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

2006－ＤＳＭ－41（９）
2006／5／12

サービスレベル管理のための優先制御機能を有したヘテロジニアス環境適応型サーバ容量計画法

中台 慎二　谷口 邦弘

NEC インターネットシステム研究所　〒211-8666 川崎市中原区下沼部 1753

E-mail: s-nakadai@az.jp.nec.com, k-taniguchi@da.jp.nec.com

あらまし　ワンセグなどの通信放送融合の進展により、Web サイト負荷の変動が高まると懸念されている。負荷変動の大きな環境においてサービスレベルを維持する為には、負荷に応じて適応的にサーバを割当てる自律的な運用管理が有用である。これにはサービスレベル維持に必要なサーバ容量を算出する機能が必要である。しかし、従来はサーバ台数のみを算出する機能しかなく、サーバ構成がヘテロジニアスなデータセンタへの適用は困難であった。また、サービス間に優先制御をする機構がなく、重要度の低いサービスがサーバを確保したまま、重要度の高いサービスの品質が劣化するという問題があった。本提案技術では、ヘテロジニアス環境に対応するために、サービス品質/負荷およびサーバ性能の関係をモデリングし、これと負荷分散装置の品質を均衡化する振る舞いとから導出される整数計画問題を解く。また、優先制御を実現するためにファジィ制御を導入する。

キーワード　サービスレベル管理，自律制御，ファジィ制御，ヘテロジニアス，優先制御，サーバ容量計画法

Server Capacity Planning with Class of Service Function for Service Level Management in Heterogeneous Environment

Shinji NAKADAI,　Kunihiro TANIGUCHI

Internet Systems Research Laboratories, NEC Corporation

1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, 211-8666

Abstract Web sites associated with TV programs are in the face of sharp fluctuation in a load. To keep the quality of such services, it is useful to allocate servers in accordance with the load. Such an autonomic service level management system requires a server capacity planning function. However, existing capacity planning functions are not applicable to heterogeneous environments, and they lack a priority allocation function, which avoids undesirable situation that the quality of important service deteriorates. In our approach, the capacity planning in heterogeneous environments is realized by the resolution of an integer programming that is induced by modeling of servers and services and quality-equalization function of load balancers. The priority allocation function is realized by a fuzzy control.

Keyword Service Level Management, Self-Management, Fuzzy Control, Heterogeneous, CoS, Capacity Planning

1. はじめに

1.1. Web サイトにおける負荷変動

Web サイトは、時に突発的な負荷変動を受ける。例えば、大規模災害や事件が発生すると、多くのユーザが情報入手などのために Web サイトにアクセスする。このような負荷変動は、品質劣化を引き起こすだけでなく、サービスの継続も困難にさせることさえある。

さらに、この負荷変動の発生頻度は、通信放送融合の進展により増加すると予想される。Web サイトが放送番組と連動することで、ユーザのアクセスタイミングが同期しやすくなるからである。

1.2. サーバ動的割当の有用性

負荷変動の大きな Web サイトのレスポンスタイム品質を維持する手段として、複数サービスで共有するプールサーバを負荷変動に応じて割当てることで、低コスト化と想定外負荷への対応が実現できる。仮に、想定されるピーク負荷を対処可能な規模のサーバをサービス毎に固定的に割当てると、低負荷時には遊休状態になるため投資対効果が悪化する。

このような負荷変動に応じたサーバ動的割当は、人手ではなく、自律的に行われることが望ましい。人手では作業頻度・量が増加し、人件費や作業ミスによる障害が増加するからである。

本研究は従来行われている自律的なサーバ動的割当に関する研究[1][2]の中で、特にサーバ容量計画に属するものである。サーバ容量計画とは、どのような資源をどれ程割当てれば、契約等で定められたサービスレベルを維持できるかを判断する機能である。

本書の構成を示す。想定する要件とこれに対応する関連研究、および本研究の特長を2章で述べる。提案手法と実験結果を3章と4章で各々述べ、5章でまとめる。

2. 要件および関連研究

2.1. 自律的サービスレベル管理の要件

自律的サービスレベル管理システムの要件を7項目挙げる。要件設定にあたっては、ホスティング事業者がデータセンタを運営し、サービスプロバイダからサービス運用を請け負うことを想定した。

A) サービスレベル維持

負荷変動を受けても、契約で定まったサービスレベルに違反しないように品質(QoS : Quality of Service)を維持する必要がある。

B) バースト負荷対策

予想外のバースト負荷発生時にも、サービスレベル違反を最小限にする必要がある。

C) ヘテロジニアス環境対応

異なる性能のサーバが混在するヘテロジニアス環境であっても、性能の差異を考慮し適切量のサーバを割当てる必要がある。本来必要な性能に満たない性能のサーバが割当てられると品質が低下し、必要以上の性能のサーバが割当てられると稼働率が低下する。

D) 自律システムの安定化

制御は、発振/振動が少なく安定的である必要がある。

E) 優先制御の実現

資源の少ない際に優先度(CoS : Class of Service)の高いサービスに優先割当する機能が必要である。これは、重要度の低いサービスが資源を確保しつつ、重要度の高いサービスの品質が劣化する状況を回避する。

F) 多層構成対応

サービスが多層のサーバから構成されている場合には、必要な層にサーバ割当を行うことが必要である。

G) 課金を考慮した資源割当

コストモデルの導入など、サービスプロバイダが過剰な QoS/CoS 設定をしない動機付けが必要である。

2.2. 従来研究

前節要件と対応する従来研究について述べる。従来、サービス負荷変動に対して適応的に制御を行うものとして、サーバ内パラメータ制御とクラスタ制御があった。サーバ内パラメータ制御は、最大クライアント数等のサーバ内パラメータを制御する方式[3][4][5]である。クラスタ制御は、クラスタを成す複数サーバの台数や構成を制御する方式である。後者は、前者に比べて複雑な依存関係を考慮して構成変更する必要があり、長い制御時間も要するが、制御によって得られる品質向上効果が高い。

サービスレベル維持技術としては、サーバ内パラメータを制御する制御理論ベースの研究[3]がある。また、制御対象システムのモデリングとして、システム変数間を動的に相関分析/重回帰分析する研究がある[6]。

突発的な負荷対応技術としては、負荷予測機能付き資源割当[1]がある。構成変更に要する時間と同等の時間分だけ将来のリクエスト到着率を自己回帰和分移動平均(ARIMA)モデル[7]で予測し、予測リクエスト到着率で目標レスポンスタイムが維持可能となるようにサーバを割当て、突発的負荷に対処する。

ヘテロジニアスなサーバ環境下でクラスタ構成を制御する技術は、従来行われていない。

制御の安定化は、サーバ内パラメータの PID (Proportional, Integral and Derivative)制御にて、固有値が安定条件を満たすように係数が選択されることで実現されている[3]。

サービス間優先度割当を実現可能なクラスタ制御方式については、従来行われていない。

多層構成対応については、サーバ間構成を待ち行列網モデルでモデル化し、これを平均値解析(MVA: Mean Value Analysis)法で層間の負荷情報を計算することで、必要な層にサーバ割当量を見積もる研究[2]が行われている。

課金体系を考慮した資源割当として、Fuzzy 制御により利益を最大化するアドミッション制御[8]がある。

2.3. 本研究の特長

本研究では、前記要件のうち A)～E)を満たすサーバ容量計画法を提案する。特に C)ヘテロジニアス環境に対応し、E)優先制御をクラスタ制御で実現していることを特長とする。

3. 提案手法

提案する制御フレームワークと制御対象の構成について 3.1 と 3.2 で述べ、主要構成モジュールを 3.3 以降で述べる。

3.1. 制御フレームワーク

本方式は、フィードバック制御系とフィードフォワード制御系からなる。これら２つの系のブロック線図を図１に示す。また、これを元に設計した管理システムのソフトウェアモジュール構成を図２に示す。

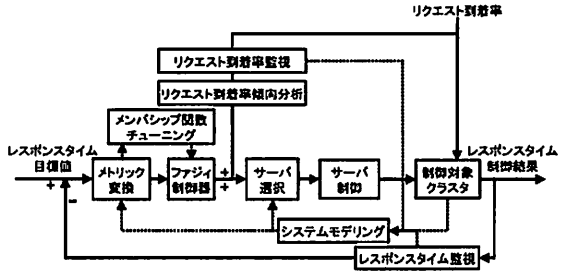

図１　ブロック線図

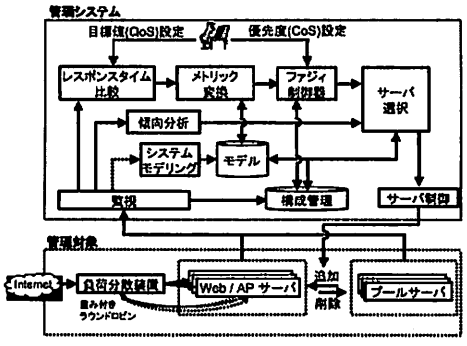

図２　モジュール構成

3.1.1. フィードバック制御系

フィードバック制御系ではレスポンスタイムを監視し、これを目標値に誘導するようにサーバ資源量を増減させる。この系は、レスポンスタイム比較、メトリック変換、ファジィ制御、サーバ選択で構成される。メトリック変換とサーバ選択は、性能モデルと構成管理に格納されるデータを参照する。

レスポンスタイム比較は、管理者がレスポンスタイム目標値を設定する為のインターフェースを持つ。この値は、サービスプロバイダとの契約により定まるサービスレベルを参考に設定される。フィードバック制御系は全体として、この絶対値で指定されるサービス品質 QoS を維持する。

メトリック変換は、各サーバのレスポンスタイムをクラスタ全体のリクエスト到着率に換算する。これにより、サービス品質が負荷量に変換され、性能の異なるサーバの処理能力と比較可能となる。換算は、サーバ性能とサービス品質・負荷の間のモデルを用いる。

ファジィ制御では、保守的なサーバ割当てと CoS の実現、及び安定化を行う。保守的なサーバ割当ては、サービスレベルを違反しにくい運用管理方針をファジィルールとして設定することで実現する。具体的には、品質劣化時のサーバ追加に積極的であり、品質良好時のサーバ削除に消極性であるという方針である。CoS は、残りプールサーバ台数に応じた資源割当量を、サービス毎にルール記述可能とすることで実現する。安定化については、ファジィ制御器や後述の整数計画問題を含む制御系の解析的な安定化条件を求めることが困難であるため、安定化指標が高まるようにメンバシップ関数をヒューリスティックに最適化する。

サーバ選択は、ファジィ制御器から出力されるフィードバック量が、どのプールサーバの組合せと対応するかという整数計画問題を計算する。ここでのフィードバック量は、レスポンスタイムを目標状態に誘導するために、クラスタ全体でどれ程のリクエスト到着率を調整すれば良いかを表す量である。プールサーバの処理能力は、どれ程のリクエスト到着率であれば、目標状態のレスポンスタイムとなるかを表す量である。

サーバ制御は、制御対象のサーバに Web アプリケーションファイルを展開し、このサーバをロードバランサのリクエスト割振り先に追加する制御などを行う。

3.1.2. フィードフォワード制御系

フィードフォワード制御系は、リクエスト到着率の傾向分析を行い、予測される変化分を相殺するようにサーバ資源量を調整する。フィードバック制御系ではレスポンス劣化後に資源追加が行われるが、このフィードフォワード制御系では品質劣化を伴わずに品質が維持される。この系は、傾向分析とサーバ選択、サーバ制御から構成される。

傾向分析では、サーバ制御で要する時間分だけ将来時点のリクエスト到着率を予測する[1]。サーバ選択では、そのリクエスト到着率と監視結果の差分を算出し、これと対応するプールサーバの組合せを算出する。

3.2. 管理対象負荷分散システム

ヘテロ環境への対応を特徴とする本手法では、制御対象負荷分散装置の重み付きラウンドロビン(WRR)を動的・半動的に動作させる。WRR とは、各サーバへのリクエスト転送割合に重みをつけ、サーバ負荷を均一化する方式である。動的な動作とは、負荷分散装置が常時サーバを監視し、その結果に基づき重みを変更する。半動的な動作とは、管理者がこの重みを変更する。

3.3. システムモデリング

システムモデリングは、レスポンスタイム$r_{ij}(t)$をリクエスト到着率$\lambda_{ij}(t)$およびサーバ性能$\mathbf{s}_j$と関係付ける。$\mathbf{s}_j$は、例えばCPUクロックやメインメモリ容量からなる(式(2))。i,jはサービスとサーバを識別する。

$$r_{ij}(t) = f_i(\lambda_{ij}(t), \mathbf{s}_j) = g_{ij}(\lambda_{ij}(t)) \quad (1)$$

$$\mathbf{s}_j = (s_j^{cpu}, s_j^{memory}, \cdots)^T \quad (2)$$

式(1)は、M/M/1/∞モデルを適用するとサーバjの性能$\mathbf{s}_j$に依存する$\mu_{ij}(t)$を用いて式(3)と表せる。この$\mu_{ij}(t)$は、$\mathbf{s}_j$に任意の写像ϕで得られる$\mathbf{s}_j'$の線形結合として線形重回帰分析などを用いてモデル化する（式(5),(6)）

$$r_{ij}(t) = (\mu_{ij} - \lambda_{ij}(t))^{-1} \quad (3)$$

$$\mathbf{s}_j' = \phi(\mathbf{s}_j) = (s_j^{cpu}, s_j^{cpu^2}, s_j^{memory}, \cdots)^T \quad (4)$$

$$\mu_{ij} = \boldsymbol{\beta}_i^T \mathbf{s}_j' \quad (5)$$

$$\boldsymbol{\beta}_i = (\beta_{i0}, \beta_{i1}, \ldots)^T \qquad (6)$$

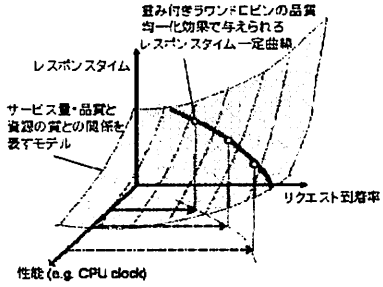

図3 システムモデルとWRRによる品質均一化効果

式(1)とWRRによるレスポンスタイム均一化効果とを模式的に図3で示す。時刻tのあるレスポンスタイムを$r'_i(t)$とすると、サーバ性能がs_jであるサーバがレスポンスタイム$r'_i(t)$の下で処理できるリクエスト到着率$\lambda_{ij}'(t)$は式(7)と見積もられ、M/M/1/∞モデルを想定すると式(8)となる。

$$\lambda'_{ij}(t) = g_{ij}^{-1}(r'_i(t)) \qquad (7)$$

$$\lambda'_{ij}(t) = \boldsymbol{\beta}_i^T \boldsymbol{\phi}(\mathbf{s}_j) - \frac{1}{r'_i(t)} \qquad (8)$$

3.4. メトリック変換

メトリック変換はシステムモデルを用いて、制御の目標であるレスポンスタイムを、サーバの処理能力とのマッチングを可能なリクエスト到着率に変換する。

サーバjに配備されたサービスiのレスポンスタイムを$r_{ij}(t)$、その目標値を$r_{ig}(t)$とする。$r_{ij}(t)$を$r_{ig}(t)$に誘導するために必要な偏差$e_i(t)$は、サーバ集合をJとして式(9)で表される。

$$e_i(t) = \sum_{j \in J} \left(\frac{r_{ij}(t) - r_{ig}(t)}{\partial r_{ij} / \partial \lambda_{ij}} \right) = \sum_{j \in J} (r_{ij}(t) - r_{ig}(t))(\boldsymbol{\beta}_i^T \boldsymbol{\phi}(\mathbf{s}_j) - \lambda_{ij})^2 \qquad (9)$$

3.5. ファジィ制御器

保守的なサーバ割当てと優先制御および安定化を実現するために、メンバシップ関数のチューニングが行われるファジィ制御器を導入した。この制御器の構成を述べた後に、各々の利用方法について述べる。

3.5.1. ファジィ制御器の構成

ファジィ制御器は、メトリック変換出力$e_i(t)$を受け取り、その時間変化成分$d_i(t)$と、プールサーバ台数$p(t)$およびサービスに割り当てられているサーバ台数$a_i(t)$とから、要求する資源量と対応する$u_i(t)$を出力する。

$$u_i(t) = h_i(e_i(t), d_i(t), p(t), a_i(t)) \qquad (10)$$

$$d_i(t) = e_i(t) - e_i(t - \Delta t) \qquad (11)$$

ファジィ制御器は、ファジィ化、ファジィルール、非ファジィ化で構成される（図4(a)）。ファジィ化は前記4入力変数値を、メンバシップ関数を用いて言語表現に変換する（図4(b)）。ファジィルールは言語表現化された状態と、要求資源量をシステム管理者が認識できるルールで表現したルールの組合せである。ルールの組合せの例を、図5に示す。非ファジィ化は、メンバシップ関数とmin-max合成重心法を用いて言語表現化された要求資源量を数値化する。

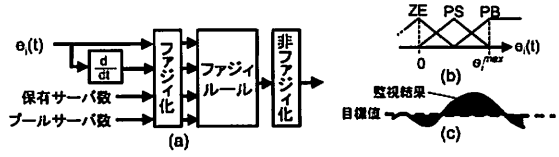

図4 ファジィ制御器の構成

3.5.2. 保守的なサーバ割当て

サービスレベルを違反しにくい運用管理方針をファジィルールとして設定する。この運用管理方針とは、品質劣化時のサーバ追加に積極的であり、品質良好時のサーバ削除に消極性であるという方針である。このファジィルールの例を図5および図6に示す。

3.5.3. 優先制御

サービス優先度は、前件部に目標値との差とその傾向だけでなく、プールサーバ台数と当該サービスに割当済のサーバ台数を設けることで実現する。

4変数前件部と1変数商件部は、割当方針と優先度とに分けられる。割当方針は、目標値との差と、その傾向に対して、どの程度の資源要求量を割当てるかを設定するルールセットである。優先度は、当該サービスの割当済サーバ台数と、プールサーバ台数に対して、どの割当方針を設定するかを表すルールセットである。

割当方針の例を図5と図6に示す。図5は、レスポンスタイムが目標値より少し悪くなりやすい謙虚な割当方針である。図6は、レスポンスタイムが目標値より少し良くなりやすい貪欲な割当方針である。

謙虚な割当方針 M		品質傾向: とても良好化	少し良好化	変化なし	少し悪化	とても悪化
目標値との比較 品質（レスポンスタイム）	とても良い	NB *1	NB	NS	NS	ZE
	少し良い	NB	NS	NS *2	ZE	ZE
	適度	NS	NS	ZE *3	ZE	PS
	少し悪い	NS	ZE	ZE *4	PS	PS
	とても悪い	ZE	ZE	PS	PS	PB

PB：多くの資源を要求する　NB：多くの資源を返却する
PS：少しの資源を要求する　NS：少しの資源を返却する
ZE：何もしない

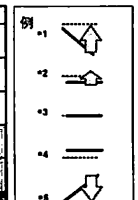

図5 謙虚な割当方針Mのルールセット

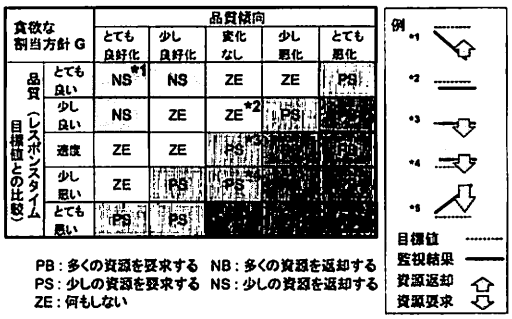

図6　貪欲な割当方針Gのルールセット

優先度の例を図7と図8に示す。図7は、プールサーバを優先的に確保しやすい高優先度サービスのルールセットである。図8は、プールサーバを解放しやすい低優先度サービスのルールセットである。

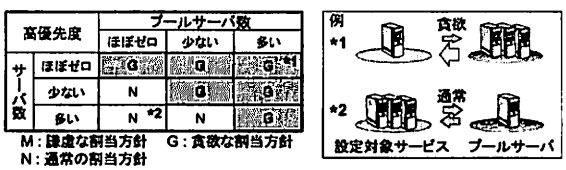

図7　高優先度サービスの割当方針テーブル

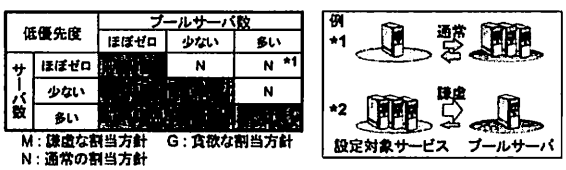

図8　低優先度サービスの割当方針テーブル

3.5.4. 安定化

サーバ選択において非線形な制御方法を含むため、ヒューリスティックに安定化を図る必要があり、そのためにメンバシップ関数のチューニングが行われるファジィ制御器を用いた。

チューニングは、不安定性を最小になるようにメンバシップ関数の定義域の最大値を変更する。図4(b)での最大値はe_i^{max}である。不安定性は、レスポンスタイムの監視結果と目標値の差の絶対値を時間積分した値（図4(c)の斜線部）で評価する。チューニングのアルゴリズムとしては、焼き鈍し法などを用いる。

3.6. サーバ選択

サーバ選択は、制御対象サービスの品質を目標値まで誘導するため、あるいは予測リクエスト到着率変化に適応するため、サーバ増減命令$k_i(t)$を出力する。この$k_i(t)$は、サービスiにサーバjを割当てるか否かの命令$k_{ij}(t) \in \{1,-1,0\}$を用い、$k_i(t)=(k_{i0}(t),k_{i1}(t),\ldots,k_{ij}(t))^T$で定義される。$k_{ij}(t)$の値1は追加命令を、値-1は削除命令を意味し、値0は制御を行わないことを意味する。

性能が異なるサーバを、要求される資源量と比較可能とするために、前述の性能モデルとWRRによるレスポンスタイム均一化効果から得られる関係式を用いる。各サーバがある一定レスポンスタイム$r'_i(t)$の元で処理可能なリクエスト到着率と等価な容量$c_{ij}(t)$は、式(12)により求める。ここで$r'_i(t)$は、フィードバック制御では目標値$r_{ig}(t)$であり、フィードフォワード制御では各サーバ監視結果$r_{ij}(t)$の平均値とする。

$$c_{ij}(t) = g_{ij}^{-1}(r_i'(t))$$
$$r_i'(t) = \begin{cases} r_{ig}(t) \\ \sum_{j\in J} r_{ij}(t)\lambda_{ij}(t) \Big/ \sum_{j\in J} \lambda_{ij}(t) \end{cases} \quad (12)$$

ファジィ制御器あるいは傾向分析から受ける資源要求量$u_s(t)$と対応する、各サーバの容量$c_i(t)$の組合せは、式(9)のように整数計画問題となる

$$\min \quad \sum_{j\in J} k_{ij}(t)c_{ij}(t) - u_i(t)$$
$$s.t. \sum_{j\in J} k_{ij}(t)c_{ij}(t) - u_i(t) > 0 \quad (13)$$

本システムでは、この$k_{ij}(t)$を数分程度の制御ループの中で決定する必要がある。そのため、計算時間の短いGreedyアルゴリズムを用いる。

3.7. リクエスト到着率傾向分析

リクエスト到着率の傾向分析は、ARIMAモデル[9]を用いる。AR成分、MA成分、階差成分の各次数は、複数次数についてモデル生成した後に、対数尤度をもとに選択し、サーバ制御に要する時間τ後のリクエスト到着率$\lambda_i'(t+\tau)$を得る。サーバ選択に入力する$u_i(t)$は、予測増加にどれだけ反応するかを表すパラメータκ_iを用いて式(14)で与えられる。

$$u_i(t) = \kappa_i(\lambda_i'(t+\tau) - \lambda_i(t)) \quad (14)$$

4. 実験

4.1. システム構成

管理対象システムは負荷分散装置、Web/APサーバ、DBサーバからなる。負荷分散装置はAlteon AD3を用い、Web/APサーバはTomcat5.5とJBoss4.0を用いた。このWeb/APサーバ7台を自律制御の対象とした。性能は6台をCPU 3.0GHzのサーバとし、1台をCPU 1.7GHzのサーバとした。DBサーバとしては、MySQL5.0を用いた。J2EEアプリケーションには、Duke's Bankを用いた。

管理システムは主にJavaで構築した。Java VMはSun jdk5.0、各モジュールはOSGi(Open Services Gateway Initiative)バンドルとし、OSはRedHat Linux 9とした。

4.2. 実験結果

4.2.1. システムモデル

本試作のサービスをサーバに配備して得られたリクエスト到着率に対するレスポンスタイムと、これをオフラインで作成したモデルを図 9 と式(15)に示す。

$$\mu_{ij} = 3CPU_j^2 + 5CPU_j + 3 \qquad (15)$$

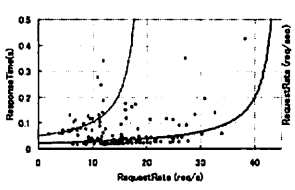

図 9 管理対象サービスのモデル

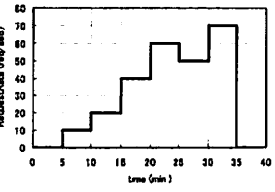

図 10 クライアントから発生させた負荷

4.2.2. サービスレベル管理機能評価

本運用管理システムの基本機能であるサービスレベル管理機能を評価するために、管理対象サーバに対して変動負荷を与え（図 10）、レスポンスタイムとスループットを評価した。評価対象として、レスポンスタイムの閾値監視でサーバ増減を行う方式についても同様の負荷を与えた。

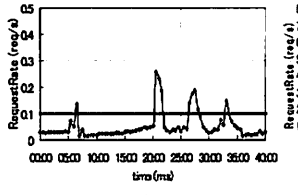

図 11 本制御方式によるレスポンスタイム

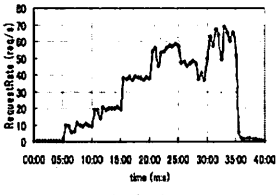

図 12 本制御方式によるリクエスト到着率

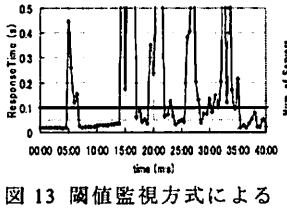

図 13 閾値監視方式によるレスポンスタイム

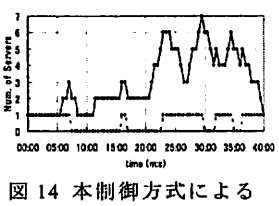

図 14 本制御方式によるサーバ台数

本制御方式により得られたレスポンスタイムとリクエスト到着率を図 11 と図 12 に示す。図中には、0.1 秒の誘導目標レスポンスタイムも示されている。閾値監視方式により得られたレスポンスタイムは図 13 となる。追加閾値を 0.15 秒とし、削除閾値を 0.05 秒とした。本制御方式により制御されたサーバ台数の変化を図 14 に示す。図 14 では、全サーバ台数を実線、低性能サーバ台数を破線で示している。

閾値監視方式では、レスポンスタイムの劣化は最大 7.2 秒、制御目標を超えた期間は全体の 36%であった。一方、本方式では劣化は最大 0.25 秒であり、制御目標を超えた期間は全体の 13%であった。

なお、この評価ではメンバシップ関数はチューニングせずに固定的に与えている。

5. まとめ

性能の異なるサーバが存在するテロジニアス環境においても、負荷変動の大きなサービス品質を維持し、優先制御機能を有するサーバ容量計画法を提案した。

ヘテロジニアス環境にて基本機能であるサービスレベル管理機能を評価した。閾値監視による自律制御ではレスポンスタイムが 7.2 秒にまで劣化し、目標値を超える期間が 36%存在する変動的負荷に対しても、同 0.25 秒と 13%と良好なサービスレベル管理機能を有することを確認した。今後優先制御機能の評価を行う。

謝辞

本研究は、総務省からの委託研究の成果である。

文　献

[1] E. Lassettre, D. W. Coleman, Y. Diao, S. Froehlich, J. L. Hellerstein, L. Hsiung T. Mummert, M. Raghavachari, G. Parker, L. Russell, M. Surendra, V. Tseng, N. Wadia, and P. Ye, "Dynamic Surge Protection: An Approach to Handling Unexpected Workload Surges with Resource Actions that Have Lead Times," 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM2003), Heidelberg, Germany, October 2003.

[2] B. Urgaonkar, P. Shenoy, A. Chandray, and P. Goyalz, "Agile, Dynamic Provisioning of Multitier Internet Applications," 2nd IEEE International Conference on Autonomic Computing (ICAC 2005), Seattle, USA, June 2005.

[3] J. L. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, "Feedback Control of Computing Systems," Wiley InterScience, 2004.

[4] X. Liu, X. Zhu, S. Singhal, and M. Arlitt, "Adaptive Entitlement Control of Resource Containers on Shared Servers," 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), Nice, France, May 2005.

[5] Y. Diao, N Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics With Application to the Apache Web Server," 8th IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, April 2002.

[6] Y. Diao, F. Eskesen, S. Froehlich, J. L. Hellerstein, Alexander Keller, Lisa F. Spainhower, M. Surendra, "Generic On-Line Discoverty of Quantitative Models for Service," 8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), Colorado Springs, USA, March 2003.

[7] G. E. Box and G. M. Jenkins, "Time Series Analysis: Forecasting and Control," Holden-Day, 1976.

[8] Y. Diao, J. L. Hellerstein, S. Parekh, "Using fuzzy control to maximize profits in service level management," IBM System Journal, Volume 41, No 3, 2002.