計算機アーキテクチャ　115－6
設　計　自　動　化　78－6
（1995．12．14）

履歴を考慮した動的負荷分散法

杉浦 弘幸, 市川 周一, 島田 俊夫
名古屋大学工学部
〒 464-01 名古屋市千種区不老町

概要

分散システムを効率良く使用するための方法として、動的負荷分散がある。この方法では、ユーザがプロセスを計算機に投入すると、その計算機はプロセスを処理する計算機の選択を行なう。

従来の方式では、このプロセス処理計算機の選択を負荷の大きさや処理能力に応じた確率でランダムに行なっている。しかしこの方式は必ずしも適切な計算機を選択するとは言いがたい。

本稿では、確率を計算機選択に用いる方式と違い、負荷の大きさ及び計算機のスループットの履歴から算出された優先度を用いた方式の提案とその評価を行なう。

Dynamic Load Balancing Method in Consideration of History for the Distributed Systems

Hiroyuki Sugiura, Shuichi Ichikawa, Toshio Shimada
Faculty of Engineering, Nagoya University
Furo-cho,Chikusa-ku, Nagoya, 464-01, Japan

Abstract

Dynamic load balancing is the method to use the distributed systems efficiently. In this method, the process-thrown processor selects process-executed processor when users throw process.

In conventional methods, Selecting the process-executed processor is realized with probabilistic strategy. But there is a posibility of selecting unsuitable processor.

In this paper, we propose and eveluate the new method using priority calculated by number of task and throughput of processor, which is different from using probability at selecting processor.

1 はじめに

現在、計算機の低価格化やネットワーク技術の発達などにより、複数の計算機をネットワークで接続し、資源の共有を図る分散システムが急速に普及してきている。この様な分散システムでは、動的な負荷分散を行なうことで、一部の計算機だけに負荷が集中するのを回避し、応答時間の短縮など、システム全体の性能・効率を上げることができる。

動的負荷分散は、負荷の割当方式により2つに分類することができる。単一のスケジューラにより負荷の割当を一元管理するディスパッチャ集中型と各計算機で管理するディスパッチャ分散型である。集中型は制御が簡単ではあるが、耐故障性や大規模な分散システム下での運用の困難さなどの問題により、分散型の研究の方が数多く行なわれている [1][2][3]。

ディスパッチャ分散型の方式ではユーザが負荷(プロセス)を投入すると、その計算機は処理計算機の選択を行なう。従来の方法では、負荷の大きさ・処理能力に応じた確率でランダムに計算機の選択を行なっている。これにより一つの計算機への負荷の集中を回避することができるが、負荷が小さい、あるいは処理能力の大きな計算機が利用可能な時でも、負荷の大きい計算機や処理能力の小さい計算機を選択する場合があるため、必ずしも適当な方式であるとは言いがたい。

本稿では、このことを踏まえ、履歴を考慮した優先度による負荷分散方式を提案する。この方式では従来方式と違い処理計算機の選択に確率を用いない。優先度の算出には負荷の大きさと計算機のスループットを用いる。

2 負荷分散方式

2.1 計算機の負荷状態のランク付け

計算機にプロセスが投入されると、その計算機は投入されたプロセスの処理計算機を選択する。最適な選択を行なうためには、他の計算機のより正確な負荷情報が必要となってくる。しかし、より正確な負荷情報を得るための多量の通信が逆に性能を劣化させる原因となる。

そこでこの通信量を抑える方式として負荷状態をいくつかのランク (Light, ..., Heavy) に分類する方法がある [4][5]。各計算機はこの負荷のランクが変化した時のみそれを同報通信で他の計算機に知らせることで、通信量を減少させることができる。また、他の計算機からの負荷のランクは各計算機が負荷情報として保持しておく。

2.2 重負荷回避と軽負荷選択

ランクを用いた処理計算機決定方式として、重負荷回避と軽負荷選択がある。

重負荷回避は、計算機にプロセスが投入された時、その計算機の負荷のランクが最大(Heavy)でないなら、その計算機にプロセスを割り当て(図1上)、Heavy なら最小 (Light) な計算機を探し、そこへプロセスを割り当てる (図1下) 方式である [4]。この方式では、投入先の計算機がHeavy でなければ必ずその計算機で処理されるので、プロセスを投入した計算機によって応答時間の差が大きく生じることがある。

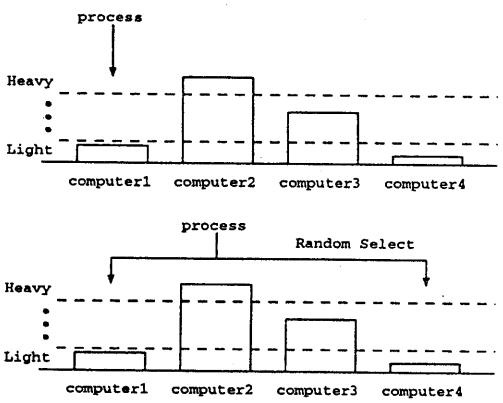

図 1: 重負荷回避

一方、軽負荷選択は、プロセスが投入された時に Light である計算機の中から処理計算機を選択する方式である (図2)。この方式はどの計算機からプロセスを投入しても重負荷回避方式ほどの応答時間の差が生じることはあまりない。

しかしプロセスの移動の頻度が多くなるので、ネットワークの転送速度に性能が左右される。辻・上野・山本・池田らはこの軽負荷選択方式にさらに重み利用方式を組み合わせた軽負荷選択重み利用方式 (WLS) を提案している [5]。この WLS は、プロセスが投入されると、統計的負荷情報より算出される確率を用いて、システム全体の Light の計算機から処理計算機を選択する。

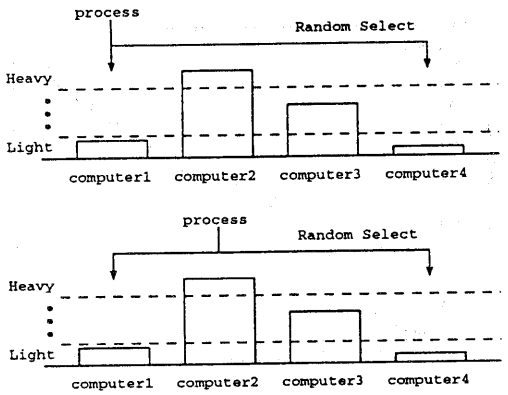

図 2: 軽負荷選択

2.3 優先度利用方式

本稿では軽負荷選択方式に優先度を加えた方式を提案する。この方式では処理計算機の選択に確率を用いない。

優先度利用方式では、計算機にプロセスが投入された場合、負荷ランクが Light の計算機のうち最も優先度の高い (大きい) 計算機を処理計算機として選択する。ただし、全計算機が Light でない時は投入計算機で処理する。

2.3.1 負荷のランクによる優先度

負荷が少ない計算機は軽負荷の状態が長く続く。したがって、軽負荷の状態が長い計算機は負荷が少ない可能性が大きい。よって、この様な計算機に対して優先度を増加させることは自然な考えである。また逆に、負荷が多い計算機は重負荷の状態が長く続く。よって、重負荷状態の長い計算機は負荷が多い可能性があり、優先度を減少させる必要がある。こうしたことで負荷の少ない計算機が処理計算機として選択されるようになる。

本稿では、時刻 t で計算機 i が保持する計算機 j の優先度 $P_{ij}(t)$ を以下の様に定義した。

$$P_{ij}(t) = P^*_{ij} + \alpha(t - \tau) \qquad (1)$$

ここで、P^*_{ij} を前回負荷のランクが変化した時の優先度、τ を前回負荷のランクが変化した時の時刻とする。

また、α は負荷のランクが小さくなるほど大きくなり、負荷のランクが大きくなると小さくなるような数である。

本稿では、この優先度 $P_{ij}(t)$ を用いた方式を PS(優先度選択方式:Priority Selection) と呼ぶ。

2.3.2 スループットを考慮に入れた優先度

処理能力の大きい計算機と小さい計算機の 2 台から処理計算機を選択する場合、ともに同程度の負荷を持っているとすれば、処理能力の大きい計算機を選択した方が良い。

負荷のランクから算出した優先度は、各負荷ランクの累積時間しか考慮されていないため、計算機の能力を十分に反映しているとは言いがたい。そこで、計算機の能力を示す尺度であるスループットを優先度に組み込むことで、計算機能力に応じた選択ができるようになる。

時刻 t で計算機 i が保持する計算機 j のスループットまで含めた優先度 $\tilde{P}_{ij}(t)$ を以下の様に定義した。

$$\tilde{P}_{ij}(t) = \tilde{P}^*_{ij} + (\alpha + \beta * T_j)(t - \tau) \qquad (2)$$

ここで、$\tilde{P}^*_{ij}$ を前回負荷のランクが変化した時の優先度、τ, α は式 (1) の時と同様である。

β は定数とし、T_j は計算機 j の過去のスループットをあらわす。本稿では T_j の値に負荷のランクの 3 状態前からのスループットの平均値を用いる。またスループットは負荷のランクが変化した時に、ランクの報告と一緒に他の計算機に同報通信で報告する。

本稿では、この優先度$\tilde{P}_{ij}(t)$を用いた方式をEPS(拡張優先度選択方式:Expanded Priority Selection)と呼ぶ。

3 性能評価

3.1 シミュレーションモデル及び条件

2.3で示した負荷分散方式に対して、シミュレーションを用いた性能評価を行なった。シミュレーションでは以下の様な仮定をした。

1. 3台の計算機をネットワークで接続し、計算機iの処理能力はμ_iとした。
2. 計算機のプロセッサはFCFS(先着順サービス)スケジューリング。
3. ネットワークはFCFS単一サーバでモデル化し、転送速度を1250(Kbyte)とした。
4. 計算機iへのプロセスの到着率は平均がλ_i(process/sec)のポアソン分布に従うものとした。
5. プロセスの処理要求時間は計算機の処理能力μ_iが1の時に平均1(sec)の指数分布に従うものとした(処理能力がμの場合平均$\frac{1}{\mu}$の指数分布に従う)。またプロセスのファイルサイズはすべて200(KByte)とした。
6. 負荷のランクを決める尺度は待ち行列にあるプロセス数とし、ランクの数はLightとHeavyの2つとし、しきい値をプロセス数2とした(2以下がLight)。
7. 優先度の算出に用いるα及びβをそれぞれ、$\alpha_{Light} = 1, \alpha_{Heavy} = -1, \beta = 0.1$とした。

シミュレーションでは、プロセスの平均到着率を変化させた場合の平均応答時間をPS、EPS、WLSの3方式で測定した。なお、シミュレーションは50回の独立試行を行ない、各試行ではシミュレーション開始60秒後から600秒後までの応答時間を集計した。

3.2 計算機能力が均質なシステムにおける評価

まず、計算機の処理能力が$\mu_i = 2(i = 1, 2, 3)$と均質な分散システムの場合を考える。平均到着率は以下の2通りとしてシミュレーションを行なった。

(a)$\lambda_1 = \lambda_2 = \lambda_3 = \lambda$とし、$\lambda$を変化

(b)$\lambda_1 = 0.2, \lambda_3 = 1.8$とし、$\lambda_2$を変化

結果をそれぞれ図3、図4に示す。

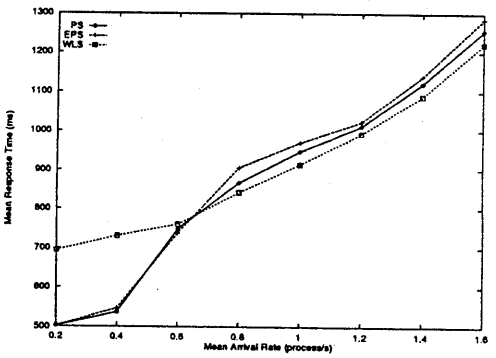

図 3: 処理能力が均質な場合 (a)

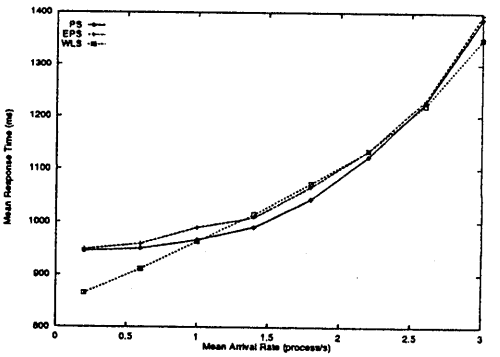

図 4: 処理能力が均質な場合 (b)

結果から、PSとEPSにあまり差が見られないことがわかる。この様な能力均質な場合では、ある計算機だけスループットが大きいといった状況が起こりにくい。したがってPSとEPSに差が生じなかったのである。

また、WLSとPS、EPSにもほとんど差は出ていない。

3.3 計算機能力が不均質なシステムにおける評価

次に、計算機の処理能力が$\mu_i = i(i = 1, 2, 3)$と不均質な分散システムのシミュレーションを行なった。平均到着率は
(a)$\lambda_1 = 0.2, \lambda_3 = 1.8$ とし、λ_2を変化
(b)$\lambda_1 = \lambda_2 = \lambda_3 = \lambda$とし、$\lambda$を変化
(b)$\lambda_1 = 1.8, \lambda_3 = 0.2$ とし、λ_2を変化
の3通りを行なった。
結果をそれぞれ図5、図6、図7に示す。

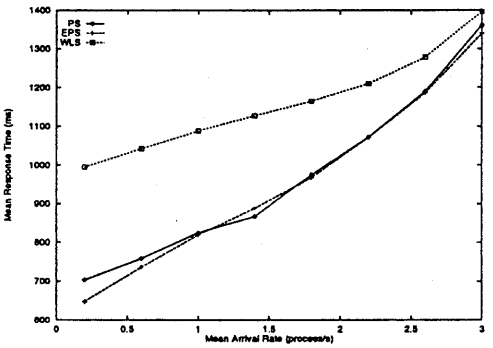

図 5: 処理能力が不均質な場合 (a)

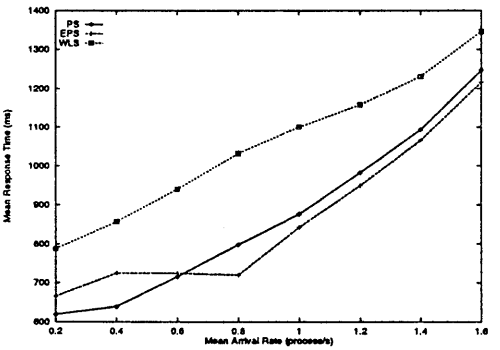

図 6: 処理能力が不均質な場合 (b)

結果よると、EPSの方がPSに比べ優れていることがわかる。ただし図6の到着率が低い場合おいてPSの方がEPSより勝っている。これはこの範囲では負荷のランクの変化があまり起こらないので、スループットの情報が十分に伝わらないためである。

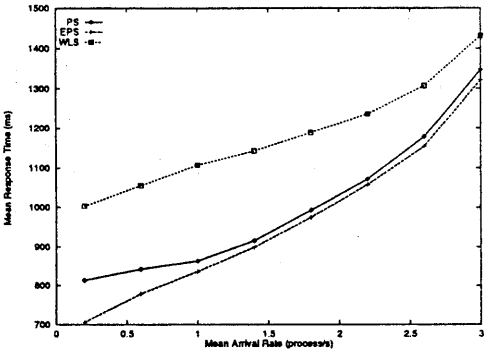

図 7: 処理能力が不均質な場合 (c)

WLSとPS、EPSの間に大きく差が出ており、提案方式が優れているのがわかる。

3.4 プロセスの到着率が変化するシステムにおける評価

最後に、$\mu_1 = \mu_2 = 1, \mu_3 = 4, \lambda_3 = 1.8$ とし、$\lambda_1 = \lambda_2 = \lambda$が時間によって変化する場合についてシミュレーションを行なった。なお、λは以下の2通りについて行なった。

$$(a)\lambda = \begin{cases} \text{variable} & 120(s) < t < 240(s) \\ 0.2 & otherwise \end{cases}$$

$$(b)\lambda = \begin{cases} \text{variable} & \begin{array}{c} 120 + 40T(s) < t, \\ t < 140 + 40T(s) \\ \text{(T=0, 1, ..., 10)} \end{array} \\ 0.2 & otherwise \end{cases}$$

結果をそれぞれ図8、図9に示す。

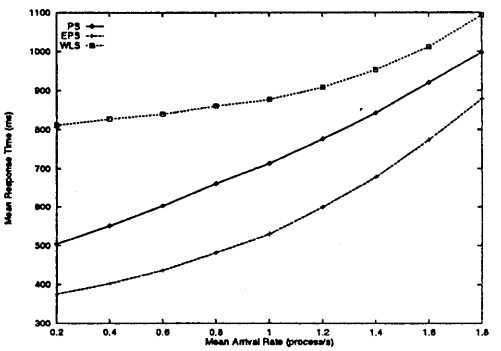

図 8: 到着率が変化する場合 (a)

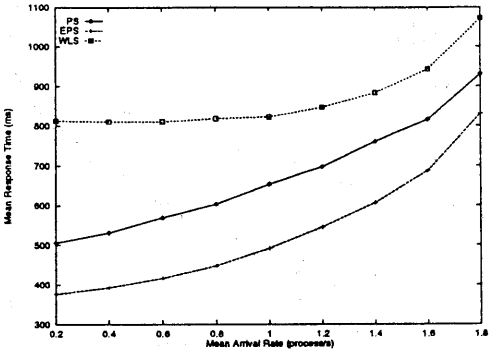

図 9: 到着率が変化する場合 (b)

結果より、WLS に比べ PS の方が、PS よりも EPS の方が平均応答時間が短くなっており、ここでも提案方式が優れていることがわかる。

4 まとめ

本稿では、分散システムに投入されたプロセスの処理計算機選択で、確率を用いず、負荷の大きさ及び計算機のスループットの履歴より算出された優先度を用いた負荷分散を提案した。

シミュレーションの結果から、提案方式は従来の確率を用いた方式より、計算機の処理能力が不均質な場合に、優れていることがわかった。

今後の検討として、計算機の処理能力が均質な場合での性能向上や、計算機台数、ネットワーク転送速度等の変化による提案方法の性能評価を考えている。

参考文献

[1] Katherine M. Baumgartner and Benjamin W. Wash: "GAMMON: A Load Balancing Strategy for Local Computer Systems with Multiaccess Networks", IEEE Trans. on Computers, Vol.38, No.8, Aug.1989.

[2] Ravi Mirchandaney, Don Towsley and John A. Stankovic: "Analysis of the Effects of Delays on Load Sharing", IEEE Trans. on Computers, Vol.38, No.11, Nov.1989.

[3] Yung-Terng Wang and Robert J. T. Morris: "Load Sharing in Distributed Systems", IEEE Trans. on Computers, Vol.C-34, No.3, Mar.1985.

[4] 鈴木伸夫, 山本幹, 岡田博美, 池田博昌: "分散システムにおけるプロセスバッファを用いた動的負荷分散方式", 信学技報, CPSY95-39, 1995.

[5] 辻敦宏, 上野仁, 山本幹, 池田博昌: "統計的負荷分散情報を用いた自律負荷分散制御", 信学技報, CPSY95-61, 1995.