MNC(Maximum Neighbor Connection: 最大隣接結合) 並列計算機を用いる．MNC 並列計算機は，ある PE から見て 1 つの QPR を介して通信可能なプロセッサ数が最大になる結合方式の 1 つである．したがって，自律分散制御を行なう場合，より多くの隣接 PE から情報を得ることができる有利な構造である．

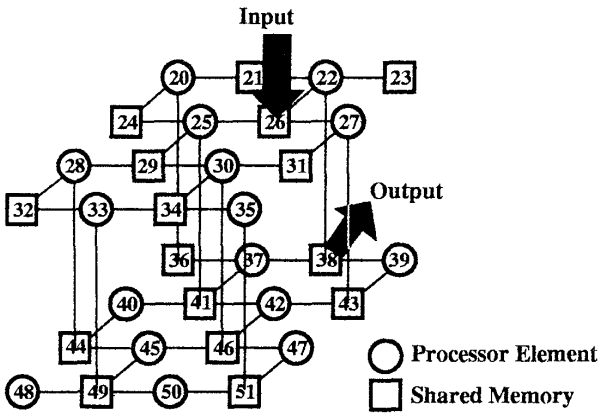

図 1: 実験に用いた MNC 並列計算機の結合

3　処理問題

データ処理時に PE 間で依存関係のない $y = f(x)$ 型の問題を扱うことにした．各 PE はデータ処理のための関数 $f()$ をあらかじめ持つ．ソースから並列計算機に入力される大量のデータ $x_i(i = 1, 2, \cdots)$(データ X と呼ぶ) は，各プロセッサで並列的に y_i(データ Y と呼ぶ) に変換され，並列計算機からシンクに出力される．x_i と y_i は 1 対 1 で対応しており，すべてのデータについて，データ X から Y への変換時間は同一であることを想定している．

4　自律分散アルゴリズム

自律分散制御では並列計算機内部の各 PE が共有メモリを介して得られた局所情報から自律的に自身の状態を判断し，処理を行なう．各 PE が行なう処理は，隣接する PE との情報交換、取得した情報に基づく行動計画、行動の実行であり，各 PE がこれらを繰り返し実行することで全体として処理が進行する．PE が処理を行なうときには，

- 制御に用いる情報
- PE が行なう行動の種類と内容
- 行動の計画方法

が重要になってくる．

4.1　参照情報

各プロセッサ PE_i が制御に用いる情報を、次の状態ベクトル $\boldsymbol{X}$ とした．

$$\boldsymbol{X} = \left[X_i,\ Y_i,\ RX^B_{ij},\ RY^B_{ij}\right] \qquad (1)$$

表 1 に、式 (1) に出てくる状態量の意味を示す．

表 1: 制御の状態量

X_i, Y_i	PE_i が保持している X・Y の量
X_{i_j}, Y_{i_j}	PE_{i_j} が保持している X・Y の量
$SX^B_{i_j}$, $SY^B_{i_j}$	PE_i から PE_{i_j} 宛の X・Y の量
$RX^B_{i_j}$, $RY^B_{i_j}$	PE_{i_j} から PE_i 宛の X・Y の量

4.2　行動の種類

PE_i が行なう行動は X を Y に変換する問題処理と負荷分散のための転送処理の 2 種類である．データには X

*Switching and Mobile Systems Operations Unit, NEC Ltd.
†Dep. of Information engineering, Faculty of Engineering, Nihon University

とYの2種類があり，MNC並列計算機ではQPRを介してPE間のデータ転送を行なうため，PEからQPRへの転送をPUT，その逆をGETとすると4つ転送処理が考えられる．

- 問題処理のための行動 (Xから Yの変換)
  - **PROC** ： $f()$ により X_i を Y_i に変換
- 負荷分散のための行動 (データ転送)
  - **XPUT, YPUT** ： X_i, Y_i をローカルメモリからQPRに転送
  - **XGET, YGET** ： RX_{i_j}, RY_{i_j} をQPRからローカルメモリに転送

効率的な処理を行うためには，これら5つの行動を組み合わせて実行しなければならない．

4.3 各行動の実行判断

XGET, YGET, YPUTについては，目標状態が一意であるため，隣接PE間の平均データ量に基づいて判断した．XPUTとPROCについては，PE_i の負荷の適正値 $\widetilde{X_i}$ よりも保持データが多いときにはXPUT，少ないときにはPROCを行なうようにした．

4.3.1 プロセッサ PE_i の負荷の適正量 $\widetilde{X_i}$

各時刻における，自PEを基準にして隣接PEの処理能力を相対的に表す状態量 PA(Processor Ability) を用いて，PE_i の負荷の適正量 $\widetilde{X_i}$ を，

$$\widetilde{X_i} = \frac{\sum_j (PA_{i_j} X_{i_j}) + X_i}{\sum_j d_{i_j} e_j + 1} \qquad (2)$$

とした．ただし，PA_{i_j} は以下の式で表される．

$$PA_{i_j}(t) = PA_{i_j}(t - t_{pa}) + \frac{G_{pa}}{t_{pa}} \int_{t-t_{pa}}^{t} \frac{\left(X_i(s) - X_{i_j}(s)\right) \left|X_i(s) - X_{i_j}(s)\right|}{\{X_i(s)\}^2} ds \qquad (3)$$

式 (3) の記号は G_{pa}:ゲイン、t_{pa}:PA 更新時間間隔、X_{i_j}:PE_i から見た PE_{i_j} の保持データ量を示す．
PA_{i_j} は PE_i を基準とした PE_{i_j} への PA である．式(3)より，PA は，将来の自PEと隣接PEの保持データ量の比を1に近づけるように作用することがわかる．

5 実験

16台のPEと16台のQPRから構成される図1のMNC並列計算機に制御アルゴリズムを実装し，実験を行なった．処理問題は様々な重さの処理を実現できるように，C言語のfor文による空ループとした．データXはfor文のループ数を含むサイズ8[byte]とし、1,000,000[個]のXをソースのQPR26から計算機に流し込んだ．データYのサイズも8[byte]とし、シンクのQPR38から回収した．
PE数16までの部分は実機を用い、それ以降はシミュレータを用いて，PE台数を1~128の間で変化させたときに，処理効率，総転送データ数を測定した結果を図2，3に示す．処理問題は100[loop/data]，1000[loop/data]の2種類を取り扱った．

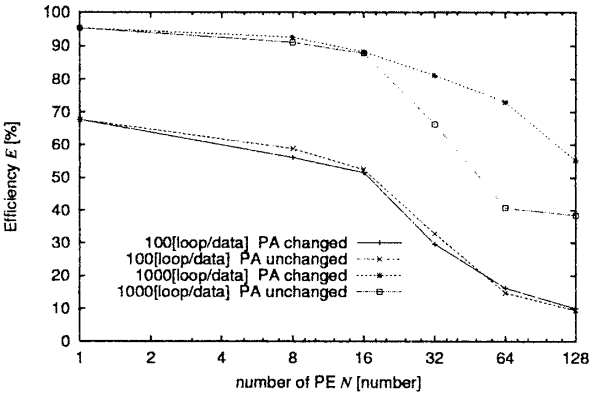

図 2: シミュレータによる処理効率の測定

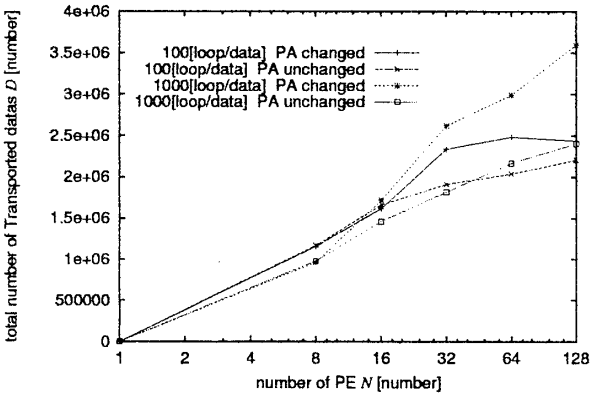

図 3: シミュレータによる総転送データ数の測定

図2，3より，負荷の軽い100[loop/data]問題に対しては，データ転送を抑制して転送オーバーヘッドを削減でき，重い1000[loop/data]処理問題に対しては，積極的なデータ転送による負荷分散を行なって，計算機の資源を有効に利用できることが確認できた．

6 おわりに

本稿ではプロセッサ同士が共有メモリにより局所的に結合するMNC並列計算機における自律分散アルゴリズムを提案した．このアルゴリズムは，ある処理問題に対して，それが必要とする計算機の規模を動的に推定して，計算機の資源を有効に利用する制御が行なえることを確認した．