2T－4

# プロセスの特徴を考慮した動的負荷分散についての一考察

佐藤 展章　野中 秀俊　伊達 惇
北海道大学工学部情報工学科

## 1 はじめに

本報告は個々のプロセスの特徴を考慮に入れた動的負荷分散についての研究である。

近年、ネットワークを介して動的に負荷を分散する方式が色々と試みられているが、非並列プログラミング言語を用いた場合の分散において効率の良い方式が確立されたとは言えないのが現状である。

一般に、あるプロセスの使用資源量 (終了までに必要な CPU 時間、主記憶量、および補助記憶量) の予測はかなり困難であることが分かっている。しかし、効率の良い分散を行うためには各々のプロセスに関して各使用資源量を予測することが大変重要である。

今回、我々は、すでに終了したプロセスの特徴と使用資源量のデータを、以後に生起する類似したプロセスの使用資源量予測の手がかりとして用いることで、効率の良い動的負荷分散を行う方法を提案し、シミュレーションにより、従来の方法との比較結果を述べる。

## 2 プロセスの特徴

負荷分散に関わる多くの論文が指摘するように、プロセスの終了までに必要な使用 CPU 時間は効率の良い分散を行う場合、重要なファクタとなる。

一般に、この使用 CPU 時間の分布は超指数分布に従うことが知られているが [1]、これより我々が知り得る分散戦略上の利点は “現在までにより多くの CPU 時間を費やしているプロセスほど、より多くの使用 CPU 時間を必要とするプロセスである可能性が高い” ことである。

これはプロセス全体の分布を考えたものであって、個々のプロセスは抽象的である。そこで、ある程度プロセスを具体化した場合を考える。

表 1: プロセスのファクタ例

| プロセス実行開始時に既知 | 同 未知 |
|---|---|
| 起動コマンド名 | 使用 CPU 時間<br>使用主記憶量<br>使用補助記憶量<br>スリープ状態の有無 |

プロセスとは、ある目的を持った作業であり、大抵の場合繰り返し実行される。同じ目的を持つ複数のプロセスについて、その使用資源量の統計をとり、それを手がかりとすることで効率の良い負荷分散を行うことが可能であると考えられる。

プロセスを特徴づける種々のファクタには多くのものがあるが、ここでは簡単のため分かり易いものを挙げた (表 1)。ファクタには、大きく分けてプロセスを実行する時点で既知のものと未知のものがある。

ここでカテゴリという用語を定義する。同じ目的を持つプロセスは同一のカテゴリに属する。また、あるプロセスをどのカテゴリに入れるかを判断する方法をカテゴライズ方法と呼ぶこととする。

基本的なカテゴライズ方法として、既知のもの (表 1 の例では起動コマンド名) をカテゴリとして、未知のもの (使用 CPU 時間等) をその属性値と考えることが挙げられる。

## 3 負荷分散方法

今回提案する負荷分散方法は、前述の考え方に従って、生起されるプロセスをカテゴリに入れ、以後、同じカテゴリに入ると考えられるプロセスが生起された場合に、以前のプロセスの使用資源量の平均、分散を考慮して負荷分散すべきか判断するものである。

例を挙げて説明すると、あるプロセスが属するカテゴリの使用 CPU 時間の分散が小さい場合、そのプロセスの使用 CPU 時間をカテゴリの使用 CPU 時間の平均とみなし積極的に負荷分散を行なう。逆に、そのプロセスが属するカテゴリの使用 CPU 時間の分散が大きい場合、負荷分散に伴うリスクが大きいことを示しているので消極的に負荷分散を行なう。リスクを少なくすることにより、効率の良い負荷分散が可能となる。

---

A Study of Dynamic Load Balancing Based on Characteristics of Processes
Nobuaki Sato, Hidetoshi Nonaka, and Tsutomu Da-te
Faculty of Engineering, Hokkaido University
Nishi 8, Kita 13, Kita-ku, Sapporo 060, Japan

# 4 評価と考察

前述の負荷分散方法に基づき、シミュレーションによる評価実験を行なった。以下に評価実験の諸元を示す。

ネットワーク形状 比較的小さな (接続プロセッサ数が高々 20 台) バス型結合ネットワークを想定している。これは、大規模ネットワークの場合の情報収集の手間を軽減するためである。

プロセッサ すべて同一の機能を有するが、処理速度は異なる。タイムスライス値が等しいラウンドロビン方式で処理する。プロセスはポアソン分布に従い到着する。

プロセスのカテゴライズ方法 今回は簡単のため“起動コマンド名”をカテゴリとして採用する。つまり、起動コマンドが 100 種類ある場合、カテゴリも 100 個ある。

カテゴリの属性値として、表 1 からは、使用 CPU 時間、使用主記憶量、使用補助記憶量、およびスリープ状態の有無が挙げられるが、今回は簡単のため使用 CPU 時間のみを使用する。主な統計量は使用 CPU 時間の平均および分散である。

プロセス 生起されるプロセスを、使用 CPU 時間に基づき大きく 3 つのグループに分ける。

1. 比較的小さな平均値を持ち、分散が小さいグループ
2. 比較的大きな平均値を持ち、分散が小さいグループ
3. 一様分布を持つグループ (分散大)

各々のグループについてコマンドを 10 種類とし、合計で 30 種類のコマンドを用いるが、第 1・2 グループに属する 20 コマンドについては、分散の大きさが少しずつ異なるようにしてある。

評価の指標としては、平均応答時間を用いる。

実験は、負荷分散をしない場合 — No Load Balancing、プロセッサ上のプロセス数 (プロセスリングキュー長) を用いて負荷分散を行う場合 — L.B. (Queue Length)、そして今回提案するプロセスの特徴を用いて負荷分散を行う場合 — L.B. (Category) について行った。

結果を図 1 に示す。横軸はプロセス到着率 [個/単位時間]、縦軸は平均応答時間 [単位時間] である。プロセス到着率が大きいほど、システム負荷が大きいことを意味している。

図 1 より、比較的システム負荷が小さい場合、負荷分散を行った方がやや有利であり、比較的負荷が大きい場合、今回提案したプロセスの特徴を用いて負荷分散行う方がかなり有利であることが分かる。

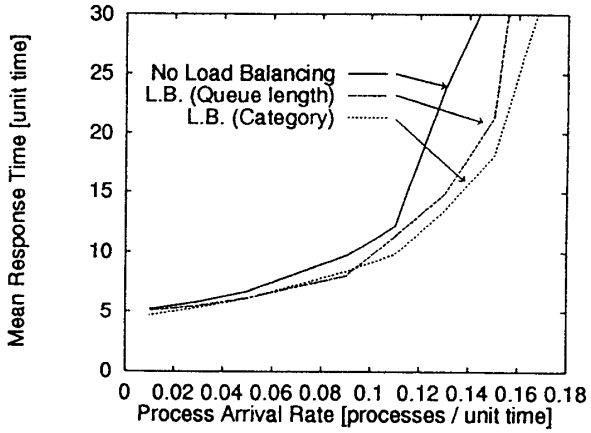


図 1: プロセス到着率と平均応答時間の関係

平均的に見て、平均応答時間は、負荷分散をしない場合に比べ、プロセッサ上のプロセス数を用いて負荷分散を行った場合、システム負荷が比較的小さいとき ( プロセス到着率 0.1 [個/単位時間] 以下 ) で 9 [%] 、システム負荷が比較的大きいとき (同 0.1 ～ 0.13 [個/単位時間]) で 27 [%] 短縮された。また、今回提案した、プロセスの特徴を用いて負荷分散を行った場合、システム負荷が比較的小さい場合で 13[%] 、システム負荷が比較的大きい場合で 35 [%] 短縮された。

負荷分散を行った場合の平均応答時間の短縮率が大きめであるのは、シミュレーションをプロセス転送にやや有利に設定した点が考えられるが、実験本来の目的である、プロセスの特徴を手がかりに負荷分散を行うことでの効率の改善は、実験結果に反映されている。

# 5 おわりに

プロセスの特徴を考慮した動的負荷分散方式を提案し、評価実験の結果、平均応答時間の短縮が確認された。今回は使用 CPU 時間のみを属性値としたが、今後は使用主記憶量や使用補助記憶量の経時的変化も負荷分散の手がかりとして用いることで、プロセスの移送にかかるコスト — 転送サイズの問題についても解決の糸口を見い出せる可能性があると考えられる。

# 参考文献

[1] Sotetsu Ri, Yusheng Ji, Jun Matsukata, and Shoichiro Asano, “負荷ベクトルを用いた負荷分散方式の検討,” 電子情報通信学会論文誌, D-I, Vol.J76-D-I, No.3, pp.118-129, 1993 年 3 月.

[2] Anna Hać, “A Distributed Algorithm for Performance Improvement Through File Replication, File Migration, and Process Migration,” *IEEE Trans. Software Eng.*, Vol.15, No.11, pp.1459-1470, Nov. 1989.