仮想並列マシンにおける負荷分散に関する研究

松田亘弘† 藤井章博† 根元義章†
†東北大学大学院情報科学研究科

仮想並列マシンにおいて効率良い処理を行うには負荷分散を行うことが必要である．現在，逐次プロセスを対象とした負荷分散手法が仮想並列マシンに用いられている．しかし，逐次プロセスを対象とした負荷分散手法を仮想並列マシンに適用すると負荷の集中が起こるという問題点がある．本論文では，負荷の集中を改善する仮想並列マシン向けの負荷分散アルゴリズム，しきい値可変 Bidding アルゴリズム (VTB 法) を提案し，評価を行う．評価の結果，提案法は負荷の集中を低く抑え，かつ従来法と同等以上の平均応答時間を示すことが分かった．

1 はじめに

現在，並列処理を行う手法の一つとして，仮想並列マシンを用いる方法がある．仮想並列マシンにおいて効率良い並列処理を行うには，次の二点を考慮する必要がある．

- 各ノードの処理能力は異なる
- 各ノードはマルチユーザ，マルチタスク環境で動作しているため負荷が動的に変化する

以上の点を考慮して処理を行う手法として負荷分散がある．

現在，仮想並列マシンを実現するソフトウェアとして PVM[1]，MPI，p4 等が発表されており，その中でも PVM が広く用いられている．しかし，これらのソフトウェアはネットワークでつながれた計算機群を用いて並列処理を行う機能を提供するが，各ノードの負荷情報を得て，タスクを割り当てる，すなわち負荷分散を行う機能を有していない．

仮想並列マシンにおいて負荷分散を行うためには，各ノードの負荷情報を収集する機能を持つ外部スケジューラが必要であるが，ここで用いられている負荷分散アルゴリズムを仮想並列マシンに適用すると負荷の集中という問題 [5] が発生する．

本論文では，仮想並列マシンにおいて負荷の集中を抑える負荷分散アルゴリズム，しきい値可変 Bidding アルゴリズムを提案する．また，計算機シミュレーションにより評価を行う．

2 仮想並列マシンのモデル化

2.1 システムモデル

A Study on Load Sharing for Virtual Parallel Machine, Takehiro MATSUDA, Akihiro FUJII, Yoshiaki NEMOTO, Graduate School of Information Sciences, Tohoku University

まずシステムモデルとして仮想並列マシンのシステム構成をモデル化する．仮想並列マシンは図 1に示すように複数のノードがネットワークを介して接続されているものとする．仮想並列マシンを構成するノードの数は N台とする．また，$i\,(1 \leq i \leq N)$ 番目のノードを $node_i$ と表すことにする．また，ノードはプロセスおよびプロセスを分割したピースを処理するものとする．プロセス，ピースに関しては節で述べる．

各ノードには，仮想並列マシンを構成するための機能を提供する VM デーモン (Virtual Machine デーモン) が実装されている．ノードは通常の計算機の機能も合わせ持つ．VM デーモンは次のような機能を持つ．

負荷情報の収集 時間 T [s] ごとにシステム内の他の全てのノードの負荷情報を収集する．

並列プロセスの管理 並列プロセスの制御，分割されたタスクの割り当て管理を行い，効果的な並列処理を実施する．

データの送受信管理 並列プロセス，タスク間の通信を管理し，通信が正しく行われることを保証する．

VM デーモンはこれらの機能を提供することで複数のノード間の協調動作を行い，仮想並列マシンを実現する．

仮想並列マシンは多くの異機種からなる計算機群によって構成されている場合が一般的であると考えられる．そこで，各ノードの処理能力を示す値として P_i [I/s] を用いる．P_i [I/s] は $node_i$ が 1 秒間に P_i個の命令 (instruction) を処理できることを示す値である．

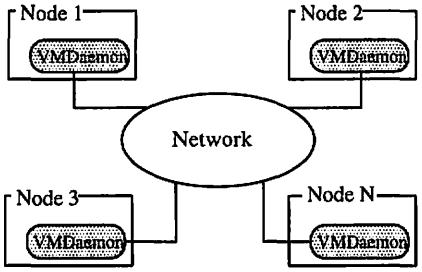

図 1: 仮想並列マシンのシステムモデル

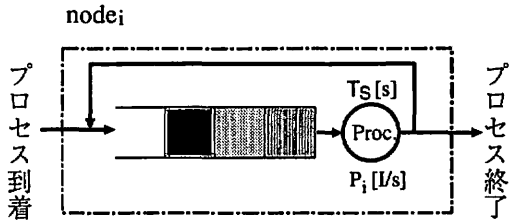

図 2: ノードでの処理モデル

現実の仮想並列マシンでは，ノード間を接続する通信機構には通信遅延時間が存在する．ここでは，通信遅延時間が仮想並列マシンにおいて実行の対象となる並列処理にかかる時間に比べ十分短いと考え無視する．

2.2 ノードでの処理モデル

ノードでのプロセス処理モデルとしてはFIFOやラウンドロビンなどがあげられるが，本研究では現実のUNIXシステムを考慮し，ラウンドロビンスケジューリングを想定した．

この状況をモデル化し，図2に示す．ノードは，単一プロセッサと単一キューで構成される．ノードにはプロセスが到着し，到着したプロセスはプロセッサで処理される．プロセスはプロセッサで処理されるまでキューで待つ．

ラウンドロビンスケジューリングにおいて，プロセスの処理を切り替える時間間隔であるタイムスライスの長さをT_s [s] とする．また，処理するプロセスの切替えに要する時間は無視できるものとする．また，タイムスライス時間内にプロセスの処理が終了したなら，直ちに次のプロセスへ切り替わるものと仮定する．

2.3 プロセスのモデル

仮想並列マシン上の各ノードで処理すべきプロセスとして，以下の2種類を仮定する．

並列プロセス VMデーモンの制御によって割り当てられるプロセス

バックグラウンドプロセス 各ノードに独自に到着し，到着したノードだけで処理されるプロセス

以下では，並列プロセス，バックグラウンドプロセスのモデル化を行う．

並列プロセスのモデル

科学技術計算等を行うプロセスは並列化が可能であるが，ここではそのプロセスをピースに分割して扱う．並列プロセスはタスクプールパラダイムを用いて実行されると考える．タスクプールパラダイムとは，並列プロセスとして扱う処理が図3に示すように予め適当な粒度のタスクに分割されてプールに蓄えられているとする概念である．これらの分割されたタスクを「ピース」と呼ぶことにする．

また，1つの並列プロセスの並列度を示す値としてピースの数を定義する．到着する並列プロセスはd_{para}個のピースから構成され，各々は独立で依存関係はない．

次に並列プロセスの処理の過程を図4に示す．プールに蓄えられているピースはVMデーモンの負荷分散アルゴリズムに従って，システム内の各ノードに割り当てられる．ピースに相互依存関係はないため，ピースが処理される時に他のピースと通信する必要はないものとする．

並列プロセスは平均λ_{fg} [個/s] のポアソン分布に従ってシステムに到着する．また，並列プロセスを構成するピースの要求処理量は平均ST_{fg} [I] の指数分布に従うとする．要求処理量とは，ピースの処理が終了するまでに必要な命令数である．

ピースの要求処理量が指数分布に従うと仮定する理由は，以下である．並列処理を行う際，プログラマは問題領域を均等になるようにd_{para}個に分割する．しかし，分割された問題領域のサイズが等しくても，その処理量も均一になるとは限らない．従って，ピースの要求処理量は指数分布に従うと仮定することにより，並列プロセスのふるまいを記述する．

バックグラウンドプロセスのモデル

ノードで処理するプロセスには，並列プロセスばかりでなく並列化が不可能なプロセスや自ノードでの

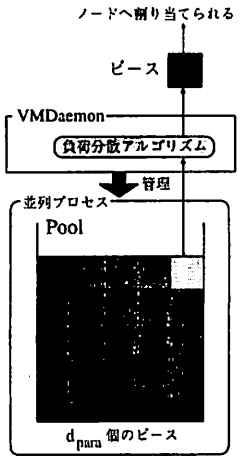

図 3: タスクプールパラダイム

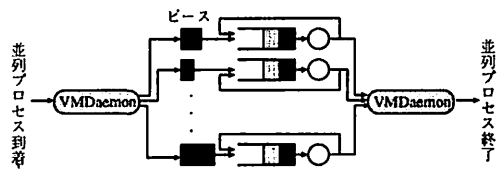

図 4: 並列プロセスのモデル

み処理すべきプロセスも存在する．これらをバックグラウンドプロセスと規定する．バックグラウンドプロセスのモデルを図5に示す．バックグラウンドプロセスは，それが到着したノードだけで処理されるプロセスである．これは，各ノードのバックグランドの負荷として表れ，仮想並列マシンの側からは各ノードの処理能力を減少させるという影響を与える．

各ノードにおいて，バックグラウンドプロセスは平均λ_{bg} [個/s]のポアソン分布に従って到着する．また，バックグラウンドプロセスの要求処理量はST_{bg} [I]の指数分布に従うものとする．

3 しきい値可変 Bidding アルゴリズム

本節ではしきい値可変 Bidding アルゴリズム (Variable Threshold Bidding Algorithm:VTB 法) を提案し，その動作の説明を述べる．その後，シミュレーションによる評価について述べる．

3.1 ノードの処理能力に基づいたしきい値の設定法

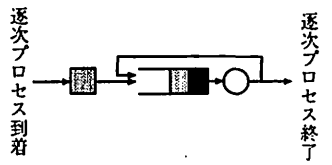

図 5: バックグラウンドプロセスのモデル

従来法では，各ノードの処理能力に比例してしきい値を設定した．そのため，従来法におけるしきい値の設定法の問題点として，各ノードにおいて独自に処理すべきプロセスの負荷を考慮していないことが挙げられる．

現実のシステムにおいて，各ノードは独自に処理すべきプロセスを持ち，その数はノードの処理能力に比例しない．そのため，しきい値の設定において，各ノードで独自に処理されるプロセス，バックグランドプロセスも考慮にいれて，しきい値を設定する必要がある．

仮想並列マシンには

- 各ノードのプロセッサの処理能力は異なる
- 各ノードが独自に持つ負荷の大きさは異なる

という特徴がある．そのため，各ノードの処理能力と独自に持つ負荷の大きさに応じてしきい値を設定することにより，効率良い負荷分散が期待できる．

次に VTB 法における各ノードのしきい値の設定法について述べる．$node_i$は図6に示すように，ピースとバックグランドプロセスが別々に独立して到着するモデルとなる．ここで，$ST_{fg} = ST_{bg} = ST$と

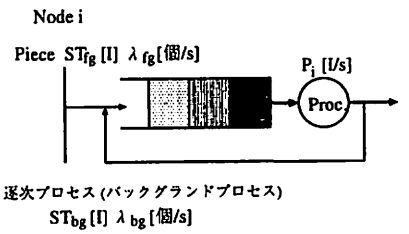

図 6: 各ノードの処理モデル

すると，$node_i$は図7に示すような$M/M/1$待ち行列システムとして近似できる．

これより，ピースの平均応答時間W_i [s]は以下と

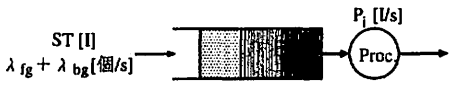

図 7: 近似された各ノードの処理モデル

なる.

近似した $M/M/1$ 待ち行列システムにおけるプロセスの平均到着率λとプロセスのサービス率μを以下のように定義する.

$$\lambda = \lambda_{fg} + \lambda_{bg} \ , \ \mu = \frac{P_i}{ST} \tag{1}$$

また, システム負荷ρと平均キュー長QL_iの関係を次に示す.

$$\rho = \frac{\lambda}{\mu} = \frac{QL_i}{1+QL_i} \tag{2}$$

式(1),(2)よりピースの平均応答時間W_iは以下となる.

$$W_i = \frac{1}{\mu - \lambda} = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu(1-\frac{QL_i}{1+QL_i})} \tag{3}$$

$$= \frac{ST(1+QL_i)}{P_i} \tag{4}$$

以上のように$node_i$におけるピースの平均応答時間を近似的に求めることが出来る.

並列プロセスの応答時間は最も応答時間の長いピースによって影響される. そのため, ノードにピースを割り当てる時には, 割り当てた全てのピースが同じ応答時間で処理されることが望まれる. よって, 負荷分散の方針としてピースが同じ応答時間で処理される, すなわち, $W_i = W_j \ (i \neq j)$ となるようにピースを割り当てることとする.

次に, $W_i = W_j$となる時の$node_i$と$node_j$の平均キュー長の関係を式(5)示す.

$$\frac{ST(1+QL_i)}{P_i} = \frac{ST(1+QL_j)}{P_j} \tag{5}$$

式(5)より, QL_iを求めると

$$QL_i = \frac{P_i}{P_j}(1+QL_j) \tag{6}$$

となり, $node_i$と$node_j$のキュー長の関係式が得られる.

ここで, システム内で最も処理能力の低いノードの処理能力をP_{min}とし. そのノードのしきい値をθ ($\theta = 0,1,2,\cdots$) をすると, 式(6)より$node_i$のしきい値th_iは式(7)で表される.

$$th_i = \frac{P_i}{P_{min}}(1+\theta) \tag{7}$$

VTB法では式(7)を用いて各ノードのしきい値を設定し, システム負荷に応じてθを変化させ, 各ノードのしきい値を変化させる.

3.2 しきい値可変 Bidding アルゴリズムの記述

図8にVTB法のフロチャートを示す. VTB法は, 定期的に収集する負荷情報に基づいて, システム負荷が大きい時は各ノードのしきい値を大きくし, システム負荷が小さい時は各ノードのしきい値を小さくする.

VTB法は以下のように, システム負荷が高いか低いかの判断を行う. 仮想並列マシンを構成するノードのうち, αN (但し,αは $0 \leq \alpha \leq 1$ の定数) 台のノードの負荷がしきい値を越えている時, システム負荷は高いと判断する. また, そうでない時はシステム負荷は低いとみなす.

次に提案するVTB法の動作を述べる.

1. 定期的に収集する負荷情報に基づいて, システム内のノードで負荷がしきい値以下のノードの数 ($n\_BusyNode$) を調べる.
2. $\alpha \leq n\_BusyNode \ (0 \leq \alpha \leq 1)$ であるならばθを1増加する. そうでないならば, θを1減らす. 以上のように, 各ノードのしきい値を再設定する.
3. 自ノードにピースが割り当てられているかどうか調べ, 割り当てられていないならば自ノードにピースを1つ割り当てる.
4. 負荷がしきい値未満であるノードにピースを1つ割り当て, 負荷がしきい値以上であるならばピースを割り当てない.
5. (3), (4) の動作を全てのピースを割り当てるまで繰り返し, 全てのピースを割り当てたら終了する.

3.3 シミュレーションによる評価

VTB法の有効性を, シミュレーションにより従来法である Bidding アルゴリズムと比較することにより評価する. シミュレーションでは,

1. 並列プロセスの応答時間
2. 負荷の集中の抑制

を評価する.

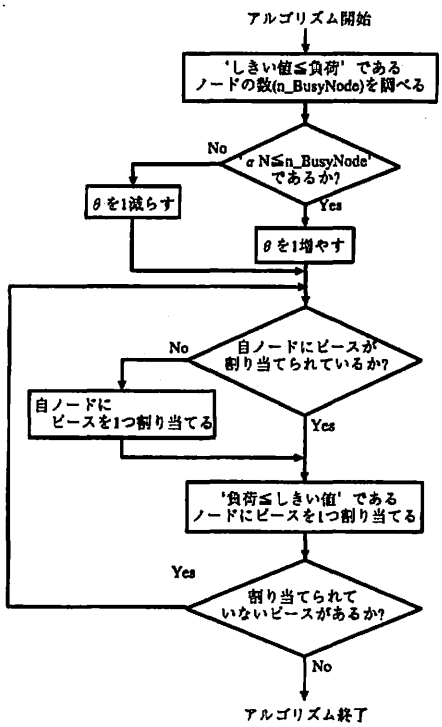

図 8: しきい値可変 Bidding アルゴリズム

3.3.1 シミュレーション方法

[シミュレーション条件]

ノード数	$N = 20$
要求処理量	平均 50 $[I]$ の指数分布
到着率	$\lambda_{bg} = 0.166$
並列プロセスの並列度	20
負荷情報収集間隔	0.2, 0.4, 0.6 $[s]$

[評価基準]

- 並列プロセスの平均応答時間
- *OverLoad* の負荷の平均と分散

  負荷の時間推移から *OverLoad* を評価する．*OverLoad* の定義を式 (8) に示す．

$$OverLoad = \begin{cases} load_i - th_i & (th_i \leq load_i) \\ 0 & (load_i \leq th_i) \end{cases} \quad (8)$$

  ($load_i$: $node_i$の負荷, th_i : $node_i$のしきい値) である．

  OverLoad はノードの負荷がしきい値をどれだけ越えたかを表す値である．

3.3.2 並列プロセスの応答時間の評価

VTB 法を用いる際にはα ($0 \leq \alpha \leq 1$) を設定する必要がある．ここでαとは，システム負荷の高低を判断するための値である．

αを大きくすると VTB 法では各ノードのしきい値を変化しないため，VTB 法による並列プロセスの平均応答時間は従来法におけるしきい値$\theta = 1$の場合の平均応答時間の値に近付くと予想される．一方，αを小さくすると VTB 法は各ノードのしきい値をシステムの状態に反して余分に大きくしてしまうため，並列プロセスの応答時間は従来法に比べ増大すると予想される．よって，VTB 法を用いるには最適なαの値を決定する必要がある．そこで，はじめにαの値をどのように決定したらよいかについて調べる．

- $T = 0.2$ の時，αを変化させた場合の結果を図 9 に示す．

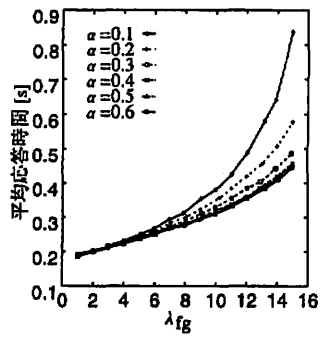

図 9: $T = 0.2$ でαを変化させた場合の並列プロセスの平均応答時間

αが大きくなるほど並列プロセスの平均応答時間は短縮されることが分かる．しかし，$0.4 \leq \alpha \leq 0.6$ で平均応答時間はほぼ等しくなる．このことから，$T = 0.2$ $[s]$ の時は，αは 0.4 付近に設定したほうが良い，すなわち，システム内の約半分のノードの負荷がしきい値を越えたら，そのシステム負荷は高いとみなすべきであることが分かる．

- $T = 0.4$ の時で，αを変化させた場合の結果を図 10に示す．

  並列プロセスの平均応答時間は

  $\alpha \leq 0.4$ の時：αが大きくなるほど短縮される．

  $\alpha \geq 0.4$ の時：αが大きくなるほど増大する．

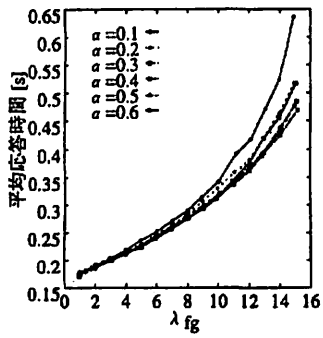

図 10: $T = 0.4$ でαを変化させた場合の並列プロセスの平均応答時間

$0.3 \leq \alpha \leq 0.5$ の時：平均応答時間はほぼ等しい．

このことから，$T = 0.4$ [s] の時はαは 0.4 付近に設定すべきであることが分かる．

- $T = 0.6$ の時で，αを変化させた場合の結果を図 11に示す．

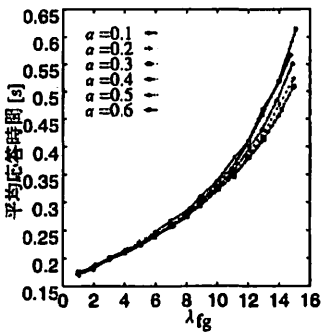

図 11: $T = 0.6$ でαを変化させた場合の並列プロセスの平均応答時間

並列プロセスの平均応答時間は

$\alpha \leq 0.4$ の時：αが大きくなるほど短縮される．

$\alpha \geq 0.4$ の時：αが大きくなるほど増大する．

$0.2 \leq \alpha \leq 0.4$ の時：平均応答時間はほぼ等しい．

このことから，$T = 0.6$ [s] の時はαは 0.4 付近に設定すべきであることが分かる．

以上のシミュレーション結果から，αの最適値は 0.4 付近に存在することが考えられる．つまり，20 台中 8 台以上のノードの負荷がしきい値を越えていたら，VTB 法ではシステム負荷が高いと判断すべきであることが分かる．8 台という値はシステム内のノードのうち高速なノードの台数である．このことからαの値はシステム内のノードのうち高速なノードが占める割合に設定すればよいと思われる．

次に VTB 法と従来法である Bidding アルゴリズムによる並列プロセスの平均応答時間を比較する．VTB 法において$\alpha = 0.4$ とする．また，従来法においてしきい値θが$\theta = 1, 2, 3$ の場合と比較した．

- $T = 0.2$ [s] の時の並列プロセスの平均応答時間を図 12に示す．図では横軸に並列プロセスの平均到着率λ_{fg}を表し，λ_{fg}の増大に対する並列プロセスの平均応答時間の増大の様子を示す．

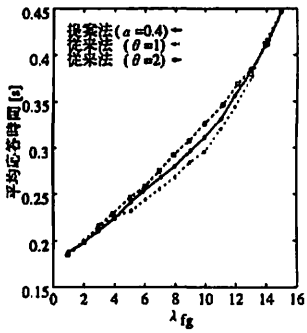

図 12: $T = 0.2$ [s] の時の VTB 法と従来法による並列プロセスの平均応答時間

図 12より，従来法における並列プロセスの平均応答時間は$\theta = 1$ の方が$\theta = 2$ の場合よりも短い．また，VTB 法は従来法の$\theta = 1$ と比べ，あまり差はなく従来法の$\theta = 1, 2$ の場合の中間ぐらいの平均応答時間を示していることが分かる．この理由は，VTB 法ではシステム負荷に合わせて各ノードのしきい値を変化させる．しきい値を小さくさせた時は従来法の$\theta = 1$ の場合に近づき，しきい値が大きくする時は従来法の$\theta = 2$ の場合に近づくためである．

- $T = 0.4$ [s] の時の並列プロセスの平均応答時間を図 13に示す．

  図 13より，従来法において並列プロセスの平均応答時間は$\theta = 2$ の方が$\theta = 1$ よりも良い性能を示すことが分かる．提案法は従来法における$\theta = 2$ の場合と変わらない性能を示すことが分かる．

- $T = 0.6$ [s] の時の並列プロセスの平均応答時間を図 14に示す．

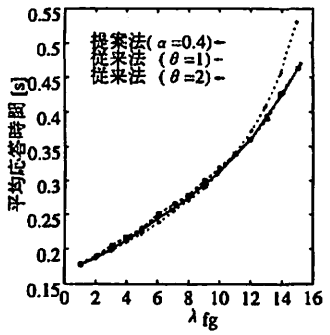

図 13: $T = 0.4$ [s] の時の提案法と従来法による並列プロセスの平均応答時間

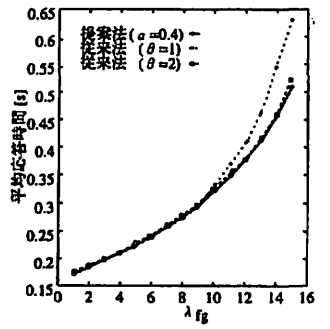

図 14: $T = 0.6$ [s] の時の VTB 法と従来法による並列プロセスの平均応答時間

図 14より，従来法において並列プロセスの平均応答時間は$\theta = 2$ の方が$\theta = 1$ よりも良い性能を示すことが分かる．VTB 法は従来法における$\theta = 2$ の場合と変わらない性能を示すことが分かる．

以上のシミュレーション結果から，VTB 法における並列プロセスの平均応答時間は，従来法において最適なしきい値が分かっているような理想的な状態での並列プロセスの平均応答時間と同等であることが分かる．従来法における最適なしきい値は負荷情報収集間隔 T，並列プロセスの平均到着率λ_{fg}などのパラメータによって変化することが明らかになった．そのため，現実のシステム上で従来法における最適なしきい値を見つけることは困難である．一方，VTB 法では任意の負荷情報間隔，並列プロセスの平均到着率において従来法において最適なしきい値を用いた場合と並列プロセスの平均応答時間の点で同等以上の性能を示す．したがって，現実のシステム上に実装する場合は提案法を用いた方が実装が容易で，かつ，良い性能を示すと考えられる．

3.3.3 負荷の時間推移の評価

以下は各ノードの負荷の時間推移を調べることで，提案法により負荷の集中がどの程度避けられるかを評価する．

従来法，提案法におけるノードの負荷の時間推移の一例をそれぞれ図15,16に示す．図15,16ともに負荷情報収集間隔 $T = 0.2$，並列プロセスの平均到着率$\lambda_{fg} = 15$，従来法におけるしきい値$\theta = 1$ の場合である．

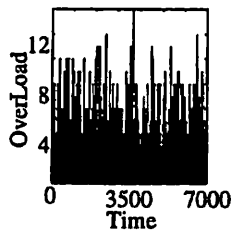

図 15: 従来法

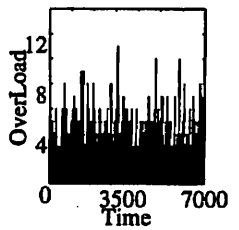

図 16: VTB 法

図 15に比べ 16の OverLoad の値にはばらつきがなく，提案法によって負荷の集中が低く抑えられている傾向があること分かる．

次に *OverLoad* の確率分布を図 17に示す．また，従来法，提案法による *OverLoad* の最大値，平均値，分散を表 1に示す．

表 1: 従来法，提案法による *OverLoad* 最大値，平均値，分散

	VTB 法	従来法
最大値	11	14
平均値	1.219	1.738
分散	3.73	7.109

図 17より，VTB 法における OverLoad の分布は従来法に比べ小さく，これより負荷の集中を低く抑える効果のあることが分かる．また，表 1より VTB 法は *OverLoad* の最大値，平均値，分散ともに従来法に比べて小さい値を示していることが分かる．以上のことから，提案法は従来法に比べ負荷の集中を

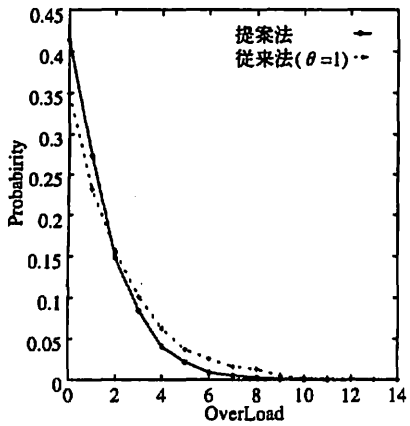

図 17: 従来法と提案法による *OverLoad* の確率分布

低く抑えることができるという優れた特性をもつことが明らかになった.

提案する VTB 法ではシステム負荷に合わせて動的に各ノードのしきい値を変化させる. この VTB 法の特徴を調べるために行ったシミュレーション結果をまとめると, その優れた特性として,

1. 従来法に比べ負荷の集中を低く抑えることができる
2. 最適なしきい値を用いて従来法を動作させた場合の並列プロセスの平均応答時間と同等の値が得られる.

という点が挙げられる.

また, 従来法では負荷情報収集間隔 T, 並列プロセスの平均到着率λ_{fg}のパラメータによって最適なしきい値が変化する. そのため, 現実のシステムに実装する際に最適なしきい値を求めることが必要となる. 一般に, 最適なしきい値を求めることは困難である. それに対し, VTB 法は任意の負荷情報収集間隔 T, 並列プロセスの平均到着率λ_{fg}の場合でも従来法における最適なしきい値を用いた場合の並列プロセスの平均応答時間が得られるため, 実装が容易である.

4 結論

本研究では, 従来仮想並列マシン上で負荷分散を行う際, 用いられる負荷分散手法の多くはバックグラウンドプロセスを対象とするものが用いられてきたにすぎない点に着目し, 並列プロセスを対象とする仮想並列マシン向けの負荷分散アルゴリズムを提案した.

提案法は負荷の集中を避けることを目的としてシステム負荷に合わせて動的に各ノードのしきい値を変化させて負荷分散を行う. 提案法により, 負荷の集中を低く抑えることが出来るようになった. また, 並列プロセスの平均応答時間については, 従来法と同等以上の性能を示すことが出来る. 負荷分散アルゴリズムを実装する時, 従来法では負荷情報収集間隔や並列プロセスの平均到着率などのパラメータによって並列プロセスの応答時間を最小にする最適なしきい値が変化する. そのため, 最適なしきい値を設定することは困難である. しかし, 提案法はシステム負荷に合わせて動的にしきい値を変化させる. そのため, 実装する時は容易に従来法における最適なしきい値を用いた場合と同等の並列プロセスの平均応答時間を得ることが出来る.

今後の課題として, システムの状況に応じてよりすばやく, より柔軟にしきい値を変化させることが考えられる. 一手法として遺伝的アルゴリズムの適用が考えられる. 遺伝的アルゴリズムの適用により, システムの負荷状況の履歴を個体が学習し, 将来の負荷変動を予測してより最適なしきい値を設定できる可能性がある.

参考文献

[1] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vadiy Sunderam "PVM:Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing" The MIT Perss, 1994

[2] Derek L. Eager , Edward D. Lazowska , and John Zahorjan: Adaptive Load Sharing in Homogeneous Distributed Systems
IEEE Trans. Software Eng. , 12 , 5 , pp.662-675 (May 1986)

[3] Songnian Zho "A Trace-Driven Simulation Study of Dynamic Load Balancing" IEEE Transaction on Software Engineering, Vol.14, No.9, September 1988

[4] Songnian Zhou, Jingwen Wang, Xiaohu Zheng, Pierre Delisle "UTOPIA: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems" Techical Report CSRI-257, Computer System Research Institute, April 1992

[5] 松田, 藤井, 根元,"PVM における負荷情報に基づいたタスク割り当て方式", 情報処理学会第 52 回全国大会講演論文集, 3L-7(1996)