マルチメディア通信と分散処理 67－6
（1994．12．2）

内部状態モデルに遺伝的操作を導入した動的負荷分散

棟朝雅晴　高井昌彰　佐藤義治
北海道大学 工学部

Abstract

本論文では、負荷状態の観測とタスク転送先の決定を遺伝的操作により学習する動的負荷分散の一手法を提案する。その前提として、分散システムにおけるノードをFIFOおよびRound-Robin待ち行列により構成されるものと仮定し、その内部状態についてモデル化を行う。

動的負荷分散では、タスクを負荷の重いノードから負荷の軽いノードへと転送することで負荷の均一化を図るが、負荷の軽いノードを発見するためには通信ネットワークを介した転送要求の送出が必要となる。この要求をランダムまたはブロードキャストにより送出した場合、無駄な要求が多数送られることが予想される。提案する手法ではマルチキャストにより特定のノード群に対し要求を送出し、その送出先のリストを遺伝的アルゴリズムの個体として学習を行う。

A genetic load balancing scheme based on an internal model of nodes in distributed systems

Masaharu Munetomo, Yoshiaki Takai, and Yoshiharu Sato
Information and Graphics Sciences, Faculty of Engineering, Hokkaido University
North 13, West 8, Kita-Ku, Sapporo 060, Japan.

Abstract

In this paper, we present a dynamic load balancing algorithm which learns a sending set of task migration requests. The algorithm is based on an internal model of nodes consisting of FIFO and Round-Robin queues.

In dynamic load balancing in general, we equalize each processor's load by migrating tasks from heavily-loaded processors to lightly-loaded ones. If we send task migration requests randomly or by a broadcast, many unnecessary messages will be sent. The proposed algorithm employs multicast messages which are sent to specified nodes. The sending set of the messages is coded into a genetic string to which genetic operations are applied.

1 はじめに

複数のノードとネットワークにより構成される分散システムを効率的に利用する上で、各ノードの負荷を一様化することが重要である。負荷分散アルゴリズムはシステムに入力されるタスクを実行前に割り当てたり、実行中にノード間で転送することで各ノードの負荷を平均化する。負荷分散アルゴリズムは大きく、静的負荷分散、動的負荷分散の2つに分類することができる。

静的負荷分散では、システムの内部状態およびタスクに関する情報を用いて、タスクの割り当てを実行前に行う。この場合、実行中の負荷変動に追従することはできない。これに対して動的負荷分散では、システムの動作中にタスクを負荷の軽いノードから負荷の重いノードへ転送することで、間接的に負荷の平均化を図る。これにより、システムの負荷変動に追従するのみならず、実行されるタスクに関して何ら情報が得られない場合においても負荷分散を行うことができる。

動的負荷分散アルゴリズムにおいて問題となるのは、(1) 負荷情報をどのようにして集め、(2) どのような基準でタスクを転送するか、ということである。負荷情報の観測とタスク転送の決定をどのように行うかによって、集中制御型と分散制御型に分類することができる。

集中制御型の動的負荷分散では、一つのノードで集中して負荷情報の管理とタスク転送の決定を行う。この方法はタスク転送の決定が比較的正確に行われるが、そのためには負荷情報の収集に多くの通信量を必要とする。

これとは対照的に、分散制御型の動的負荷分散では負荷情報の管理とタスク転送の決定は、それぞれのノードにおいて行われ、ネットワーク全体を管理するノードは存在しない。この手法の利点としては、負荷情報の観測が局所的であるために、それに要する通信量が少ないという点があげられるが、全体を管理していないために、非効率的なタスクの転送が行われる可能性がある。

分散制御型の動的負荷分散アルゴリズムの中で最も基本的なものの一つである sender-initiated algorithm[1, 2] は、タスク転送の送り手、すなわち負荷の重いノードからタスク転送要求を送出する手法である。このアルゴリズムでは、ランダムに選ばれたノードに対してタスク転送の要求を送出し、要求先のノードの負荷が軽い場合には、そこに対してタスクを転送する。そうでない場合には、負荷の軽いノードが見出されるか、ある一定回数の限度を越えるまで、繰り返し転送要求を行う。このアルゴリズムではネットワーク全体の負荷が重い場合には要求が送出されてもそれが受理される可能性が少なく、無駄な負荷転送要求が度々送出され、全体の効率が低下する可能性がある。

Kremien らは、分散制御型の動的負荷分散において Domain の概念を用いることで、システムサイズが大きくなった場合でも有効なタスク転送要求を送出するアルゴリズムを提案している [3]。実際にタスク転送要求を送出する場合には、ごく少数のタスク転送先を発見すれば十分であるので、ブロードキャストによりネットワーク全体に転送要求を送出するのは非効率的であり、ある一定数の限られたノード群に対してマルチキャストにより送出するのが妥当であると考えられる。

また、タスク転送要求の送出先を遺伝的アルゴリズム [4] により学習する手法が提案されている [5]。このアルゴリズムでは送出先のノード群を $\{0,1\}$ からなるビット列として符号化し、それに対して遺伝的操作である交叉・突然変異を適用し、適切な送出先からなる集団を構成する。

本論文では、ノードの内部状態をモデル化し、それを前提とした動的負荷分散アルゴリズムを提案する。タスクの転送要求の送出先に関して遺伝的アルゴリズムを用いた学習を適用するすることで、効率的な負荷情報の収集と利用を図り、結果としてタスク転送要求に要する通信負荷を少なくする。使用する遺伝的アルゴリズムとしては、使用する記憶容量ならびに遺伝的操作に要する計算コストを少なくするために、符号化の方法および遺伝的操作に改良を加えたものを採用した。

2 内部状態モデル

ノードの内部状態モデルとして、次のものから構成されるモデルを採用する。

1. プロセッサ (局所メモリを含む)
2. 内部待ち行列 (Round-Robin)

3. 外部待ち行列 (FIFO)

4. タスク分散プロセス (Distributor)

5. 通信入出力装置 (Communication Channel)

図1にノードの待ち行列モデルを示す。

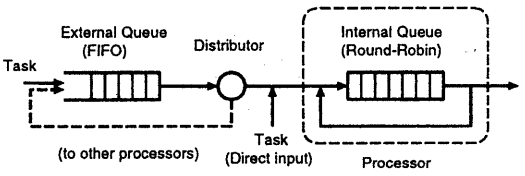

図 1: ノードの待ち行列モデル

Round-Robin による待ち行列以外に、外部に FIFO の待ち行列を付加することでプロセッサが重くなり過ぎないようにタスク入力を制御する。

粒度の小さなタスクについては、直接 Round-Robin 待ち行列に入れることを可能とする (ユーザが決定する)。負荷を調整する必要があるため、粒度の大きなタスクは一旦 FIFO 待ち行列に入り、Distributor により転送先のノードを決定する。これにより各ノードの負荷を平均化し、システムの利用効率を高める。

ノード i に関して、その内部状態を決定するパラメータとして、以下を採用する。

1. プロセッサの絶対速度 s_i

2. プロセッサの実メモリ使用率 m_i

3. 外部待ち行列の長さ L_i

4. プロセッサの負荷平均値 (load average) l_i

プロセッサの絶対速度 s_iは、そのプロセッサが許容できる負荷平均値の目安として定義される。実メモリ使用率は、スワップアウトしているものを除いた直接使用可能なメモリ空間の使用率である。外部待ち行列長は実行待ちのタスクの個数を示している。プロセッサの負荷平均値は、内部待ち行列中に存在するタスクのうち、実際に動いている数の時間平均で示される。

3 遺伝的アルゴリズムの改良

ここでは、個体集団の記憶領域を少なくする表現法である Positional Representation(以下 PR と略す) およびそれに対する遺伝的操作を提案する。

遺伝的アルゴリズム [4] では、ある文字集合 (アルファベット) に属する文字から構成される固定長の文字列として個体が表現される。一般的には $\{0,1\}$ からなるビット列が使用されることが多い。

PR ではビット列表現による長さ L の個体について、文字 1 の存在する位置 k ($k=0,1,\cdots,L-1$) を列挙した可変長のリストを個体として表現する方法である。例えば、個体 11011100 を PR で表現すると、$(0,1,3,4,5)$ となる。

PR における突然変異は新たな文字の追加、または存在する文字の消去に対応する。例えば、

$$01\underline{0}11100 \rightarrow 01\underline{1}11100 \tag{1}$$

なる突然変異を PR で示すと、

$$(1,3,4,5) \rightarrow (1,\underline{2},3,4,5) \tag{2}$$

と表現され、

$$01\underline{0}11100 \rightarrow 00\underline{0}11100 \tag{3}$$

なる突然変異は、

$$(\underline{1},3,4,5) \rightarrow (3,4,5) \tag{4}$$

となる。

つまり、突然変異を実行するためには、突然変異の発生した (2 進表現の文字列における) 位置を求め、それが PR における個体中に存在すればそれを除去し、しなければ付け加えるという操作を行うだけでよい。

また、交叉は個体ペアの間での文字の交換となる。ここでは、Uniform Crossover[6] を用いた場合について説明する。この交叉法では、それぞれの位置において確率 1/2 で個体ペア間での文字の交換を行う。以下の例では、下線部において交叉が行われ、新たな個体ペアが生成される。

$$\begin{array}{l} 10\underline{1}1\underline{1}00\underline{0} \\ 11\underline{0}0\underline{1}10\underline{0} \end{array} \rightarrow \begin{array}{l} 11\underline{1}0\underline{1}00\underline{0} \\ 10\underline{0}1\underline{1}10\underline{0} \end{array} \tag{5}$$

交叉を行う場合、ある位置において個体ペアの文字が一致する場合には文字を交換する必要がない。交換の必要があるのは、個体ペアの間で文字が違う場合である。式5で示した交叉をPRにより表現すると、以下の通りとなる。

$$\begin{matrix}(0,2,\underline{3},4)\\(0,\underline{1},4,5)\end{matrix} \rightarrow \begin{matrix}(0,\underline{1},2,4)\\(0,\underline{3},4,5)\end{matrix} \qquad (6)$$

この場合、交換の対象となった位置の中で、個体ペア間で内容の異なっている下線を引いた文字に関してのみ交換を行うと良い。つまり、PRにより表現された個体ペアに関して交叉を行う場合には、片方にしか含まれていない文字のみをチェックし、それを確率1/2で交換すると良い。

以上により、比較的個体が長く1の数が少ない場合には、PRを用いることで、記憶領域が少なくて済むだけでなく、交叉、突然変異を行う上でも効率的である。

4 動的負荷分散アルゴリズム

4.1 概要

本論文で提案するアルゴリズムは、FIFO待ち行列からRound-Robin待ち行列へタスクを投入する際にDistributorを通して、負荷が重い場合に他のノードへそのタスクを転送することで、各ノードの負荷の平均化を図る手法である。提案手法の概要を図2に示す。

それぞれのノードは以下に示すデータを保有している。

- 負荷分散の対象となるノードのリスト N
- 送出先のリストにより構成される個体集団 $P = \{S_1, S_2, \cdots, S_m\}$

それぞれのノードでは、ネットワーク全体のノードリスト Nを持っており、それを用いて送出リストに関する初期集団をランダムに生成する。ノードの増設または除去された場合にはNを更新する。

個体集団に含まれる各個体S_iは、ノードの部分集合$n_i \subset N$(ここで$|n_i| << |N|$とする)と適合度値f_iにより、$S_i =< n_i, f_i >$として定義される。各個体の表

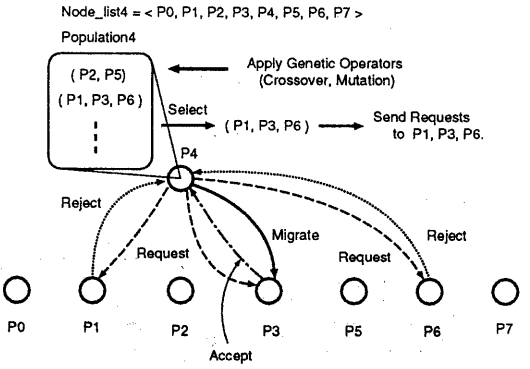

図 2: 提案手法の概要

現形式としてはPRを使用し、ノードの名前をPRにおける位置番号に対応させることにより符号化する。

負荷情報の観測を一定間隔で行い、自ノードの負荷が重くなった場合にタスク転送を開始する。タスク転送を開始する基準としては、以下の指標H_iを用いる。

$$H_i = (aL_i + bl_i)/s_i. \qquad (7)$$

ここで、a, bは正の定数である。このH_iがある閾値H以上になった場合にタスク転送の手続きを開始する。逆に、$H_i < H$の時には、タスクを自ノードのRound-Robin待ち行列に投入し、処理を開始する。

タスクを転送するために、まず個体集団より個体の持つ適合度値に比例した確率で1つの個体を選択する。そして、そこに記述されているノード群に対してタスク転送要求のメッセージ(Request)を送出する。

タスク転送要求のメッセージを受けとったノードでは、負荷が軽い場合にはAcceptメッセージが、それ以外の場合にはRejectメッセージが送り返される。タスク転送要求を受け入れるかどうかを決定する指標T_iは以下で定義される。

$$T_i = (a'L_i + b'l_i)/s_i + c'/(1 - m_i). \qquad (8)$$

ここで、a', b', c'は正の定数である。このT_iが閾値Tよりも少ない場合に、タスク転送の要求を受け入れる。

少なくとも1つのAcceptメッセージを受けとった場合に転送要求は成功し、タスクがAcceptメッセー

ジの送り元のノードへと転送される。転送されたタスクはノードの FIFO 待ち行列に投入される。本アルゴリズムでは、実行途中のタスクは転送の対象としない。

負荷の軽いノードが複数発見された場合には、待ち行列長 L_iが短くなって $H_i < H$となるまで、または転送先となるノードが無くなるまで、タスクを連続して転送する。

一方、受けとった返事が全て Reject である場合には、負荷の軽いノードを発見することができなかったので、タスクの転送は一切行われない。このとき、次のタスク転送を待つか、自ノードにより処理されるのを待つ。

タスク転送要求の手順の終了後に、その成否の結果を用いて各個体の適合度値の更新を行う。さらに、負荷の軽いノードが発見されなかった場合にのみ、一定確率で遺伝的操作である交叉、突然変異が集団内の個体に対して適用される。適合度評価、遺伝的操作の詳細については、以下で説明する。

4.2 適合度評価および遺伝的操作

各個体に適合度値が割り当てられ、タスク転送要求を送出する際、その値に比例した確率で個体が選択される。適合度値は過去一定回数のタスク転送要求メッセージの送出数および受け入れられた要求の数 (Accept メッセージを受けとった数) を用いて評価される。ある個体について、k回目のタスク転送終了後の適合度値 $f(k)$ は、以下の式により求められる。

$$f(k) \leftarrow rf(k-1) + e\frac{m(k)}{n(k)t(k)} \qquad (9)$$

ここで、rは $0 < r < 1$ なる定数、$m(k)$ は受けとった Accept メッセージの数、$n(k)$ は送出した Request メッセージの数、$t(k)$ はタスク転送要求に要した時間、e は $t(k)$ を正規化するための比例定数である。この式により、近い過去の影響を大きく取り、遠い過去の影響を少なく取るような評価基準を実現することができる。

遺伝的操作が起動された場合、一定数の個体または個体のペアが集団より選ばれ、交叉、突然変異がそれに対して適用されることで新たな個体が生成される。生成された個体は、集団内で適合度値の低い個体と置き換えられる。生成された個体の適合度値は親となる個体から継承される。交叉の場合には 2 つの親の適合度値の平均値が継承される。

4.3 処理手順

提案するアルゴリズムの具体的な処理手順を以下に示す。

1. 起動時に個体集団の初期化を行う。それぞれの個体はノードリストよりランダムに一定個数のノードを選択することで生成される。
2. 一定間隔で自ノードの負荷 H_iを観測し、その結果に従って以下の処理を実行する。
   - $H_i \geq H$である場合 (負荷が重い場合)、以下の 3~5 に示されるタスク転送処理を実行する。
   - それ以外の場合には FIFO 待ち行列のタスクを自ノードの Round-Robin 待ち行列に移す。その後、再び 1 に戻る。
3. 個体の適合度値に比例した確率により、集団内から 1 個体を選択する。
4. タスク転送要求のため、選択された個体の記述に従って、Request メッセージを送出する。
5. タスク転送要求の成否により、以下の処理を行う。
   - タスク転送要求が成功した場合、すなわち負荷の軽い ($T_i < T$を満たす) ノードが少なくとも 1 つ発見され、そこから Accept メッセージが戻ってきた場合には、そのノードへ向けてタスクが転送される。転送されたタスクは FIFO 待ち行列に投入される。Accept メッセージが複数到着した場合には $H_i < H$を満たす範囲で複数のタスクを転送する。タスク転送終了後に適合度値の更新が行われる。
   - タスク転送要求が失敗した場合、すなわち戻ってきたメッセージがすべて Reject である場合には、タスクの転送は一切行われない。その後, 適合度の更新が行われる。さらに、一定確率で遺伝的操作が集団に対し

て適用され、新たな個体が生成される。新たに生成された個体は、集団内で最も適合度の低い個体と置き換えられる.

6. 再び2に戻る。

初期個体集団の生成については、あらかじめ個体の長さの初期値を決めておき、ノードリストよりランダムにその数だけノードを選択することでそれぞれの個体を決定する。例えば、個体長の初期値を4とすると、初期個体集団はそれぞれ4つのノードを含むリストから構成される。この場合、どの個体を選択しても、4つのノードへ同時にタスク転送要求が送出される。初期個体の適合度値については、以下の式により決定される。

$$f(0) = \frac{1}{1-r} \tag{10}$$

ここで、rは式9で用いられているのと同じ定数である。

5 分散システムへの実現

現在UNIXワークステーションから構成されるLAN上に、本アルゴリズムを用いた動的負荷分散システムを構築中である。システムはNTDS[7]を基本とし、そのタスク転送要求送出法として本アルゴリズムを採用したものである。

ノードを追加・削除する場合には、各ノードの持つノードリストを更新する必要がある。ノードリストに新たにノードが登録された場合には、突然変異によりそのノードが個体集団内の要求送出リストに反映される。この場合、突然変異確率にバイアスをかけ、新たに登録されたノードが追加される確率を増やす必要がある。また、ノードリストからあるノードが削除された場合には、それを集団内の各個体に反映させ、削除されたノードを要求の送出リストから強制的に削除する。

6 おわりに

本論文では、ノードの内部状態モデルを示し、それを前提としてタスク転送要求の送出先を遺伝的操作により学習する動的負荷分散アルゴリズムを提案した。遺伝的アルゴリズムに関しては、従来のビット列による表現形式を改良し、記憶領域および遺伝的操作のための計算コストを少なくする手法を示した。今後の課題としては、タスク転送のための通信負荷の影響をモデル化することがあげられる。

参考文献

[1] D. L. Eager, E. D. Lazowska and J. Zahorjan: "Adaptive load sharing in homogeneous distributed systems", *IEEE Transactions on Software Engineering*, Vol. 12, No. 5, pp. 662–675 (1986).

[2] N. G. Shivaratri, P. Krueger and M. Singhal: "Load distributing for locally distributed systems", *IEEE COMPUTER*, Vol. 25, No. 12, pp. 33–44 (1992).

[3] O. Kremien, J. Kramer and J. Magee: "Scalable, adaptive load sharing for distributed systems", *IEEE Paralle & Distributed Technology*, Vol. 1, No. 3, pp. 62–70 (1993).

[4] D. E. Goldberg: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley (1989).

[5] M. Munetomo, Y. Takai and Y. Sato: "A genetic approach to dynamic load balancing in a distributed computing system", *Proceedings of the First International Conference on Evolutionary Computation* , pp. 418–421 (1994).

[6] G. Syswerda: "Uniform crossover in genetic algorithms", *Proceedings of the Third International Conference on Genetic Algorithms* (Ed. by J. D. Schaffer), Morgan Kaufmann Publishers, pp. 2–9 (1989).

[7] 吉田孝光, 高井昌彰, 佐藤義治: "Unix ネットワークにおけるタスク分散システムの実現と性 能評価", 情報処理学会研究報告, Vol. 94, No. 13, pp. 1–8 (1994).