社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2005－ＩＣＳ－139（３）
2005／3／14

# 社会性昆虫の反応閾値強化モデルを用いた適応的ネットワーク資源割当に関する研究

森　康真[†]　川村　秀憲[††]　大内　東[††]

† 北海道大学工学部　〒060-8628 北海道札幌市北区北13条西8丁目
†† 北海道大学大学院情報科学研究科　〒060-0814 北海道札幌市北区北14条西9丁目
E-mail: †{mori,kawamura,ohuchi}@complex.eng.hokudai.ac.jp

**あらまし**　複数のサーバから構成される分散型コンピュータシステムにおいて，変動する負荷に対して動的に計算機資源を各サーバに対して割り当てる技術の研究・開発が盛んになっている．本稿では，社会性昆虫の反応閾値強化モデルを用いることで，それらの機能を実現するシステムを提案し，シミュレーションを行うことでその効果を検証する．
**キーワード**　Autonomic Computing, 動的資源割当, 社会性昆虫

# A Study on Adaptive Network Resource Allocation using Response Threshold Reinforcement Model in Social Insect

Yasunao MORI[†], Hidenori KAWAMURA[††], and Azuma OHUCHI[††]

† Faculty of Engineering, Hokkaido University　Kita 13, Nishi 8, Kita-ku, Sapporo, 060-8628 Japan
†† Graduate School of Information Science and Technology, Hokkaido University　Kita 14, Nishi 9, Kita-ku, Sapporo, 060-0814 Japan
E-mail: †{mori,kawamura,ohuchi}@complex.eng.hokudai.ac.jp

**Abstract**　In a distributed computer system configurated by some kinds of servers, it becomes pupular that the research and development of technology which allocate dynamically resource to each server adapiting to changing workloads. In this paper, We propose the dynamic network resource allocation system by using response threshold reinforcement model in social insects, and verify the effect by simulation.
**Key words**　Autonomic Computing, dynamic resource allocation, social insects,

## 1. はじめに

近年，コンピュータシステムは急速に成長している．特に，世界的なインターネットの普及に伴い，巨大エンタープライズシステムの登場，ヘテロジーニアスな環境間の相互接続，ユビキタス環境の構築，Peer-to-Peer アプリケーションの発達などの技術とその必要性が，ネットワークシステムの複雑性を生み出している．この複雑性を人間の手によって管理・制御することは困難になりつつあり，さらに専門的な知識を有するエンジニアの数も不足しており，現状のネットワークシステムが抱える問題点であるといえる．そこで，これらの複雑性をコンピュータ自身に管理させるという概念が最近盛んに提唱され始めている．Jeffrey O.Kephart らは自己構成・自己修復・自己最適化・自己防衛を行なうコンピュータシステム"Autonomic Computing"を提唱している[1]．これは，人間の体が無意識の内に心拍数，呼吸数，血中酸素濃度等に応じて体温等を制御する自律神経にヒントを得たもので，この概念をネットワークシステムに応用することで，人間の介入を最小限に抑えつつ複雑性を制御し問題解決を図るというコンセプトである．この Autonomic Computing と同様な概念やそれらを実装した製品も各方面から登場しており，今後ますますこのような自己管理システムは脚光を浴び，その研究開発も進んでいくものと思われる．

Web サーバやアプリケーションサーバ，そしてデータベースサーバなどのサーバ処理やそれらとは独立したバッチ処理などの複数のタスクから構成される分散型コンピュータシステムにおいて，各タスクに対する要求量と優先度，また補強やメンテナンスによる計算機資源構成は状況に応じて動的に変化する．そこで，どの処理に対してどれだけの計算機資源を割り当てるかを決定する必要がある．望ましい計算機資源割当として，管理者が設定した優先度に従ってタスクを実行するように割り当てることが可能であること，環境変動への追従性があること，

そして資源割り当ての事前設定に多くのパラメータの調整が不要であることなどが挙げられる．Tesauro らは，Web 処理とバッチ処理に対し効用関数を定め，全体効用が最大になるようにそれらの処理に計算機資源を割り当てるシステムを提案した [2]．この研究では二種類の処理を扱っているが，実際の分散型コンピュータシステムではさらに多くの種類の処理から構成されており，複数の処理に複数の計算機資源を動的に割り当てる手法を用いることが望まれる．そのような仕組みを実現し，システム構成の変動に追従でき，ネットワークの分散性・並列性に対応できる手法として，社会性昆虫の反応閾値強化モデルがある．このモデルは，蟻や蜂などの社会性昆虫と呼ばれる種が見せる振る舞いをモデル化したものであり，多数の個体が複数のタスクを分散的に実行し，Stigmergy と呼ばれる環境を介在したコミュニケーションを利用することによって環境変動に対応しながら群れ全体の目的を達成するために柔軟に労働力を配分するという特徴を持っている [6]．そこで本稿では，これらの資源割当問題と社会性昆虫の反応閾値強化モデルの特徴の共通性に着目し，ネットワーク上での各サーバが持つ機能などをタスクとみなし，それらの処理を担当する計算機資源を一つ一つの社会性昆虫の個体とみなす事で，問題をモデル化し，コンピュータ上で構築したシミュレーション環境を用いて，その効果を実験によって検証することを目的とする．シミュレーションを行なうのは，実機実験の前に提案システムが効果を挙げられる状況を事前に検証でき，多数の資源を扱う実験を実機を用いずに低コストで実現できるためである．

本稿の構成を述べる．第 2 章で本稿が扱う問題とそのモデル化について説明を行なう．第 3 章は本システムで用いる社会性昆虫の反応閾値強化モデルとその問題への適用についてである．第 4 章で実験・考察を行い第 5 章でまとめる．

## 2. モデル化

本稿では、時間と共に変動する環境に対応するために計算機資源を各処理に割り当てる，ネットワーク資源割当問題を扱う．以下にモデルを構成する資源，プロセス，ジョブとその処理の流れ (正の整数離散時間 $\{t|1,2,\cdots\}$ に従って進む) を説明する．

### 2.1 資　源

1 つの資源は 1 台の計算機と想定する。各資源 $R=\{r_i|1 \leqq i \leqq N\}$ は 3 つの要素 $CPU_i$,$SERVICE_i(t)$, $remainCPU_i(t)$ とサーバプロセスの種類と同数のキューからなる．$CPU_i$ は単位時間当たりに資源 $i$ が実行可能な CPU サイクル数である．CPU サイクル数は処理の最小単位であるとする．$SERVICE_i(t)$ は時刻 $t$ において資源 $i$ が行なうプロセス集合の要素を，$remainCPU_i(t)$ は実行可能な CPU サイクル数の残量を表しており，0 以上の整数値を取り，下記で説明するジョブを処理する毎に減少する．各資源にはサーバプロセスに対するキュー集合 $QUE=\{que_{il}|1 \leqq i \leqq N, 1 \leqq l \leqq M\}$ があり，資源 $i$ のプロセス $l$ にジョブが到達したとき対応したキューである $que_{il}$ に並ぶ．

### 2.2 プロセス

プロセス $PR=\{pr_j|1 \leqq j \leqq L\}$ は資源が実行する処理であり，サーバを稼動させるサーバプロセス $S=\{pr_j|1 \leqq j \leqq M\}, S \in PR$ とバッチ処理を行なうバッチプロセス $B=\{pr_j|M+1 \leqq j \leqq L\}, B \in PR$ の二つに分かれる．1 つのサーバプロセスに $A$ 種類の処理が設定されているとして，1 個のジョブに対してそのサーバプロセスを完了させるのに必要な CPU サイクル数 $STEP_l^a(1 \leqq a \leqq A)$ が定義される．もし 1 つのサーバプロセスに設定される処理が 1 種類ならば単純に $STEP_l$ と書くこととする．

### 2.3 ジョブ

ネットワークシステムに対するアクセス要求はポアソン分布に従い発生し，アクセス要求があったときにジョブを生成することとする．各ジョブ $J=\{j_p|1 \leqq p \leqq n\}$ は 4 つの要素 $p_p(t)$,$st_p(t)$,$jt_p(t)$,$jtt_p(t)$ を持っている．時刻 $t$ において $p_p(t)$ はこのジョブが所属するサーバプロセスを，$st_p(t)$ はジョブの現在の状況を表しており，ジョブがまだ生成されていない未到着状態であるか (notarrive)，キューに並んでいるか (wait)，資源から処理を受けているか (active)，その処理が終了したので次のサーバプロセスを行なう資源へ移行可能な状態 (move) であるか，そして全てのサーバプロセスでの実行を終えて完了状態になった (complete) かどうかを表す．$jt_p(t)$ はこのジョブが発生してからの経過時間であり，$jtt_p(t)$ は各資源で実行された CPU サイクル数を表す．この $jtt_p(t)$ が各サーバプロセスに設定されている処理完了に必要な CPU サイクル数 $STEP_l^a$ を満たしたときに次のサーバプロセスへ移行できるとする．これらサーバプロセスの実行順はリストで定義する．

### 2.4 処理の流れ

ある時刻 $t$ において，ジョブの添え字に従って順にそのジョブの状態が決定される．ジョブが生成されない間は $st_p(t)$ を notarrive とし，そのジョブがキューの先頭以外に並んでいるならば，資源から処理を受けられない状態であり，$st_p(t)$ を wait とする．そのジョブが所属する資源のサーバプロセス別キューの先頭にいるならば，資源から処理を受けることができる状態であり，$st_p(t)$ を active とする．次に，同時刻において資源の添え字に従ってキューの先頭のジョブに対する資源による処理が以下のように行なわれる．

if($remainCPU_i(t) \geqq STEP_l^a$)
　$remainCPU_i(t) = remainCPU_i(t) - STEP_l^a - jtt_p(t)$
　$st_p(t) = move$
else if($remainCPU_i(t) < STEP_l^a$)
　if($remainCPU_i(t) \geqq STEP_l^a - jtt_p(t)$)
　　$remainCPU_i(t) = STEP_l^a - jtt_p(t)$
　　$st_p(t) = move$
　else
　　$jtt_p(t) = jtt_p(t) + remainCPU_i(t)$

$st_p(t)$ が move となったジョブはキューから取り除かれ，$jtt_p(t)$ を 0 として，次のサーバプロセスへと移行可能になる．この結果ジョブが実行することを必要とする全てのサーバプロセスを実行し終えた場合には $st_p(t)$ を complete とする．2 番目に並んでいたジョブが先頭になり，$remainCPU_i(t)$ が 0 となるかキューが空になるまで処理が続く．次の時刻で $remainCPU_i(t)$

に $CPU_i$ を代入することで CPU サイクル数が回復する．

### 2.5 資源割当行列

資源 $r_i$ は時刻 $t$ において全てのプロセスの中で 1 つを実行可能であるとし，資源 $r_i$ がプロセス $pr_j$ を実行可能を $E = \{e_{ij}|1 \leqq i \leqq N, 1 \leqq j \leqq L\}$ で表し，$e_{ij} = 0$ のとき実行しない状態，$e_{ij} = 1$ のとき実行する状態であるとする．このとき，以下の資源割当行列 $AL(t)$ を定める．

$$AL(t) = \begin{pmatrix} e_{11} & e_{12} & \cdots & e_{1L} \\ e_{21} & e_{22} & \cdots & e_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ e_{N1} & \cdots & \cdots & e_{NL} \end{pmatrix} \quad (1)$$

この行列の値を決定することを資源割当と定義する．従って本問題はこの値を環境の変動に合わせてどのように更新していくかを求めるものである．

## 3. 社会性昆虫の反応閾値強化モデル

アリやミツバチなどの社会性昆虫の群れの中では採餌活動や巣の防衛活動など，異なるタスクがそれぞれのタスクに特化した各個体によって同時に行なわれており，この現象は仕事分割 (division of labor) と呼ばれている [3]．この社会性昆虫の振る舞いについて，Bonabeau らは反応閾値を用いてモデル化を行なった [4]．このモデルでは，全ての個体は全てのタスクに対してそれぞれ反応閾値 $\theta_{ij}$ を，そしてタスクは刺激強度 $s_j$ を持っている．$\theta_{ij}$ は個体 $i$ のタスク $j$ に対する実行しやすさを表し，タスクに対する値が低いほどそのタスクに反応しやすくなる．$s_j$ はタスク $j$ が発する刺激であり，大きいほどそのタスクが個体による処理を必要としている状態を表している．これらのパラメータより個体 $i$ がタスク $j$ を実行する確率 $P_{ij}(t)$ を離散時間ステップ毎に以下の式で定める．

$$P_{ij}(t) = \frac{s_j^2(t)}{s_j^2(t) + \theta_{ij}^2(t)} \quad (2)$$

これはシグモイド曲線を意味しており，$s_j = \theta_{ij}$ となるときに $P_{ij}(t) = 0.5$ となる．また，$\theta_{ij}$ は離散時間ステップ毎に更新される．もし個体 $i$ がタスク $j$ を実行した場合には

$$\theta_{ij}(t+1) \longleftarrow \theta_{ij}(t) - \xi \quad (3)$$

もし個体 $i$ がタスク $j$ を行なわなかった場合には

$$\theta_{ij}(t+1) \longleftarrow \theta_{ij}(t) + \psi \quad (4)$$

という式で更新される．ここで $\xi$ は学習係数を，$\psi$ は忘却係数を表している．この更新式は，あるタスクを行なった場合にはそのタスクに対する反応閾値が引き下げられその個体は今後よりそのタスクを行ないやすくなるが，それ以外のタスクの反応閾値は引き上げられるので実行確率が下がるということであり，タスクに対する専門化が時間を追うごとに進むことを意味している．また閾値には最小値 $\theta_{min}$，最大値 $\theta_{max}$ が定められる．

**問題への適用**

本節では本稿が扱う問題，すなわち資源割当行列の値を動的に更新するための解決策として社会性昆虫の反応閾値強化モデルを適用する．時刻 $t$ の資源 $i$ において $\theta_{ij}(t)$ はプロセス $j$ に対する反応閾値を，$s_j(t)$ は各プロセスにおける刺激強度を，そして $P_{ij}(t)$ は資源 $i$ のプロセス $j$ に対する優先度を表している．優先度が最大のプロセスに対して資源割当が行なわれる．

本稿では 3 つの設定を用意する．設定 1 と 2 では優先度を決定する式の違いによる影響の観察を，設定 3 は理想的な状況を想定することで設定 1，2 との比較を行なうことを目的としている．設定 1 では資源 $i$ のプロセス $j$ に対する優先度 $P_{ij}(t)$ を式 2 のように定める．また，設定 2 では

$$P_{ij}(t) = \frac{s_j^2(t)}{s_j^2(t) + \alpha\theta_{ij}^2(t) + \beta(\frac{1}{d_l(t)})^2} \quad (5)$$

とする．$s_j$ はプロセス $j$ の刺激強度であり，サーバプロセスの場合にはキューに並んでいるジョブ数，バッチプロセスの場合には任意の値を設定することとする．$\alpha$ と $\beta$ は係数である．$d_l(t)$ は時刻 $t$ においてサーバプロセス $l$ のキューに並んでいるジョブを全て実行完了させるために必要な時間であり，

$$d_l(t) = \frac{|Que_l(t)| \times AVESTEP_l}{SUMCPU_l(t)} \quad (6)$$

で表される．$|Que_l(t)|$ は時刻 $t$ においてサーバプロセス $l$ を実行する各資源のキューに並んでいるジョブの数の和であり，$AVESTEP_l$ はサーバプロセス $l$ が一つのジョブを完了させるのに必要な CPU サイクル数である．1 つのサーバプロセスに複数の処理が設定されている場合にはそれらの $STEP_l^a$ の平均を与えることとする．また，$SUMCPU_l(t)$ は同時刻にサーバプロセス $l$ に割り当てられている全資源の $CPU_i$ の和である．式 5 はジョブを処理し終わるのに時間が多くかかるプロセスに対する優先度は大きくなり，時間がかからないプロセスに対する優先度は小さくなることを意味している．これら設定 1,2 では，資源の処理待ちジョブ数を監視して，最もジョブ数が少ない資源にリクエストを割り振ることとし，もし同じジョブ数を抱える資源が複数個あった場合には $CPU_i$ の高い資源にジョブを送り出すという負荷分散アルゴリズムを用いる．

設定 3 は資源のキューに並んでいるジョブをプロセス切り替え後の各資源に均等配分していく設定 1,2 より理想的な設定であり，優先度は式 2 によって与える．

また，離散時間ステップ毎に閾値の更新が起こり，この閾値と刺激強度を元に資源の各プロセスに対する優先度が定まる．本稿ではこの優先度の和が 1 になるように正規化するか，あるいはそのままの数値で扱うかのいずれかを選択し，最も優先度が高かったプロセスに対して実行を切り替えるものとし，資源割当行列の値を更新する．すなわち各資源 $r_i$ の添え字に従って最も優先度の高い $pr_j$ を求め，資源割当行列の要素 $e_{ij}$ に 1 を代入し，$i$ 行のその他の要素に 0 を代入するものとする．ただし，あるプロセスを担当している資源が 1 台しか無かった場合にはプロセス切り替えを行なわないものとする．予備実験の結果から閾値更新毎に資源割当に極端な振動現象が見られなかった設定として，設定 1,2 は後者，設定 3 は前者の方法を用いる．

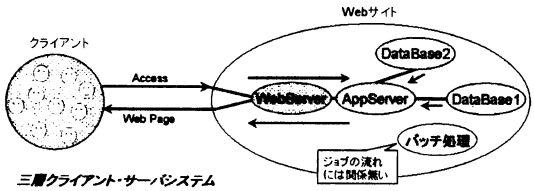


図 1　ジョブの流れ

## 4. 実験・考察

前章で構築したモデルを元に，本章ではコンピューターシミュレーションを行なう．以下の実験は全て JAVA SDK 1.4.2_06 を用いてプログラムを作成し実行した．まず実験 1 では $STEP_l^a$ の値をある時刻で変更しサーバプロセスの処理待ちジョブ数の推移を見ることで本モデルによる動的な資源割当の有効性を示す．次に実験 2 で平均到着率の増加による影響を観察し資源追加の効果を示す．最後に実験 3 においてバッチ処理に対する刺激強度によってバッチ処理量を制御する実験を行う．

### 4.1 共通設定

全ての実験は時刻 $t = 30000$ まで行い，一つの設定につき 30 回の試行を行なった．学習を行なう際の時間間隔は 10 ステップとした．ポアソン分布の平均到着率 $\lambda = 2.0$ に従って発生したジョブは図 1 のようにまず $pr_1$:Web サーバにアクセスを行い，次に $pr_2$:アプリケーションサーバ (以下 App サーバ) に処理が転送される．そこでジョブは $pr_3$:DataBase サーバ 1 と $pr_4$:DataBase サーバ 2 へアクセスし，その処理を待つ．処理が完了したジョブは再び $pr_1$:Web サーバを通りクライアントの元へ返る．これは Web サイトの構成時によく用いられる 3 層クライアント・サーバシステムをモデルにしている．また，このジョブの流れとは関係の無い，バッチ処理というプロセスを設けておく．各サーバプロセス別の $STEP_l^a$ は表 1 のよう設定する．すなわち，App サーバと DataBase サーバ 2 における $STEP_l^a$ を他より高く設定することによって，意図的にボトルネックとなる部分を作っている．Web サーバプロセスでは，行き (App サーバに到達する前) の処理による必要 CPU サイクル:$STEP_1^1$ と帰り (App サーバに到達した後) の処理による必要 CPU サイクル:$STEP_1^2$ あるので，それらの $STEP_l^a$ の平均を $AVESTEP_l$ として与える．初期設定として，5 つのプロセスにはそれぞれ単位時間当たりの処理能力が 100，200，300 の資源を 1 台ずつ割り当ててある．$pr_5$:バッチ処理の刺激強度には 50 を設定した．予備実験の結果からプロセスに並ぶジョブ数を単調増加させずに安定した推移を取るパラメータ設定として，$\xi = 1.5$，$\psi = 1.0$，$\alpha = 0.5$，$\beta = 750$ を用いる．また，$\theta_{ij}$ の初期値は 500 とし，$\theta_{min} = 1$，$\theta_{max} = 1000$ とする．

表 1　実験設定 各プロセス別 $STEP_l^a$

| | |
|---|---|
| $pr_1$:Web サーバ (行き) | 50 |
| $pr_1$:Web サーバ (帰り) | 100 |
| $pr_2$:App サーバ | 400 |
| $pr_3$:DataBase サーバ 1 | 200 |
| $pr_4$:DataBase サーバ 2 | 500 |

### 4.2 実験 1

まず本システムによる動的な資源割当の有効性を示す．表 1 を元に理想的な資源割当を表 2 の通りに予め決めておき実験中は割当を変更しない場合と，前述のモデルの通り閾値強化により優先度を変動させ動的割当を行なった場合の比較を行なう．途中，表 3 に示すように Web サーバと DataBase サーバ 1 において必要とする 1 個のジョブ当りの $STEP_l^a$ を実験の途中で変更し，それぞれの適応性を見る．また，各プロセスに割当られる資源の $CPU_i$ の和を処理力と表現する．各プロセスのキューに並んでいるジョブ数の推移をそれぞれ図 2, 図 3 に示す．動的割当を行なわない場合は，DataBase サーバ 1 の $STEP_l^a$ を変更する前までは処理待ちジョブ数の増加を抑えているが，設定変更した時刻 10000 以降には $STEP_l^a$ を増加させた DataBase サーバ 1，そして時刻 20000 以降には Web サーバでの処理待ちジョブ数が増加している．一方で動的割当を行なった場合には，閾値更新が進み資源割当が多発するために時刻 5000 までは安定しないが，その後は設定の変更にも適応している．図 4 に示すように，時刻 10000 以降は $STEP_l^a$ が増加した DataBase サーバ 1 に対して，時刻 20000 以降は Web サーバに対しての処理力の割当が増加している．このプロセスに対する処理力割当の変動は，第 2 章で述べた反応閾値の更新によりプロセスに対する優先度が変動し，その確率を元に各資源がプロセスを切り替えているために生じている．

このように，多くの $STEP_l^a$ を必要とするサーバプロセスに対して動的に資源の処理力を割当てるという適応的な資源割当によって処理待ちジョブ数の単調増加を防いでいることが確かめられ，動的な資源割当の有効性が示された．

表 2　実験 1 割当を変更しない場合の資源割当構成

| | $CPU_i = 100$ | $CPU_i = 200$ | $CPU_i = 300$ | $CPU_i =$ 合計 |
|---|---|---|---|---|
| Web サーバ | 1 台 | 1 台 | 0 台 | 300 |
| App サーバ | 2 台 | 0 台 | 2 台 | 800 |
| DataBase サーバ 1 | 1 台 | 0 台 | 1 台 | 400 |
| DataBase サーバ 2 | 1 台 | 3 台 | 1 台 | 1000 |
| バッチ処理 | 0 台 | 1 台 | 1 台 | 500 |

表 3　実験 1 各サーバプロセス別 $STEP_l^a$

| | 初期設定 | 時刻 10000 | 時刻 20000 |
|---|---|---|---|
| $pr_1$:Web サーバ (行き) | 50 | 50 | 150 |
| $pr_3$:DataBase サーバ 1 | 200 | 300 | 100 |

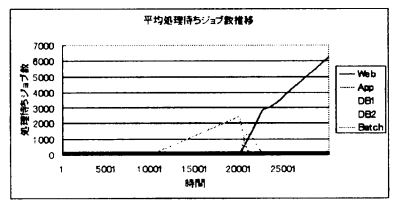


図 2　実験 1 動的割当を行なわない場合 (10 回平均)

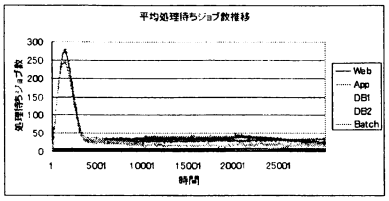


図 3　実験 1 動的割当を行う場合

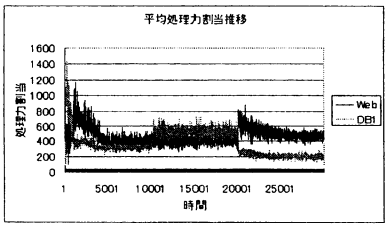


図 4　実験 1 平均処理力割当推移

### 4.3 実　験　2

本実験では，途中でジョブの平均到着率と資源の数を変化させた場合のシステムの挙動を観察する．時刻 10000 においてジョブの平均到着率を $\lambda = 2.4$ に変更する．これは Web サイトの google におけるページランクが上がることでアクセス数が増加するような状況を表している．次に時刻 15000 において単位時間当たり合計 1000 の処理力の資源を追加する．何台の資源によって合計 1000 の処理力を与えるかが問題となるが，事前に行なった実験の結果よりジョブ 1 個当りの完了時間が最短で，完了したジョブ数が最大であった設定として，設定 1,2 では $CPU_i = 250$ の資源を 4 台，設定 3 では $CPU_i = 100$ の資源を 10 台追加することとする．

図 5 は各設定における全プロセスのキューに並んでいる処理待ちジョブ数の総計の推移である．平均到着率増加後は各設定ともジョブ数は増加しており，資源追加後には減少した後安定している．図 6 は設定 1 の，図 7 は設定 2 における平均処理力割当推移である．これらの設定の結果には大きい差は見られなかった．平均到着率増加後にバッチ処理に対する処理力割当が減少しており，より多くの処理力を必要とする他のプロセスに割当られていることがわかる．資源追加後にはバッチ処理に対する処理力割当は元の水準に回復している．また，設定 3 は処理待ちジョブをプロセス切り替え時に資源に均等配分する設定であるために設定 1,2 より資源の処理力に余裕がある状態である．その結果，平均到着率増加後はバッチ処理への処理力割当は減少するものの設定 1,2 と比較して高い値を保っており，資源追加後には平均到着率増加前よりも多くの処理力がバッチ処理に割当られていることがわかる (図 8)．一方でその他のプロセスに対する処理力割当は資源追加後も安定していることから，余った資源処理力がバッチ処理に対して割り当てられていることがわかる．これはバッチ処理の刺激強度を 50 に設定してあるために，図 9 に示すようにその他のプロセスの処理待ちジョブ数が 50 以下で推移しているときにはバッチ処理に対する優先度が高くなるためである．この性質を用いて，バッチ処理に対する刺激強度という一つのパラメータを変更することのみで 1 個当りのジョブ完了時間・完了個数，そしてバッチ処理への資源処理力割当量に影響できる可能性が示唆される．

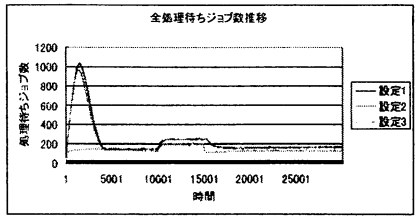


図 5　実験 2 全処理待ちジョブ数推移

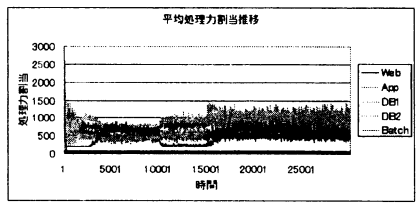


図 6　実験 2 設定 1 平均処理力割当推移

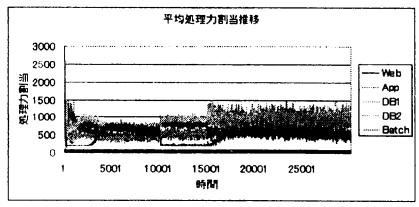


図 7　実験 2 設定 2 平均処理力割当推移

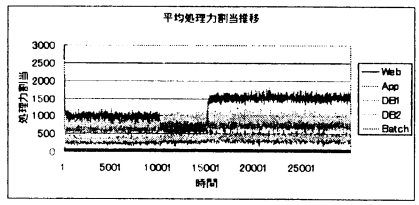


図 8　実験 2 設定 3 平均処理力割当推移

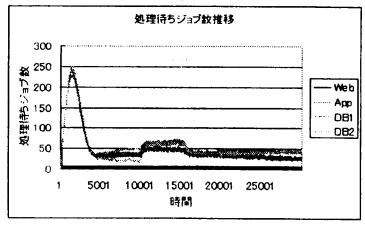


図 9　実験 2 設定 3 プロセス別処理待ちジョブ数推移

### 4.4 実 験 3

実験 3 では，バッチ処理に対する刺激強度の値の変動によるバッチ処理への資源割当量に対する影響について考察する．本実験の中で，バッチ処理はアクセス要求によるジョブの処理の流れとは関係が無い独立したプロセスである．これは例えばある Web サイトにおいて，コンテンツへのアクセス要求を受け付けるのと同時にコンテンツを充実させるため (例えば検索エンジンのクローリング処理など) にアクセス要求を処理するのとは別の処理を行なわなければならない状況をモデル化してある．この状況において，Web サイト運営者は全体の資源処理力に対してどの程度の処理力をバッチ処理のような独立したプロセスに当てるかを決定しなければならない．Y.Diao らは Web サイト運営者が資源の CPU とメモリの目標使用率を定め，そのポリシーに従って Web サーバ (Apache) の設定を変動する負荷に対して自動設定するシステムを構築した [5]．このように，サイト運営者が資源の能力に対してどれだけの処理力をコンテンツへのアクセス処理に割り当てるかについてのポリシーを設定し，それに応じて自律的に処理力割当を行なうという仕組みを，この社会性昆虫の反応閾値モデルの刺激強度の設定変更により実現できるかを確かめることを目的として実験を行なう．

各設定に対して，バッチ処理に対する刺激強度をそれぞれ 50,150,250 とすることによって実験を行なった．結果を図 10〜図 12 に示す．

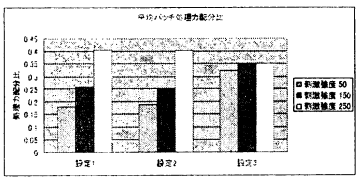


図 10 実験 3 平均バッチ処理力配分率数 比較

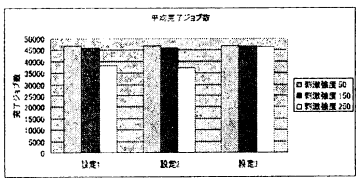


図 11 実験 3 平均完了ジョブ数 比較

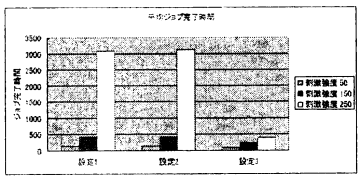


図 12 実験 3 平均ジョブ完了時間 比較

各設定において刺激強度を大きくするに従ってバッチ処理に対する処理力配分の比率が高くなっていることがわかる．一方で処理を完了できたジョブ数は減少し，ジョブ 1 個当りの完了時間は増大するというトレードオフ関係が見られる．設定 1,2 では刺激強度を 250 とした場合ではジョブ完了時間は飛躍的に増加しており，アクセス要求を行なうクライアントにとっては望ましくない結果であるといえる．もし双方を共に改善したいのであれば実験 2 で示した資源を追加することによる効果を得る必要がある．だが一方で資源を追加すること無く，限られた資源処理力の中でジョブを処理する一連のサーバプロセスと，それとは独立したバッチプロセスのそれぞれに対して，管理者が希望する配分に応じて自律的に処理力を割り当てる機能も必要である．その機能を社会性昆虫の反応閾値モデルにおける刺激強度の値を増減させるだけで実現できる可能性が実験により示されたが，刺激強度の設定のみで任意のジョブ完了時間・完了個数，そしてバッチ処理への資源処理力割当量を得ることについては，まだこれから研究の必要性があるといえる．

## 5. おわりに

本稿では社会性昆虫の反応閾値強化モデルを用いて適応的にネットワーク資源割当を行なうシステムを提案し，そのモデル化とシミュレーション環境を構築し，実験を行なった．サーバプロセスの必要 CPU サイクル数の変更，ジョブの平均到着率の変更，そして資源の追加によって動的な負荷状況を生み出し，これに対するシステムの挙動を観察した．その結果，資源割当を動的に変更することができ，処理待ちジョブ数を安定させることができた．また，バッチ処理に対する刺激強度の設定によりシステム全体に影響を及ぼすことがわかった．今後はより現実に即したモデルの拡張を行なうとともに，実機での実証実験を行なう必要がある．

**謝辞** 本研究を進めるにあたり，Autonomic Computing に関するアドバイスをして頂いた日本アイ・ビー・エム（株）大古俊輔氏に感謝いたします．

### 文 献

[1] Jeffrey O.Kephart, David M.Chess, "The Vision of Autonomic Computing", IEEE Computer, 36(1):41-52, 2003.

[2] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, Steve R. White, "A Multi-Agent Systems Approach to Autonomic Computing", Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS'04), pp. 464-471, 2004.

[3] Eric Bonabeau, Guy Theraulaz, Marco Dorigo, "Swarm Intelligence: From Natural to Artificial Systems", Oxford Univ Pr (Sd), 1999.

[4] Eric Bonabeau, Andrej Sobkowski, Guy Theraulaz, Jean-Louis Deneubourg, "Adaptive task allocation inspired by a model of division of labor in social insects", In Bio-Computation and Emergent Computing, edited by D.Lundh, B. Olsson, and A.Narayanan, 36-45. Singapore: World Scientific, 1997.

[5] Y.Diao, J.L.Hellerstein, S.Parekh, J.P.Bigus, "Managing Web server performance with AutoTune agents", IBM SYSTEM JOURNAL, VOL 42, NO 1, 2003.

[6] 大内 東, 山本 雅人, 川村 秀憲, 柴 肇一, 高柳 俊明, 當間 愛晃, 遠藤 聡志, "生命複雑系からの計算パラダイム—アントコロニー最適化法・DNA コンピューティング・免疫システム", 森北出版, 2003.