待ち行列理論に基づく複数ドメイン間の自律的広域負荷分散

山根敏志*， 美園和久*，木村順子*，藤本洋平*

概 要

本論文では複数ドメイン間で、個々のドメインの自律的な判断によって広域負荷分散を行う方式を提案する。この方式は待ち行列理論を利用するものであり、平均応答時間やリクエスト到着率などの測定可能・操作可能な量のみを用いて自律的に負荷分散を行うことができる。さらに、実際の Web システムを用いた実証実験を行った結果、効果的なパフォーマンス・ゲインを得た。

An autonomic wide load balancing over multiple domains based on queuing theory

Toshiyuki Yamane*, Kazuhisa Misono*, Junko Kimura*, Yohhei Fujimoto*

Abstract

We present a method of autonoic load balancing over multiple domains. The method utilizes queuing theory and unables us to balance load autonomically by only observable or controllable quantities such as mean response time and request arrival rates. Furthermore, we have applied our method to demonstrative experiments on real Web systems and achieved effective performance gain.

1 はじめに

ブロードバンド・プラットフォームを基盤としたインターネットの情報サービスが普及するにつれて、サービス利用者数は指数関数的に増加し、特定のサーバーやコンテンツへのアクセス集中により、サーバー・レスポンスの低下やネットワークの輻輳といった問題が顕在化している。さらにトラフィック量の単純な増加以上に問題になるのが、その変動の激しさである。例えば、関心の高い新製品の発売、マスコミへの URL 紹介などの理由により、Web サイトに集まるトラフィックは大幅に変動する。しかし、通常システムはそれぞれの ICT(Information and Communications Technology) 資源が予想される最大負荷に対応できるように構築されているため、ICT 資源の平均使用率はきわめて小さく、ICT 資源の有効利用がなされていない状況である。

そこで、ICT 資源の有効活用のための技術として従来のシングル・ドメイン内の負荷分散ではなく、物理的にも離れた複数ドメイン間での広域負荷分散による ICT 資源の統合的利用が期待される [3],[4],[5]。これは現行のシングル・ドメイン内での負荷分散技術を発展させたもので、自ドメイン内での ICT 資源不足が発生した場合に複数の他ドメインに跨り負荷分散を実施することで、各ドメインが余剰に所有している ICT 資源を活用しようというものである。

一方、近年では Web サービスと呼ばれるネットワーク分散処理技術が登場し、SOAP(Simple Object Access Protocol) などのリモート通信

*株式会社日本 IBM(IBM Japan, Ltd.)

機能に基づいて XML 形式のメッセージを交換し、インターネットを通じて出自の異なるアプリケーション・システムを柔軟に連携させることが可能となった。しかし、この Web サービスの導入により、拡大を続ける分散ネットワーク・システムの世界におけるプロバイダー同士の関係がさらに複雑化し、SLA(service level agreement) の遵守はむしろ困難なものとなった。現代の Web システムはこのように複雑化した異機種混合の分散システムという形で緩やかに相互接続されているため、複数ドメインに跨る ICT 資源の有効利用は一層困難な状況である。

このように複雑化する一方の IT 環境を救うべく現実的になりつつあるのが、自身の状態に応じて動作し、あたかも生命体のようにシステム自身が自律的に自己管理する「オートノミック・コンピューティング」のビジョンである [1]。オートノミック・コンピューティングとは、これまでのような複雑さを増すばかりの自動化ではなく、抽象度のレベルを上げ、より高いところから全体を見渡すことで IT システムの維持・運用にかかるコストや複雑さを削減させることを目的としたものである。オートノミック・コンピューティング・アーキテクチャーに基づく管理対象エレメントの制御ループは MAPE ループと呼ばれ、次の４つの部分、**１．監視 (Monitor)**(管理対象から収集された情報の集約および報告), **２．分析 (Analyze)**(IT 環境のモデル化 (抽象化) を行う), **３．計画 (Plan)**(分析結果から、目標達成に必要なアクションを策定), **４．実行 (Execute)**(計画を解釈し、計画の実行指令を制御) に分かれている。この MAPE ループを実装したコンポーネントはオートノミック・マネージャーと呼ばれる。

本研究の目標は突発的に負荷が変動するインターネット環境下において、オートノミック・コンピューティングの枠組みにそって複数ドメインに跨る広域負荷分散を自律的に行うことによって、限られた ICT 資源をより有効に使用する方策を提示することである。(図１参照。) 具体的には、複数の優先順位の異なるアプリケーションが存在する状況で、最優先アプリケーションの応答時間が悪化してきたとき、低優先度のアプリケーションへのリクエストを余剰の ICT 資源をもつ他ドメインに転送することで最優先アプリケーションの応答時間 SLA(Service Level Agreement) を維持する、というものである。この際、待ち行列理論の平均値解析に基づき、平均応答時間やリクエスト到着率などの測定可能・操作可能な量のみを用いて自律的に負荷分散を行う。これにより、予期せぬ急激なアクセスによって自ドメイン内の ICT 資源だけでは応答時間 SLA が満たされない場合においても、人間が介在することなくシステムが隣接するドメインの ICT 資源を利用することで、応答時間 SLA を遵守できる。さらに、この方式を実際の Web システムに適用した実証実験についても報告する。

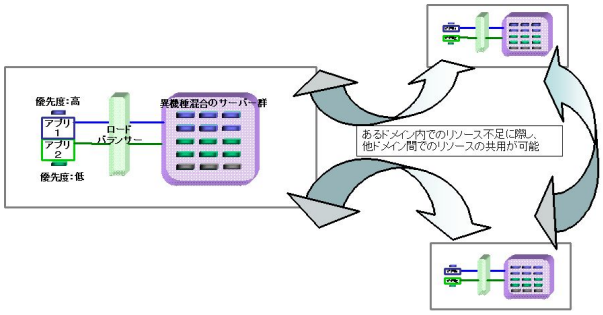

図１．複数ドメイン間の自律的負荷分散

2　待ち行列理論による負荷分散

2.1　待ち行列モデル

待ち行列理論はヒトやモノがある処理プロセスへ到着し、サービスの順番を待ち、サービスを受けて立ち去る状況をモデル化しており、ヒトやモノが到着する間隔、サーバーがサービスを提供する時間に確率的な分布を考える。待ち行列モデルは一般に次に示すような要素からなる。A: 客の到着時間間隔分布 (M ＝指数分布, etc)
S: サービス時間の分布
m: サーバー数
B: システムの容量 (システム内に入れる客の最大数), バッファサイズ (有限値，∞)
K: システムに来る客の数 (有限値，∞)

SD: サービスの方式, Service Discipline (First-Come-First-Served(FCFS)), Round Robin, etc)
このとき、待ち行列は A/S/m/B/K/SD のように表記される (Kendall の記法)。

待ち行列理論を負荷分散へ応用する研究は多数存在する。たとえば、[7] においては、複数の計算資源ノードがネットワークを介して結合したモデルを考え、ノード全体での平均応答時間を最適化するような個々のノードへの負荷の割り振り方を求める問題を取り扱っている。従来の負荷分散は、この研究も含めて全体のパフォーマンスを一元管理できるという仮定のもと、あらかじめ最適な負荷分散割合を求めておく静的な手法であるといえる。しかし、Web サービスのような異質なドメイン同士がネットワークを介して緩やかに結合し、突発的なトラフィックの増減に常にさらされるシステムにおける負荷分散では静的な全体最適化のモデルは適さない。また、Web サービスにおいてはリクエストの処理が必然的に複数のシステムにまたがるため、これまでの特定のシステム内の詳細なパフォーマンス情報に基づいた負荷分散では適応できない。

個々のドメインがそれぞれの ICT 資源の動作状況に応じて自律的に、かつ、実際に測定可能な応答時間やリクエストの到着率を元に動的に負荷分散を行うというモデルが望まれる。実際のドメインのパフォーマンス状況を考慮しつつ、オートノミック・コンピューティングの枠組みに沿って自律的に複数ドメイン間での負荷分散を行うためには、「分析」「計画」を行うコンポーネントとしてのパフォーマンス・モデルおよび MAPE ループに沿って動作するオートノミック・マネージャーを定義することが必要である。そこで、本研究ではオートノミック・マネージャーが参照するモデルとして待ち行列モデルの中でも最も基本的な M/M/1 モデルを用いる [6],[7],[2]。具体的には図 2 のように単一ドメイン内の各アプリケーションごとに M/M/1 待ち行列モデルを考え、このモデルに基づきリクエスト到着率を操作することで各アプリケーションの平均応答時間をコントロールするというアプローチを取る。M/M/1 は窓口が一つ、ユーザーの到着が独立で同一の生起率 λ の Poisson 分布、窓口におけるサービス時間も独立で同一の指数分布に従うというモデルである。また $B = K = \infty$, SD=FCFS である。窓口で単位時間あたりに処理できるユーザーの数を μ_c とすると、平均のサービス時間は $1/\mu_c$ となる。

待ち行列理論において、トラフィックや応答時間、スループットなどの性能指標の平均値の間の関係式を与える理論を平均値解析という。M/M/1 モデルの場合の平均値解析により、応答時間の平均値 R_c は

$$R_c = \frac{1}{\mu_c - \lambda_c/m_{\mathrm{total}}}$$

となることが知られている。ここで、添え字の c は各アプリケーションをあらわす。また、m_{total} はシステム全体のリソースを表しているが、これは仮想的な量であって必ずしも実際のシステムの物理的なリソースと厳密に対応している必要はなく、次節に述べる事前の負荷テストによって推定すべき量である。

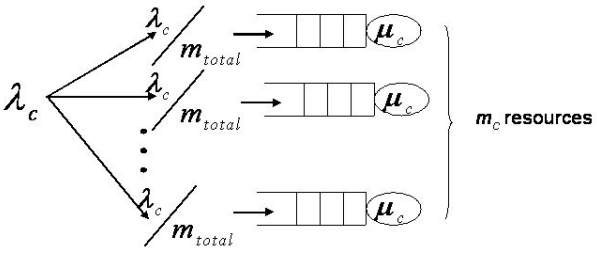

図 2．M/M/1 モデルによるコンピューター・システムのモデリング

2.2 モデリング・パラメーターの推定

この節では事前の負荷テストの結果から、モデルのパラメーターを推定する手法について述べる。各アプリケーションの M/M/1 システムによるパフォーマンス・モデル作成のために、各アプリケーションのリクエストの到着率を変えて、負荷テストを行う。理論曲線を実測値に対してフィッティングさせ、以下のモデルの未

知数 μ と m_{total} を推定する。

$$R = \frac{1}{\mu - \lambda/m_{\text{total}}}$$

このままのモデルの形では推定しにくいので、次のような変換を行って線形モデル (単回帰モデル) に変換する：

$$\begin{aligned} y &= 1/R \\ a &= 1/m_{\text{total}} \\ y &= \mu + a\lambda \end{aligned}$$

こうしておいて、単回帰モデルを当てはめることで未知数 μ と a を推定する。ただしパラメーター a はすべてのアプリケーションのモデルに対して共通のパラメーターである。以下、これらの未知数を推定する方法を２つのアプリケーションが存在する場合について述べる。アプリケーションが３つ以上の場合も同様である。実測データ $(x_{1i}, y_i), (x_{2i}, z_i); i = 1, \cdots, n$ に対して次の平均二乗誤差

$$S(a, \mu_1, \mu_2) = \sum_{i=1}^{n} (y_i - ax_{1i} - \mu_1)^2 + (y_i - ax_{2i} - \mu_2)^2$$

を考える。３つのパラメーター a, μ_1, μ_2 に関して S を偏微分し、次の正規方程式を得る：

$$\begin{pmatrix} D & A_1 & A_2 \\ A_1 & n & 0 \\ A_2 & 0 & n \end{pmatrix} \begin{pmatrix} a \\ \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} E + F \\ B \\ C \end{pmatrix}.$$

ここで、
$A_1 = \sum_{i=1}^{n} x_{1i}, A_2 = \sum_{i=1}^{n} x_{2i}, B = \sum_{i=1}^{n} y_i, C = \sum_{i=1}^{n} z_i,$
$D = \sum_{i=1}^{n} x_{1i}^2 + x_{2i}^2, E = \sum_{i=1}^{n} x_{1i} y_i, F = \sum_{i=1}^{n} x_{2i} z_i$
とおいた。

図３に今回実証実験で用いたオンライン証券取引アプリケーションとオンライン・ショッピングアプリケーション (4 節参照) に対する測定結果を示す。なお、平均応答時間の単位は秒であり、測定時間は各点に対して 5 分である。単回帰モデルを当てはめた結果次の結果を得た。
$y = 1.878 - 0.057x$, (オンライン証券取引)
$y = 3.278 - 0.057x$, (オンライン・ショッピング)

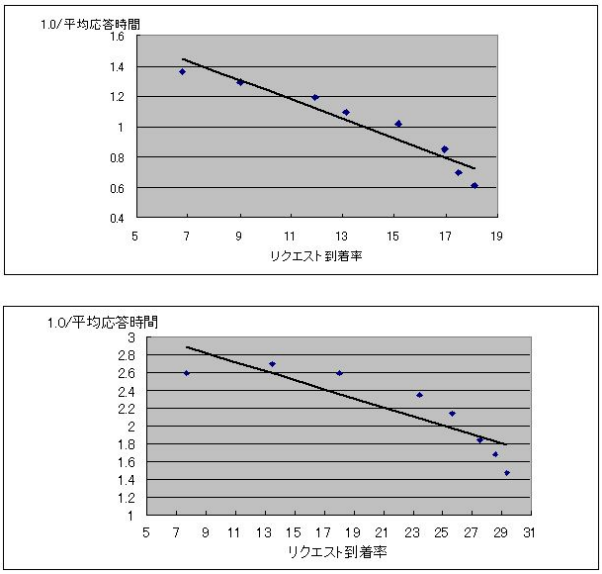

図３．モデルパラメーターの推定。上：オンライン証券取引アプリケーション、下：オンライン・ショッピングアプリケーション

従って、モデルのパラメーターとして、

$$\mu_{\text{A}} = 1.878, \quad \mu_{\text{B}} = 3.278, \quad m_{\text{total}} = 17.54$$

が得られた。ここで、$\mu_{\text{A}}, \mu_{\text{B}}$ はそれぞれ、オンライン証券取引アプリケーションとオンライン・ショッピングアプリケーションのリクエスト処理率である。また、それぞれのモデルの平均二乗誤差および goodness-of-fit 値 (R^2 値) は、表１のとおりである。

表１．M/M/1 モデルの評価

	平均二乗誤差	R^2
オンライン証券取引	0.047	0.9104
オンライン・ショッピング	0.367	0.755

2.3 負荷分散のためのアルゴリズム

待ち行列理論における平均値解析は通常、システムに対する負荷を入力として与え、そのもとでのシステムのパフォーマンスを見積もるために用いられる。本論文では逆に、所望のパフォーマンスを得るためには負荷をどの程度にコントロールしなければならないかを見積もる手段として平均値解析を捉え、負荷分散のためのアルゴリズムを導くことにする。以下で具体的に、アプリケーション数が２の場合に優先順

位の高いアプリケーション c が SLA 違反を起こした場合について述べる。アプリケーション数が増えても同様である。これはオートノミック・マネージャーの制御ループの計画と実行に相当する。

1. 現在のリクエスト到着率 λ_c^{now} より必要なサーバー・リソース m_c^{new} を次の式より、応答時間の平均が $r \times R_c^{SLA}$ となるようにを決定する。

$$m_c^{new} = \frac{\lambda_c^{now}}{\mu_c + 1/r \times R_c^{SLA}}$$

ここで r は安全率 (マージン、0 以上 1 未満の値)、R_c^{SLA} はアプリケーション c の平均応答時間 SLA を表す。マージンとしては SLA 違反状態では 0.8、SLA 遵守状態では 0.5 という値を採用した。

2. ドメイン内では各アプリケーションのリクエスト到着率に比例したサーバー・リソースが自動的に割り当たるものと仮定する。そして、アプリケーション c に m_c^{new} のサーバー・リソースが割り当てられるような、優先順位の低いアプリケーション d へのリクエスト到着率を次で計算する。

$$\lambda_d^{new} = \frac{m_d^{new}}{m_c^{new}} \lambda_d^{now}, \quad m_d^{new} = m_{\text{total}} - m_c^{new}. \tag{1}$$

3. 次の式で与えられるアプリケーション d の過剰分のリクエストを他ドメインに転送する。(この値が負ならば負荷分散割合を変更して、自ドメインへのリクエストの割り振りを増やす)

$$\Delta\lambda_d = \lambda_d^{now} - \lambda_d^{new}.$$

4. 転送先ドメインの決定
他ドメインの M/M/1 待ち行列モデルにもとづいて、アプリケーション d への転送量が変わったときのアプリケーション c の平均応答時間の変化を推定し、他ドメインにおいて最優先アプリケーションの SLA 違反が起きないことを確認した上で、アプリケーション c に最も影響がないドメインを選択する。

5. 計算結果の吟味
各アプリケーションのリクエスト到着率や平均応答時間は、時々刻々ランダムに変動する量である。たとえ同じ量の負荷であったとしても、その応答時間はモデルが想定するものとは大きく食い違うということがありうる。そこで、アルゴリズムがランダムに変動する量をリアルタイムに扱っても、負荷分散システムとして安定して動作させるためには、上記の計算の結果をそのまま実行に移す前にまず結果を吟味する必要がある。今回は、以下のようなケースにおいては負荷分散の指令を出さないこととした。
• SLA 違反が検知されているにもかかわらず、計算の結果得られた必要なリソース量が現在のリソース量を下回った場合
• SLA 違反が検知されていないにもかかわらず、計算の結果得られた必要なリソース量が現在のリソース量を上回った場合

6. 転送先ドメインにおいても負荷が上昇して最優先アプリケーションの SLA 違反が起こった場合には、同様の処理が行われて場合によっては SLA を満たすために更に隣接のドメインに転送することも行う。

3 自律的広域負荷分散システムの概要

3.1 システム構成

今回構築した負荷分散システムの各コンポーネントの機能は以下のとおりである。

1. 統括監視システム
統括監視システムは各ドメインに 1 つ配置される。ドメイン内のアプリケーションに関する情報を一括管理する。ドメイン・エージェントから取得した自ドメインのパフォーマンス情報を RMI/IIOP (CORBA) を用いたリモート通信によっ

てサービスとして他ドメインへ公開する役割を持つ。また、自ドメインのパフォーマンス情報に基づいて負荷分散の割合を決定する。統括監視システムは以下のコンポーネントから構成される。

(a) モニタリング・マネージャー
自ドメインおよび他ドメインのパフォーマンス情報（モニタリング情報）を収集し、待ち行列モデルへ提供する。

(b) 待ち行列モデル
最優先アプリケーションの応答時間を監視し、パフォーマンス情報に基づいて他ドメインへのリクエストの負荷分散割合を計算する。

(c) コントローラー
ドメイン・エージェントを経由して、ドメインでのパフォーマンス情報を取得する。統括監視システムから指示を受け、ドメイン・エージェントへ新しい負荷分散割り当てを指示する。

2. ドメイン・エージェント
自ドメイン内のパフォーマンス情報を一括収集し、集計した情報を統括監視システムに報告する。統括監視システムからの負荷分散割合の変更要求があった際、自ドメインのロードバランサーに対してドメイン間の負荷分散割合の命令を発行する。

3. モニタリング・エージェント
各サーバーに配置され、Web サーバーの出力するログから各アプリケーションの一定時間あたりのリクエスト到着率/スループット、および平均応答時間のモニタリングを行う。パフォーマンス情報は、一定サイズのバッファに First-In-First-Out 方式で記憶され、ドメイン・エージェントからの要求に応じて返す。

複数ドメインにおける各コンポーネントの配置を図４に示す。

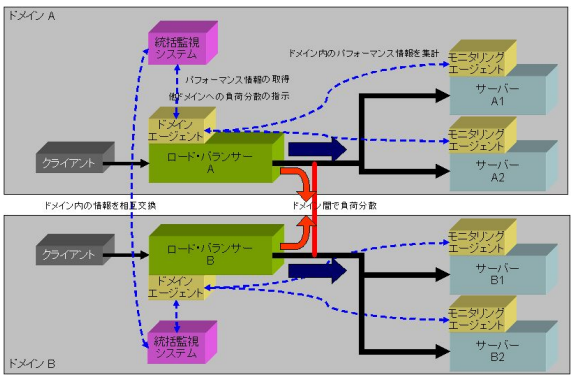

図４．自律的負荷分散システムの全体像

統括監視システムは、一定間隔ごとにモニタリング対象の各アプリケーションのパフォーマンス情報（リクエスト到着率・応答時間）を取得している。これらを一定期間分だけ内部のデータバッファに保持し、その間の平均応答時間を用いて SLA を評価する。今回はモニタリング・マネージャーのモニタリング情報の履歴設定はモニタリング間隔＝２秒、データバッファのサイズ＝ 20 とした。また、待ち行列モデルの監視インターバルは 10 秒に設定した。システム全体の SLA 違反に対する反応処理時間は、当然これらの値に伴って変化する。これらの値を大きくした場合、パフォーマンス情報の統計的変動は小さくなり、瞬間的なスパイク状のリクエスト増に対する過敏な反応は抑制される。しかし、過度に大きな値を指定すると SLA 違反の検知が遅れるため調整が必要である。一般的には、これらのモニタリングに関する設定値は負荷分散割合の変化のオーバーヘッドに依存して決定すべきである。なぜなら、通常 ICT 資源の動的な配置変更には大きなオーバーヘッドが発生するため、これらを短期間に増減させるとその操作自体がシステムに対して負荷をかけてしまうためである。しかし、本研究の統括監視システムでは負荷分散をロード・バランサーのドメイン間の負荷配分調整のみで実現しているため、比較的負荷分散割合の変更のオーバーヘッドは小さく、モニタリングに関する設定値も小さくすることができるため反応処理時間を短くすることができる。

3.2 複数ドメイン上への展開

本手法では全ドメインのパフォーマンス情報をそれぞれのドメインが収集しているため、ドメイン数が N である場合に、$O(N^2)$ の通信が発生してしまう。後述のネットワークに対する通信量の測定より、モニタリング間隔＝2秒、履歴数 (データバッファサイズ) ＝ 20 にて、17kbps (片方向通信) の負荷がかかることがわかっており、この値から 100Mbps の帯域が飽和するドメイン数を計算すると約 70 ドメインが本システムにおけるドメインの最大数となる。また、実際には統括監視システムに使用するネットワークは、ユーザーからにリクエストを受け付ける負荷分散装置も使用しているため、現実のドメイン数は 10 ドメイン程度が上限となり、複数ドメインを利用してのスケーラビリティの拡張は制限されることとなる。これらを防ぐための方法として、ドメインのグループ化を行い、同じグループ内のみで ICT 資源の共有、すなわちリクエストの転送を行う方法が考えられる。また図５にあるように、これらのグループを多段につなげることにより、ドメインやグループの階層化を行うことによってもこれらの問題が解決可能である。

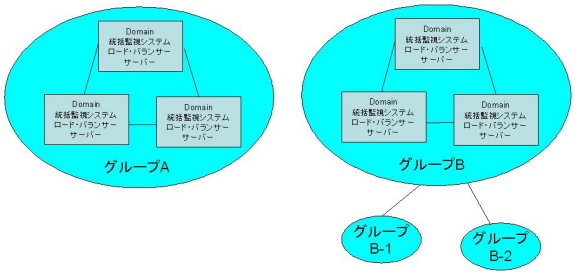

図５．ドメインのグループ化

4 実証実験

4.1 実証実験のシステム構成

本実証実験では実験環境として Linux, Windows, AIX(IBM Unix) の異機種システムが混在した２つのドメイン (ドメインＡおよびドメインＢ) で構成された Web システムを構築し、この実験環境に対してピーク性のある Web アプリケーションを２つのドメインで同時に実行させることにより、以下の観点から負荷分散技術の有効性の確認を行う。

・各ドメイン内の異機種サーバー資源の動作状況のモニタリング及び自律制御が有効に行われること。

・事前設定された SLA 応答時間を遵守した状態で各ドメイン内の異機種サーバー資源の有効利用が図れること。

各ドメイン内のロード・バランサーは、ドメイン内の負荷分散を実行する「シングルドメイン用」と、複数ドメイン間の負荷分散を実行する「マルチドメイン用」の２段構成とする。また、各ドメインの Web システムに対しユーザーアクセスによる負荷を生成する「負荷シミュレーター」を配置し、それらを共通の「負荷コントローラー」にて操作する。そして、ドメインＡ、ドメインＢに対し、研究開発を行った各コンポーネントを図６のように配置する。

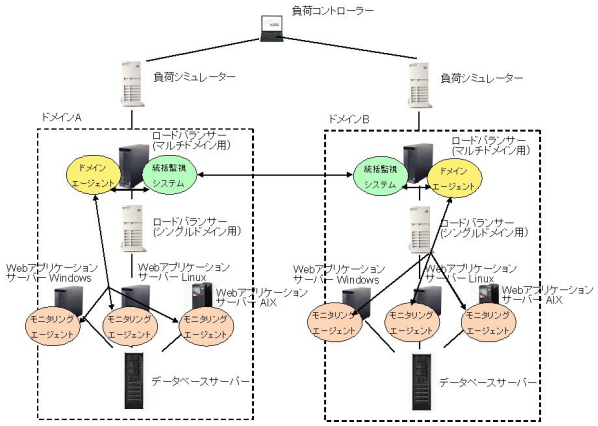

図６．ドメインＡ、Ｂにおける各コンポーネントの配置

実証実験用に、高優先度と低優先度の２種類の典型的なユーザー・アプリケーションを選定する。優先度が高いアプリケーションとしてオンライン証券取引 (以下アプリ A と呼ぶ) を選定し、株の売買シナリオを定義する。また、優先度が低いアプリケーションとしてオンライン・ショッピング (以下アプリ B と呼ぶ) を選定し、

注文照会シナリオを定義する。これらのシナリオを負荷シミュレーターに実行させ、両ドメインに対して一定のユーザー数が同時に Web システムを利用しているという状況を生成する。また、この実証実験においては、平均応答時間 SLA としては事前に定義されたユーザー・シナリオにおいてログインからログアウトまでに要した平均応答時間に関する規約であると定義し、アプリ A の目標平均応答時間 SLA を 2 秒と設定した。

4.2　システム動作の概略

図 7 はドメイン A における統括監視システムが取得したアプリ A、アプリ B のそれぞれのリクエスト到着率、応答時間、およびドメイン B に対する負荷転送率（負荷分散割合）の時間変化を表した図である。

テスト開始時には、アプリ A のみ負荷（ユーザー数：10）を与えている。この状態から、アプリ A への負荷はそのままでアプリ B への負荷を増大（0 ユーザーから 20 ユーザー）させ始めた。その結果、アプリ A の応答時間は SLA の目標値である 2 秒を超え、SLA 違反状況が発生する。このとき、統括監視システムが SLA 違反を検知し、SLA を回復させるため負荷分散割合の計算を行う。統括監視システムでは、瞬間的なスパイク状の到着率上昇に対して、システムが過敏に反応しすぎることを抑制するために、過去のある時間区間にわたるモニタリング情報の平均値を基に負荷分散割合を計算している。従って、モニタリング・エージェントから取得された最新の値と、統括監視システムの計算に使用されるモニタリング情報の統計値との値は食い違がある。その結果、統括監視システムによるアプリ B の他ドメインへの転送後も、アプリ A の SLA 違反は回復しないため、ロード・バランサーへの割り当て変更指示が繰り返される。こうして徐々に他ドメインへの負荷分散の割合は増加していき、最終的に約 65%の負荷分散割合で他ドメインへの転送は一定となる。この結果、アプリ A の応答時間は SLA 目標値である 2 秒に回復した。SLA の回復後も、一定の時間定常負荷をかけ続けるがアプリ A の SLA 目標値である 2 秒は保持されているため、負荷分散割合の計算は行われず、負荷分散割合は一定した状態を保ち続けている。

次にアプリ B の負荷を 20 ユーザーから 1 ユーザーに減少させる。その結果、統括監視システムは他ドメインへ割り振ったアプリ B を自ドメインに戻してよいかどうかを判断するために、再び待ち行列モデルにより計算を行う。この結果十分な安全を判断された場合にのみ、他ドメインへ割り振ったリクエストの割り振り戻しを実行する。上記の判断が統括監視システムで行われた結果、アプリ B の他ドメインへの転送は減少していく。そして最終的に他ドメインへの転送量である負荷分散割合は、実験開始時と同様 0 に戻る。このとき、事前にアプリ A の SLA に違反しないかをチェックしながら転送率を変化させているため、アプリ A は SLA 目標値の 2 秒を超えることはなく、SLA 遵守の状態が保持される。

また、優先度低アプリケーションの負荷量を一定（ユーザー数：10）にし、優先度高アプリケーションの負荷量を増減する（ユーザー数：１→１０→１）場合も同様の動作で SLA 基準を満たすことを確認した。

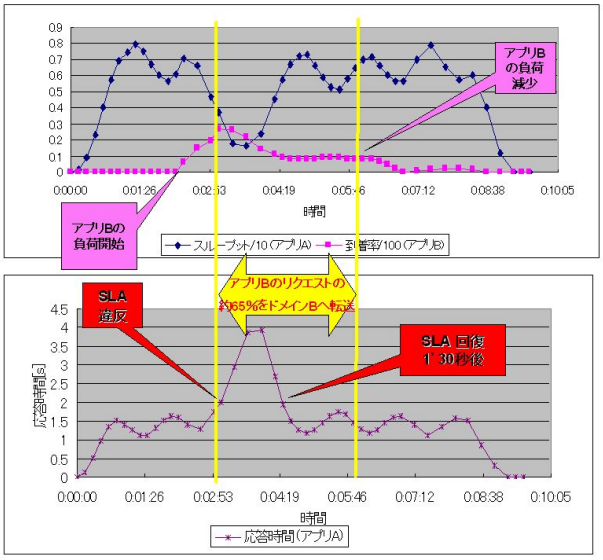

図 7．全測定データの時間変移

4.3 自律的広域負荷分散システムの評価

本研究開発の有効性検証の指標として次のネットワークへの負荷、オーバーヘッド率およびゲインを用いる。

ネットワークへの負荷

統括監視システムは、SLA 違反時における SLA 回復のための新たな負荷分散割合の計算過程で、他ドメインのパフォーマンス情報を必要とする。通常複数のドメインはネットワーク的に離れた位置に存在するから、ドメイン間の通信によるネットワーク負荷の影響を考慮する必要がある。現時点での統括監視システムの実装では、他ドメインのパフォーマンス情報の取得の際、他ドメイン上に蓄積された全データを取得している。よって、パフォーマンス情報の履歴数を多く設定した場合、ネットワークの通信量もそれに伴い増加することが予想される。そこで統括監視システムがパフォーマンス情報を取得する際のネットワークへの影響を確認するため、データバッファサイズを変化させながらネットワークの通信量を測定した。測定の際の設定は
パフォーマンス情報取得インターバル：2 秒
データバッファサイズ：0, 10, 20, 30, 40
とした。統計的な揺らぎの影響を少なくするためには、ある程度のデータバッファサイズが必要となる。しかし、この値が大きすぎると反応速度への影響があるため、極端に大きな値の設定はできない。図８は上記の設定における、ネットワークの通信量を示したものである。前述の予想通り、データバッファサイズを大きくするにつれ通信量も大きくなることが確認できる。測定された通信量は 100kbps 以下であり、今日のネットワークの帯域や Web アプリケーションの通信量と比べても無視できるほど十分に小さいといえる。

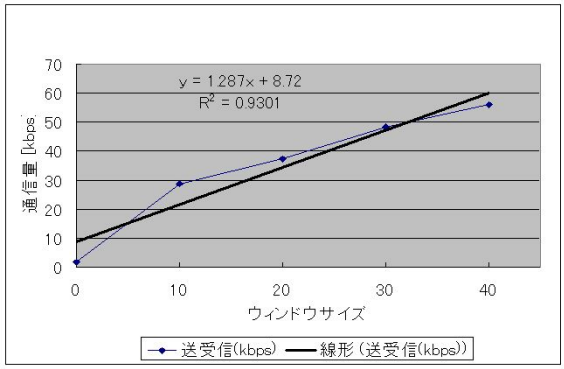

図８．データ・バッファサイズによる通信量の変化。回帰直線も図示してある。

オーバーヘッド

自律的広域負荷分散技術のオーバーヘッドについて調べるため、モニタリング･エージェントを停止させた状態において負荷を段階的に増大させた場合の CPU 使用率とスループット（単位時間当たりのシナリオ実行数）と、モニタリング・エージェントを起動させた状態においての CPU 使用率とスループットを測定し、両者の比較を行った。エージェント起動あり・なしで CPU 使用率、スループットいずれにも大差はなく、エージェント起動による影響は CPU 使用率で +5% 以内、スループットで -1% 以内に収まっていた。従って、統括監視システムが与えるオーバーヘッドは、無視できる程度に小さいといえる。

ドメイン処理能力のゲインの評価

アプリ B の負荷を一定（１０ユーザーで固定）にし、アプリ A の負荷を変化させていったとき、以下で定義されるスループットのゲインの変化をみる。

$$\mathrm{Gain}[\%] = \frac{\text{スループット (負荷分散有)} - \text{スループット (無)}}{\text{スループット (無)}} \times 100.$$

負荷分散システムによるゲインを図示したものが図９である。この結果から、負荷分散システムを利用することにより、複数ドメイン間に

渡って ICT 資源を自律的に有効活用するとが実現でき、負荷分散システムを使用しない場合に比べて、ゲインが増加していることが検証できた。

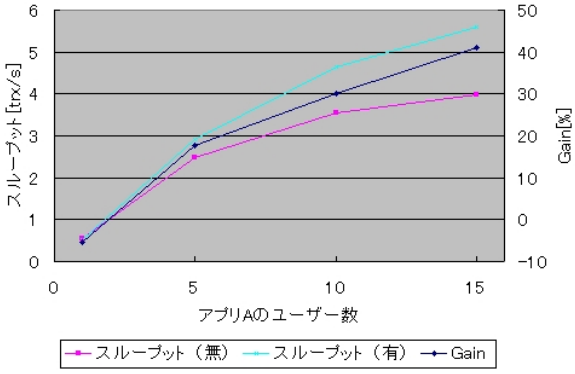

図９．スループットのゲインの変化

5　まとめ

本論文では、待ち行列理論に基づきアプリケーションごとに平均応答時間を推定するパフォーマンス・モデルを導入した。そして、与えられたリクエスト到着率を変化させることで平均応答時間をコントロールすることによって複数ドメイン間での自律的負荷分散を行うアルゴリズムについて述べた。さらに、２ドメインの実験環境を構築し、実装した各コンポーネントを用いて実証実験を行った結果、負荷が増大していったときに処理能力に関する効果的なゲインの向上を実現した。

本論文で提示した手法はシステムの詳細情報によることなく、応答時間やリクエスト到着率といった少数のジェネリックな情報のみを用いた抽象度の高いロジックでそれぞれのドメインが自律的な判断に基づいて負荷分散を行う。また、現行のシステムのようなトップダウン管理型やスポーク型ではなく、ドメイン単位での自由参画を想定した技術である。従って、不可避的に異機種混在システムにまたがる処理を行う Web サービスに適合しやすく、Web サービスなどの分散処理を利用する企業体や自治体、電子政府、学術団体、学校など特定の団体を想定した複数ドメイン間負荷分散技術において重要な意義を持つものといえる。

参考文献

[1] Tom De Wolf and Tom Holvoet, ”Towards autonomic computing, agent-based modelling, dynamical systems analysis, and decentralised control”, Workshop on Autonomic Computing Principles and Architectures (AUCOPA), 2003.

[2] Levy, R.; Nagarajarao et.al, ”Performance management for cluster based Web services”, IFIP/IEEE Eighth International Symposium on Integrated Network Management, 2003.

[3] Limin Wang, Vivek Pai and Larry Peterson, “ The Effectiveness of Request Redirection on CDN Robustness ”, Proceedings of the 5th symposium on Operating Systems Design and Implementation, p.p.345-360, 2002.

[4] 下川俊彦, 吉田紀彦, 牛島和夫, ”ネームサーバを用いた柔軟な負荷分散”, インターネットコンファレンス ’99, 107-116 (December, 1999).

[5] 馬場始三、山口英、”DNS を用いた広域負荷分散の実装、” 研究報告 - 分散システム／インターネット運用技術　No.036, 1998.

[6] Daniel A. Menasce, Virgilio A. F. Almeida, Lawrence W. Dowdy, Performance By Design: Computer Capacity Planning by Example, Prentice Hall, 2003.

[7] 亀田寿夫, 李頡, 紀一誠，情報数学講座 15 性能評価の基礎と応用，共立出版株式会社, 1998.