未来負荷予測を考慮した動的なミラーサーバ提供システム

田中　彰* 森口一郎**

サーバの将来予測負荷量（load average［一分間平均］）が最大閾値を超えた場合、同一なコンテンツを保持するmirror serverを負荷分散リストに追加し、逆にサーバの負荷量が最小閾値を下回った場合、そのサーバを負荷分散リストから削除する動的なミラーサーバ提供システムを開発した。サーバの負荷はsnmpで収集し、最小二乗法を使い負荷予測を行う。このときの負荷増加率が急であった場合は高性能なサーバを追加し、緩やかであった場合は低性能なサーバを追加する。この負荷予測システムにより、負荷の急増を早めに察知し、ミラーサーバを増加させる事で、リダイレクト先に指定されているサーバへの負荷集中を軽減させる事ができる。

キーワード：動的負荷分散、ミラーリング、snmp、最小二乗法

Dynamic Server Mirroring System Using Load Change Rate Prediction

Akira TANAKA* and Ichirou MORIGUCHI**

A dynamic server mirroring system was developed. Depending on the predicted future load average, this proposed system adds or deletes the address of a mirror server that is included in a load distribution server list. In case the load average exceeds a defined maximum threshold, the system adds one server to the server list. On the other hand, one server is deleted if the predicted loads are below a minimum threshold. The loads of server are collected through snmp, and the future loads are inferred by least-square method. If the growth rate of load data is steep, a high spec mirror server is assigned. If the growth rate is moderate, a low spec mirror server is assigned. By this load distribution system, sudden increases of load are predicted, and the concentration of load on one server which is assigned by redirection can be reduced.

Keywords: dynamic load balancing, mirroring, snmp, least-square method

*東京情報大学　総合情報学部　情報システム学科学部学生
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems, Undergraduate Student
**東京情報大学　総合情報学部　情報システム学科
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems

2013年12月12日受理

1．はじめに

近年、動画等のコンテンツが増加し、サーバにかかる負荷も増大している。そこで、負荷を軽減するためにミラーリングという手段がよく用いられている。

ミラーリング時のアクセス振り分けを行う代表的な方法にはDNSラウンドロビンがあるが、この手法はサーバの状態を確認することができない。このため、クライアントはダウン状態あるいは高負荷状態のサーバにアクセスしてしまう欠点があり、アクセスができなかったり、サーバごとに負荷の偏りを生じさせてしまったりする問題を抱えている。この問題を改善するために、様々なリクエストを振り分けるサーバの選択法の研究が行われている[1][2]。これらの研究により、サーバごとの負荷量のアンバランスは改善されてきている。しかし、サーバリソースの無駄に関しては着目されておらず、これらの論文で提案されているシステムでは、低負荷時、ミラーサーバを複数台稼働させておく必要がないにも関わらず、常にミラーサーバを一定台数稼働させておく必要があり、サーバリソースの無駄を生じさせてしまう問題を抱えている。

この問題に対し、入江はwebサーバの負荷状況に応じてミラーサーバを増減させる動的ミラーリングシステムを提案し、サーバリソースの無駄を削減した[3]。しかし、このシステムは急激なサーバ負荷増加によるリクエストの取りこぼしが生じる問題があり、動的ミラーリングのタイミングの最適化が課題となっている。

本研究ではsnmpを利用し、DNSラウンドロビンの問題点を解決するシステムを提案する。さらに、snmpで収集した過去の負荷データを元に、最小二乗法で変化率を導き出し未来の負荷量を予測することで早い段階でサーバを追加し、急激な負荷の増加に対応できるシステムを提案し、評価を行った。提案システムにより死活検知、最も低負荷なサーバへのリクエスト振り分け、負荷量に応じたサーバ数の増減の機能を追加することでDNSラウンドロビンの弱点を克服する事ができた。さらに負荷急増を早い段階で予測し、ミラーサーバが急増前に負荷分散に加わることで、特定のサーバへの負荷集中を軽減することができた。

2．システムの構成

本システムのサーバ、クライアントは全て仮想マシン上に構築されている。構築したサーバは、20MBの画像ファイル（以下、コンテンツ）を保持するbase serverが１つ、同一コンテンツを保持する性能の異なるmirror serverが２つ、リクエストの振り分けや各mirror serverの負荷を管理するcontrol serverが１つ、サーバアクセスを行うclientマシンが１つである。

2．1　システム概要

本システムでは、一般ユーザ（以下、client）がweb　serverへアクセスを行う時は、まずcontrol serverにアクセスする。control serverはsnmpにより常に各mirror serverのload average（１分間平均）を監視し、最も負荷の低いserverにclientの接続をリダイレクトさせる（図１）。またcontrol serverは負荷を監視するmirror serverのIPアドレスリスト（以下、list）を保持しており、負荷状況に応じてアドレスを追加、または削除し、負荷分散に参加するサーバの台数を変化させている。

mirror serverは、base server、high spec server（以下、high server）、low spec server（以下、low server）がある。現実的なネットワークでは、ミラーサーバごとに性能が違う可能性が高いた

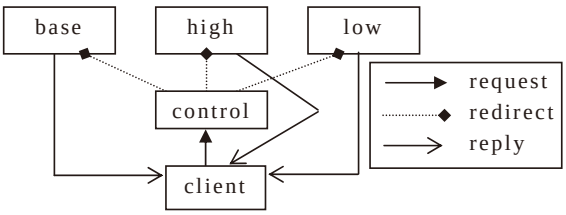

図１　システム構成図

表1　各mirror serverのメモリ量とIPアドレス

	base server	high server	low server
memory	512MB	512MB	256MB
IP	192.168.64.194	192.168.64.195	192.168.64.196

め、本研究で構築したシステムはhigh serverとlow serverとでメモリ量に差を持たせた。しかし、CPUはload averageの高負荷状態の判定基準を統一するためにどちらもデュアルコアにした。各ミラーサーバのスペックとIPアドレスを表1に示す。またどのサーバもwebサーバとして機能させるために、Apacheを導入した。

control serverは各mirror serverの負荷量を取得するsnmpマネージャーであり、これにより取得した負荷の状態に応じて、クライアントのリクエストをリダイレクトするロードバランサーの役割も担っている。またcontrol serverのwebページに接続してきたclientをリダイレクションするために、mirror serverと同様にApacheを導入した。

2．2　測定するサーバ負荷（load average）

本研究ではload average（1分間平均）をsnmpで取得し、このデータを元に負荷分散を行った。load averageはCPU稼働率のことであり、現在処理しているプロセス数と処理待ちしているプロセス数を加算した値から、CPUが実行可能なプロセス数を割ることで求めることができる。したがって、load averageが1.00（100%）以下であれば、プロセスが滞りなく実行され、それよりも大きい場合は実行に遅延が生じている事になる。load averageが高い場合、CPUかI/Oにボトルネックがあると考えられる。CPUの負荷が高い場合はCPUの増設、スワップによるI/Oの処理時間がボトルネックの場合はメモリの増設を行うことでload averageを低下させることができる。

本研究で構築したシステムは、high serverのメモリを512MB、low serverのメモリを256MBとすることで、low serverの方が多量のプロセスが発生した場合I/O出力による処理待ちプロセスが発生しやすくなっている。したがって、サーバへのデータ要求回数が同じであってもlow serverの方がload averageが増加しやすく、high serverの方がload averageが増加しづらくなっている。

なお、本システムで使用しているサーバのCPUは全て2コアであるため、load averageが2.00（200%）を超えた場合、高負荷状態であると判断した。

また本システムは各サーバの負荷状態に応じて負荷分散に参加するmirror serverを増減させるため、高負荷状態である200%よりも低い値（170%）をmirror serverの増加タイミング（以下、最大閾値）として設定した。さらに負荷量が30%以下の時、サーバ数を減少させるタイミング（以下、最小閾値）として設定した。

負荷の測定間隔はcontrol serverやmirror serverにsnmpによる負荷をかけ過ぎず、できるだけ多くのデータを取得し、より正確な未来の負荷予測を行うために3秒に1度の間隔で行った。このデータを元に、control serverは接続してきたclientを最も負荷の低いserverにリダイレクトする。また、接続してからリダイレクトされるまでの時間を0.01秒と設定し、clientにcontrol serverからリダイレクトされていることを意識させないようにした。

2．3　負荷予測手法について

本研究では未来の負荷の予測を行い、高負荷になる前に新しいmirror serverを追加する必要があるため、最小二乗法により変化量を関数化し、未来の負荷量の推定値を導出した[4]。

最小二乗法では、任意の関数式を求めることができる。もしべき関数であれば次数が増えるほど急激な増加などの予測も可能となるが、計算が複雑化する。よって、本システムでは計算が複雑でなく、瞬間的負荷変動による予測への影響が小さい二次関数式を使い負荷の予測を行った。また、最小二乗法で60秒後の負荷をどの程度正しく推測できるかを確認するため、サンプリング数を10個、20個、30個と変えて実験

を行った実験の方法は以下の通りである。

1．serverからコンテンツをダウンロードしてくるclientプログラムを使い、base serverに対してだけ負荷をかける。アクセスの発生は60秒毎に決められた回数分発生させる。発生のタイミングは全て同時ではなく、60秒の間でランダムな時間に生じさせることで、アクセス時間に差を生じさせた。各時間帯毎のアクセスを発生させる人数（以下、シナリオ）は、一日のインターネット利用平均[5]を元に作成した。シナリオの詳細は３．２　評価手法で後述する。

2．作成したシナリオを使用し、base serverに負荷をかけ、control serverのsnmpにより、load average（１分間平均）を％で３秒毎に一度取得し、シナリオ開始から終了するまで記録する。

3．最小二乗法により60秒後の負荷量の予測を行うために、シナリオ開始から終了まで記録した負荷データの中から負荷をランダムにひとつだけ選択し、連続した３秒後毎の負荷データを30個分取り出す。また取り出した最後のデータから60秒経過した負荷量を実測値として取り出す。

4．計算しやすくし、かつ図を見やすくするために、取り出した最初のデータを０秒とし、３秒、６秒、９秒、……、87秒と経過した時間に負荷量を対応させる。実測値のデータは87秒から60秒経過した147秒とする。

5．取り出した30個分のサンプルから古い時間の負荷データを削除した20個（30秒から87秒目までのデータ）と10個（60秒から87秒目までのデータ）のサンプルを作成する。

6．最小二乗法を使い、それぞれ関数式を求め、取り出した負荷の最後の負荷の時間から60秒経過した未来負荷予測値を導出し、実測値と比較する。

シナリオ開始から終了まで記録した負荷データから取り出した負荷データ30個分を表２、古

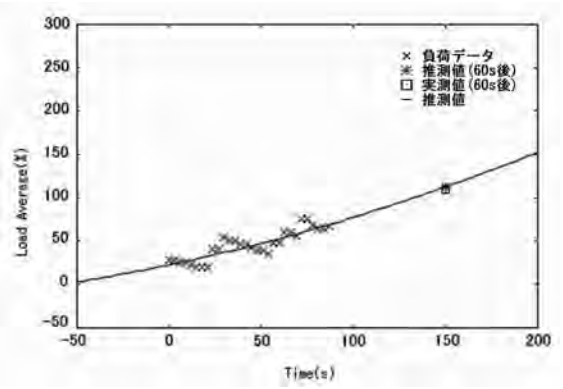

図２　サンプルデータ数30個の予測結果

表２　30個分のサンプルデータ

時間（s）	0	3	6	9	12	15	18	21
負荷量（%）	28	28	25	25	23	20	20	20
時間（s）	24	27	30	33	36	39	42	45
負荷量（%）	41	41	55	50	50	46	46	42
時間（s）	48	51	54	57	60	63	66	69
負荷量（%）	39	39	35	48	48	60	60	56
時間（s）	72	75	78	81	84	87		
負荷量（%）	76	76	69	64	64	67		

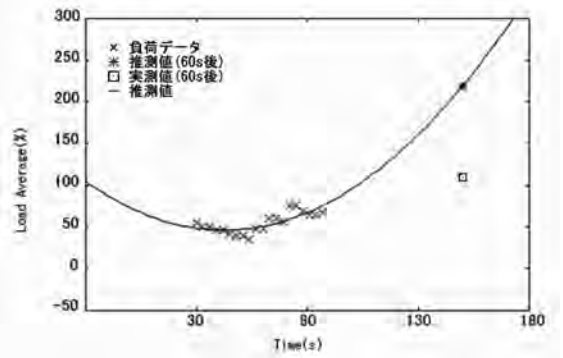

図３　サンプルデータ数20個の予測結果

表３　20個分のサンプルデータ

時間（s）	30	33	36	39	42	45	48	51
負荷量（%）	55	50	50	46	46	42	39	39
時間（s）	54	57	60	63	66	69	72	75
負荷量（%）	35	48	48	60	60	56	76	76
時間（s）	78	81	84	87				
負荷量（%）	69	64	64	67				

い順に負荷量を削除して作成した20個分のデータを表３に、10個分のデータを表４に示す。実測値と推測値の差を表５に示す。

また求めた式のxは時間（s）であり、xに時間を代入するとその時間における予測負荷量を求めることができる。

データサンプル数30個（図２）

0.0010083693*x*x+.4492206034*x+22.5505052322

データサンプル数20個（図３）

0.0150161748*x*x+-.3783456982*x+48.5713445012

データサンプル数10個（図４）

0.0103775103*x*x+0.1204097822*x+69.4432859793

実測値と推測値の差は、表５よりサンプル数30個が２％で最も差が小さく、サンプル数10個が-417％と最も大きくなった。この実験からサンプルが増えるほど推測値と実測値の差は小さくなり、より正確な負荷の推測が行えることがわかる。

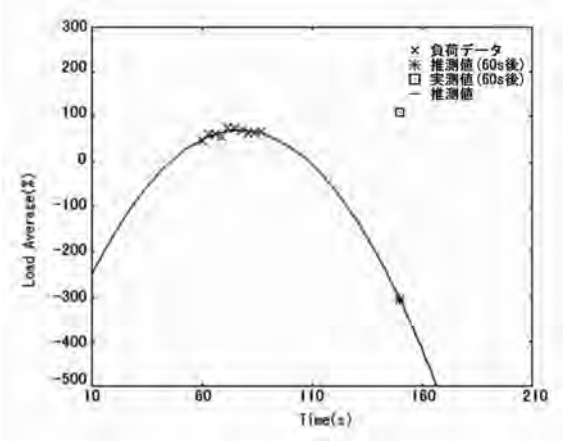

図４　サンプルデータ数10個の予測結果

表４　10個分のサンプルデータ

時間（s）	60	63	66	69	72	75	66	69
負荷量（%）	48	60	60	56	76	76	60	56
時間（s）	78	81	84	87				
負荷量（%）	69	64	64	67				

表５　実測値と推測値の差

実測値	110	
サンプル数	推測値	差
10	-307	-417
20	219	109
30	112	2

以上の結果を踏まえ、本システムでは現在の負荷データから過去30個分（90秒前まで）の記録した負荷データを使い、60秒後の負荷推定値を導出した。またこの推測した値が300％以下の場合、未来のアクセス数の増加は少ないと予測できるため、low serverをlistに追加し、300％を超えていた場合、アクセス要求量が増加してもload averageの上がりにくいhigh serverをlistに追加するようにした。

２．４　振り分け先判定プログラム

最も負荷の低いserverにclientをアクセスさせるために、listに登録されているサーバに対してsnmpを３秒に１度実行し、最も負荷の低いサーバにリダイレクトを行うようにした。

base serverのアドレスや負荷分散に使うmirror serverのアドレスは設定ファイルにあらかじめ記述しておく。この設定ファイルを読み込み、mirror serverの追加を行うようにした。

本システム稼働直後は、listにはbase serverのみ記載してある。これにより、初期状態ではbase serverの負荷量だけ監視し、アクセスは全てbase serverにリダイレクトされる。

負荷分散に参加するmirror serverの増加は、snmpwalkを行うと同時にbase serverに対して最小二乗法を行い、60秒後の負荷も予測することで行った。もし最小二乗法により求めた60秒後の負荷の予測値が170％を超えていた場合、mirror serverのアドレスを設定ファイルから読み込み、listに追加する。

これ以降は最後に追加したmirror serverに対して負荷予測を行い、mirror serverの追加が必要か判定を行うようにした。また最後に追加したmirror serverに対しては負荷予測を行った後、負荷量が30％以下でないかチェックを行い、もし30％以下であった場合そのmirror serverをlistから削除し、負荷分散に参加するサーバの数を

減少させた。

しかし、新しいmirror serverを追加直後は、負荷量は０％に近いため、新しいmirror serverが追加された後、３秒後の負荷量チェックにより30％以下の条件に適合し、listから削除されてしまう。これを回避するため、サーバがlistに追加されてから一定時間削除されない時間を30秒設け、30秒経過後でも負荷が30％以下であった場合のみ、そのmirror serverをlistから削除するようにした。

２．５　client

clientマシンはapache bench（以下、ab）というソフトを使い、clientマシン一台で複数のユーザがweb serverにアクセスしている状態を作り出し、web serverを高負荷状態にした。

abとはApacheに付属する負荷テストツールであり、主にwebサーバがアクセスによる負荷にどこまで耐えられるかを測定する用途で用いられている。

本研究ではサーバに対してより現実的な負荷をかけるために、abコマンドとシェルスクリプトを使い、アクセス人数を時間帯に応じて増減させられるようにプログラム作成した。サーバへの負荷は、各mirror serverが保有するコンテンツを取得することで行う。実行タイミングは後述するタイミングファイルに従ってランダムに実行されるようにした。

３．システムの評価手法と結果

本システムの有効性を評価するために、提案システムである負荷の未来予測をしてミラーサーバの追加や削除を行うシステムと負荷の未来予測を行わないでミラーサーバの追加と削除を行う検証用システムを用いて実験を行った。実験はclientの負荷をかけるプログラムを使ってbase serverやmirror serverに負荷をかけて行った。詳細は３．２の評価手法に記述した。

３．１　検証用システム

検証用システムは未来予測を行わず、測定を行った時点での負荷が170％を超えると、新しいサーバのアドレスをlistに加えるようしている。検証用システムの増加タイミングを170％を超えた時と設定した理由は、高負荷状態である200％よりも少し早いタイミングでサーバを増加させることで、高負荷にならないようにするためである。この検証用システムを使い、未来負荷予測を行う提案システムの有効性を検証する。また提案システムと検証用システムの違いを以下の表６に示す。

表６　実測値と推測値の差

	提案システム	検証用システム
負荷の予測	行う	行わない
負荷の予測に使用するサンプル数	30	−
負荷量の測定間隔	３秒につき１回	３秒につき１回
サーバ増加タイミング	予測負荷170％超え	測定時の負荷170％超え
サーバ減少タイミング	測定時の負荷30％以下	測定時の負荷30％以下
リクエスト振り分け先	最も負荷が低いサーバ	最も負荷が低いサーバ

３．２　評価手法

１人につき１回mirror serverからコンテンツをダウンロードしてくるプログラムをclientで複数起動させた。アクセスは60秒毎に決められた人数がアクセスを行う。この60秒毎のスパンを1phaseとする。

アクセスタイミングは０～60秒間の間でランダムになっており、このランダムな時間を生成後にタイミングファイル（テキストファイル）として保存し、何度実験を行っても同じタイミングでアクセスを開始するようにした。また提案システムと検証用システムのアクセスタイミングも同一のタイミングファイルを使用し、同じタイミングでアクセスを開始するようにした。

phaseは全部で24phaseあり、これを２周期行うため、合計48phaseコンテンツのダウンロードを行った。

なお、前の実験による負荷などが残らないようにするため、プログラムを起動してから60秒間空白時間を置いた。その後、シナリオに従いclientによるアクセスを再開した。

phase毎のアクセス人数は、一日のインターネットを利用する時間帯の平均人数と同様にアクセス数が変動するサーバを想定し、一日のインターネット利用平均を元に１時間毎の利用者の変動と、サーバに１分毎にアクセスする人数の変動を対応させて作成した（表７）。

また全体の人数は、各serverのload averageを200％付近まで上昇させるために、100人とし、全体の人数に各時間帯の接続率をかけてphase毎のアクセス人数を決めている。これにより、アクセスが急増する時間帯（phase19～23）では、31～42人が60秒の間でランダムにアクセスを行うことになり、サーバの負荷も上昇しやすくなっている。さらにアクセスが少ない時間帯（phase １ ～ ７）においては６～12人が60秒の間でランダムにアクセスを行うようになっており、サーバの負荷は低くなるようになっている。

また負荷の急増をより極端にした場合でも、提案したシステムが有効であるか評価するため、phase20から24までの接続人数を表８のように変更して実験を行った。

アクセスタイミングは、実験を開始する前に一度だけ各phaseで作成した接続人数分０～60の間でランダムに数値を作成し、タイミングファイルに保存した。この数値は各phase開始からコンテンツのダウンロードを開始するまでの時間であり、一般的なwebサイトなどでは、一般ユーザは全員が全く同じタイミングで接続するのではなく、各ユーザがランダムな時間で接続を行うため、phase中のダウンロードの開始タイミングをずらした。

評価を行う上で、listに記載されていないmirror serverの負荷量の変化も記録するために、control serverのリダイレクト先を決定するための負荷計測プログラムで実行するsnmpwalkとは別に、全てのmirror serverのデータを３秒毎に記録するプログラムを実行した。

表７　phase毎のアクセス数

phase	1	2	3	4	5	6	7	8
接続人数	6	2	2	3	6	8	12	17
phase	9	10	11	12	13	14	15	16
接続人数	20	16	17	15	15	18	17	18
phase	17	18	19	20	21	22	23	24
接続人数	19	23	31	40	42	34	22	12

表８　変更したphaseの人数

phase	20	21	22	23	24
接続人数	41	50	52	54	42

３．３　評価結果

未来予測を行うシステムの実験結果を図５に、未来予測を行わないシステムの実験結果を図６に示す。またphase20から24にかけてアクセス人数を急増させた場合の未来予測を行うシステムの実験結果を図７、未来予測を行わないシステムの実験結果を図８に示す。

サーバの負荷が急増するphase19では、図５の提案したシステムはどのサーバも負荷量が200％程度であった。これに対し図６の未来予測を行わないシステムでは、phase19で負荷が250％付近まで上がり、かなりの高負荷状態と

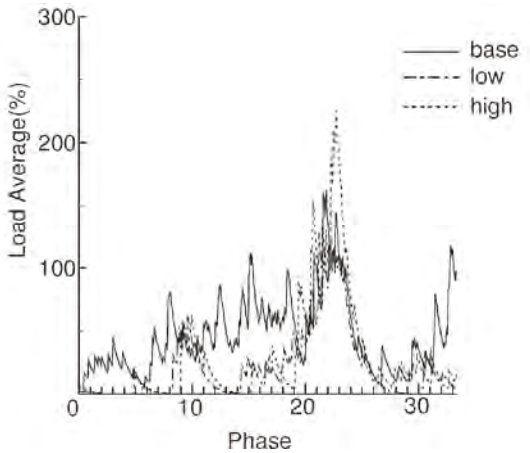

図５　提案システムの評価結果

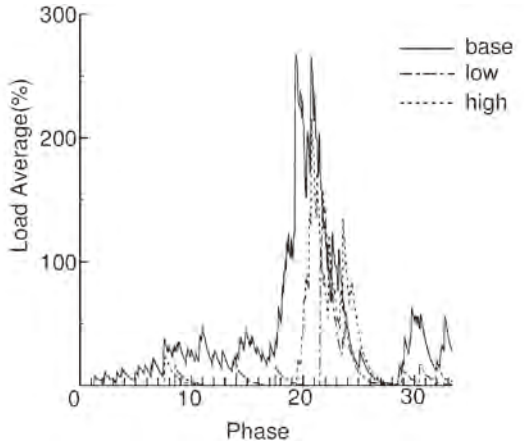

図６　検証用システムの評価結果

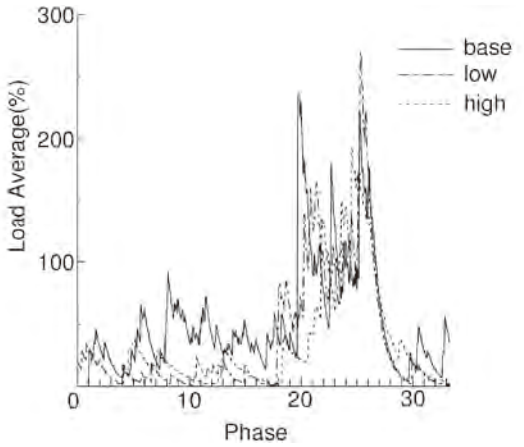

図７　負荷急増時の提案システム評価結果

なった。

また提案したシステムでは、phase16からlow serverとhigh serverが共に負荷分散に参加し、どのサーバも負荷を200％付近で抑えることができた。

図６の未来予測を行わないシステムでは負荷が急増するphase19でhigh serverが負荷分散に参加しているが、phase20でbase serverの負荷は270％近くまで上がった。low serverはphase21で負荷分散に参加するが、high serverの負荷が急激に高まり、この時点で荷量は300％近くまで上がった。

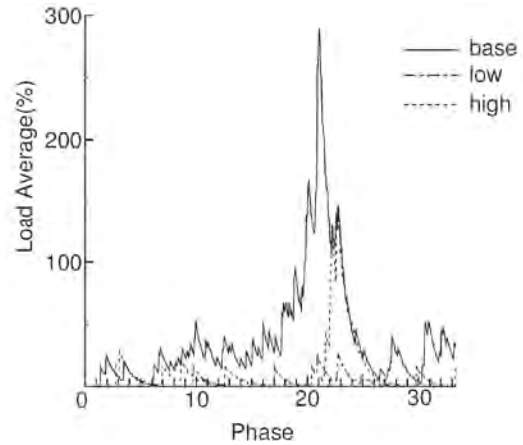

図８　負荷急増時の検証用システム評価結果

負荷を急増させた実験では、提案システムの負荷（図７）は、負荷が急増するphase19のひとつ前のphase18からhigh serverとlow serverがほぼ同時に負荷分散に参加し、どのサーバも最大250％近くまで負荷が上昇した。これに対し、比較用システムはphase19からbase serverの負荷が急激に上昇し、phase21では300％付近まで上昇した。その後phase22ですぐにhigh serverも負荷分散に参加し、base serverの負荷は150％付近まで下がった。low serverは負荷分散には全く参加せず、負荷を急増させるとサーバごとの負荷の偏りがより顕著に現れた。

３．４　考　察

本研究の目的は、DNSラウンドロビンの死活検知ができない点や、サーバごとの負荷の偏り、低負荷時のリソースの無駄な点の解決に加え、入江が提案したシステムの課題であるサーバを増加するタイミングの最適化を行うことである。

提案したシステムにより、目的のうち、以下の４点を解決することができた。

１．サーバの死活検知
２．サーバごとの負荷の偏りの緩和
３．低負荷時のサーバリソースの節約
４．負荷急増時における、サーバ増加タイミングの最適化

しかし、４. の負荷急増時における、サーバ増加のタイミングの最適化に関しては、本研究では最小二乗法を使い最適化を図ったが、別の予測手法を利用した場合、より正確な予測を出すことが出来る可能性が有るため、今後より研究を深めていく必要があると考える。

また本研究では３秒に１度snmpにより負荷の取得を行ったが、情報取得の間隔が増えるほど、負荷予測精度は低下してしまい、逆に情報取得間隔を狭めると、ミラーサーバの負荷量やトラフィック量を高めてしまう問題があるため、今後このバランス関係についてより研究を深めていく必要があると考える。

４. ま と め

提案したシステムではDNSラウンドロビンの問題点解決のために、まずサーバの負荷をsnmpで常時負荷量を監視し、サーバの負荷量を取得できないサーバはダウンしたとみなし、負荷が取得できるようになるまで（サーバが稼働するまで）アクセスを振り分けないようにし、死活検知を行うようにした。次に、負荷量が最も低いサーバにアクセスを送るようにすることで、サーバごとの負荷が偏らないようにした。入江の提案したシステムの問題点であったサーバの増加タイミングに対しては、過去の負荷データから未来の負荷量を予測し、早い段階でサーバを追加することで、特定のサーバにアクセスが集中し、高負荷になってしまう現象を軽減し、負荷の急激な増加に対応できるようにした。

今回の評価実験により、考察でも述べた通り、提案したシステムは検証用システムに比べて、負荷予測により早めに負荷の急増を察知し、サーバを増加させて分散を行うことで、入江らが開発したシステムの負荷急増時のリクエスト取りこぼし問題に対して有効であることが示せた。さらに負荷の増加率が急か緩やかであるかによってhigh serverかlow serverを提供するか決定することで、より負荷急増に対応できるシステムを構築するができた。

最近ではサーバ等の負荷分散にはオープンソースであるLinux Virtual Server（LVS）が使われることもある。LVSの負荷分散アルゴリズムには、重み付けラウンドロビンや最小コネクション数、応答速度等があるが、mirror server自体の数を負荷の状況に合わせて増減させるアルゴリズムはない。一方、本システムは負荷量を予測し、負荷状況に合わせてサーバを増減させ、サーバのリソースを活かすことができると考えられる。

ただし現時点では、本研究で設定した閾値やlow、high spec serverの選択の値やsnmpによる情報収集の間隔などのパラメータは現在暫定的に設定したものであるため、考察でも述べた通り今後パラメータを最適化して性能を高めていく必要がある。

さらに本システムでは負荷の予測に最小二乗法（二次関数）を利用したが、分析・予測の方法には統計的予測や移動平均法、指数平滑法、成長曲線、季節調整法等の様々な予測方法がある[6]。これらの予測法と比較し、より正確な手法を模索する必要がある。

参考文献（reference）

［１］佐竹伸介，稲井寛：「Webサーバクラスタにおける定期的な負荷情報に基づく非確率的なサーバ選択方法」，信学論B，vol. J88-B，No. 10，pp. 1968-1978 (2005).

［２］横田裕思，木村成伴，海老原義彦：「DNSフィルタ方式によるミラーサーバ選択法の提案と実装」，情報処理学会論文誌，vol. 44，No. 3，pp. 682-691 (2003).

［３］入江正行：「Webサーバの負荷状況に応じた動的ミラーリング機構」，香川大学工学部卒業論文 (2005).

［４］田島穂，小牧和雄：「最小二乗法と測量網平均の基礎」，東洋書店 (2001).

［５］財団法人インターネット協会：「インターネット白書2009」，株式会社インプレスR&D (2009).

［６］伊藤政志，岸野洋久，佐藤幸雄：「統計処理の手法がよくわかる本」，加藤文明社 (1985).