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

2006－OS－102 (6)
2006／5／13

応答時間の分散を利用したウェブサーバ接続数の自動設定

杉木 章義† 河野 健二‡

† 電気通信大学大学院 電気通信学研究科 情報工学専攻
‡ 慶應義塾大学 理工学部 情報工学科
E-mail: sugiki@zeus.cs.uec.ac.jp, kono@ics.keio.ac.jp

インターネットサーバの人手によるパラメータ設定は，管理コストの増大を招くことが知られており，自動化が求められている．本論文では，ウェブサーバを対象に，応答時間の分散を利用したサーバ接続数の自動設定手法を提案する．サーバは高負荷になると応答時間の分散が急激に大きくなるという現象を示す．本機構は応答時間の分散が大きくなったことを検出し，サーバ接続数を自動設定する．2つのワークロードに対して，応答時間の増加を抑えながら，スループットを最大にするサーバ接続数に自動設定することができた．

Tuning the maximum number of connections of a web server using deviation of server latency

Akiyoshi Sugiki† Kenji Kono‡

† Dept. of Computer Science, University of Electro-Communications
‡ Dept. of Information and Computer Science, Keio University
e-mail: sugiki@zeus.cs.uec.ac.jp, kono@ics.keio.ac.jp

Automatic parameter-tuning of Internet servers is demanded to reduce high administrative costs by manual tuning. In this paper, we present a tuning mechanism for the maximum number of connections of a web server. Our mechanism is based on the detection of server latency deviation because most servers show unstable behavior when it is overloaded. Experimental results show that our prototype achieved high throughput while reducing server latency on two different workloads.

1 はじめに

ウェブサーバの人手によるパラメータ設定は，多くの経験や時間を必要とし，管理コストの増大を招くことが知られている．適切なパラメータ値を得るためには，実運用に近い予測ワークロードをサーバに与え，パラメータ値を変えながら，何度もテストを繰り返す必要がある．また，ワークロードの変化に応じて，パラメータ値が適切に再設定される必要がある．そのため，近年，自動的なパラメータ設定を可能とする技術が研究されている [1, 2, 3, 4].

ウェブサーバの主要なパラメータの一つに，サーバ接続数がある．サーバ接続数はサーバに同時に接続できるクライアント数の上限を決めるパラメータである．サーバ接続数は一般に設定が難しいことが知られており，適切な値はサーバの処理能力やワークロードによって大きく異なる．

本論文では，ウェブサーバを対象に，応答時間の分散を利用したサーバ接続数の自動設定手法を提案する．サーバは高負荷になると，応答時間の分散が急激に大きくなるという現象を示す．本機構は応答時間の分散が急激に増加したことを検出し，サーバ接続数を自動設定する．応答時間の分散とサーバ接続数の対応関係を回帰直線を当てはめながら求め，分散の増加によって回帰直線の傾きが急激に大きくなる直前の値にサーバ接続数を自動設定する．

本機構を Linux 上で動作する Apache ウェブサーバ [5] を対象に実装した．サーバ接続数設定機構を外部ライブラリとして実現しており，サーバの実行時に Apache とオペレーティングシステム (OS) 間に介在する．そのため，既存のサーバや OS を改変せず利用可能である．サーバ接続数の設定や応答時間の測定は，システムコール呼び出しを横取りし，その引数を書き換えることで行う．

この実装を用いて実験を行ったところ，2つのワークロードに対して，応答時間の増加を抑えながら，スループットを最大にするサーバ接続数に自動設定することができた．自動設定された値は，手動設定

で求めた最適値に十分近い．また，ワークロードを突然変化させた場合にも，約10分程度で最適値に近いサーバ接続数に自動設定することができた．

以下，2章ではウェブサーバの接続数の分析を行う．3章でサーバ接続数の自動設定機構を示す．4章で実装について示し，5章で実験とその結果を示す．6章で関連研究について述べ，7章でまとめを示す．

2 ウェブサーバ接続数

2.1 サーバ接続数設定の必要性

ウェブサーバのサーバ接続数は，サーバに同時に接続することができるクライアント数の上限を決めるパラメータである．Apacheウェブサーバでは，サーバ接続数は`MaxClients`パラメータとして知られている．

サーバ接続数は適切な値に設定することが難しい．適切な値はハードウェア性能，ワークロードに大きく依存して決まる．サーバ接続数はボトルネックとなる箇所の最大性能によって決まり，CPU，メモリ，ディスクなどさまざまな箇所がワークロードの変動によって，ボトルネックとなる．また，ボトルネックとなる箇所が分かっても，利用可能な資源の量から，サーバ接続数の適切な値を導き出すのは難しい．

ワークロードによって，適切なサーバ接続数の値が大きく異なることを示すため，2つのワークロードで実験した結果を図1に示す（詳細な実験環境は5.1節を参照）．使用したワークロードは，標準的なウェブサイトを想定したSPECWeb標準，大きな画像を多数含むサイトを想定したSPECWeb大の2つである．それぞれのワークロードに対し，サーバ接続数を変えながら，スループットと平均応答時間を測定した結果を示す．測定のばらつきを抑えるため，移動平均で平滑化してある．

図1をみると，ワークロードごとにスループットや平均応答時間が大きく異なっていることが分かる．同じサーバ接続数200をみても，SPECWeb標準のスループットは5.6MB/sであり，SPECWeb大のスループットは21.6MB/sと4倍近い差がある．また，同じワークロードでも，サーバ接続数によってスループットや応答時間が異なる．SPECWeb標準において，サーバ接続数1200では15.3MB/sと最大スループットに比べ低下しているが，サーバ接続数800では最大に近い18.4MB/sが得られる．

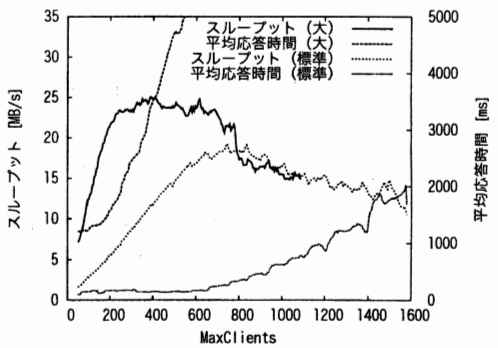


図1: サーバ接続数の手動設定結果

したがって，ワークロードごとにサーバ接続数を適切に設定することで，高負荷時のスループットの低下や平均応答時間の増加を抑えることができる．本論文では，応答時間の増加を抑えながら，スループットを最大化するサーバ接続数を求める．図1をみると，スループットが最大となるようなサーバ接続数の設定にはある程度の幅があり，その中でも応答時間が最小となる設定を目指す．SPECWeb標準の場合，サーバ接続数800を中心に最大に近いスループットが得られる．一方，応答時間はサーバ接続数800前後から増加に転じるため，適切なサーバ接続数は800である．SPECWeb大の場合には，サーバ接続数200–600の幅で最大に近いスループットが得られ，平均応答時間はサーバ接続数200で増加に転じる．よって，SPECWeb大における適切なサーバ接続数は200である．

2.2 サーバ接続数と応答時間のばらつき

本論文で提案する機構では，応答時間の分散を用いたサーバ接続数の自動設定を行う．応答時間とは，クライアントからの要求をサーバが受信してから，その要求に対する応答をサーバが完了するまでの時間である．このようにすることで，クライアントまでのネットワークによる影響を含まず，応答時間からサーバ自身の負荷状態を知ることができる．サーバが高負荷になると，応答時間のばらつきが大きくなるという現象を示す．

図2に，サーバ接続数を時間とともに変えながら，平均応答時間をプロットしたものを示す．これは2.1節のSPECWeb標準ワークロードに対して，測定した結果である．5分以降，サーバ接続数を2秒間で5ずつ増加させながら，1秒ごとに平均応答

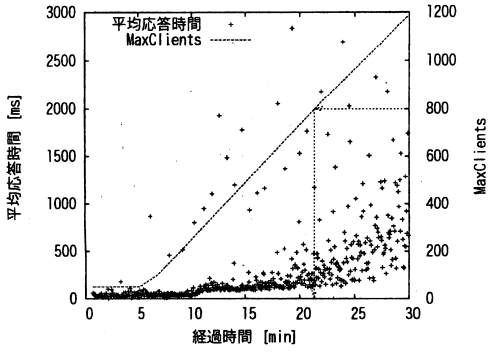


図 2: サーバ接続数と平均応答時間のばらつき

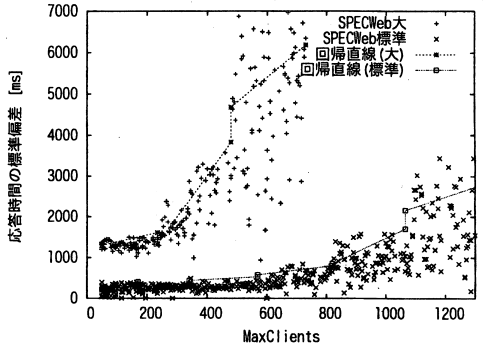


図 3: サーバ接続数と応答時間の標準偏差

時間を測定している．図 2 をみると，適切なサーバ接続数 800 付近で，平均応答時間のばらつきが急激に大きくなることが分かる．

2.3 応答時間の標準偏差の分析

本論文で提案するサーバ接続数の設定機構は，応答時間のばらつきが急激に大きくなるサーバ接続数の値を検出し，その値にサーバ接続数を設定する．

本論文では，応答時間のばらつきをはかる尺度として応答時間の標準偏差を用いる．図 3 にサーバ接続数と応答時間の標準偏差の関係を示す．これは，図 1 の実験で使用した 2 つのワークロードの結果である．

図 3 をみると，低負荷時にはほぼ一定だった標準偏差が，サーバが高負荷になると急に大きくなることが分かる．標準偏差自身も測定によって求められた値であるためばらつきが見られるが，標準偏差が急激に増加するサーバ接続数は，それぞれのワークロードの適切なサーバ接続数と一致する．SPECWeb 標準では，一定であった標準偏差が適切なサーバ接続数 800 前後で増加を始める．SPECWeb 大では，適切なサーバ接続数 200 前後から標準偏差が一定から増加に転じる．

なお，平均応答時間の増加を基準として，サーバ接続数を設定することも考えられるが，うまくいかないことがある．平均応答時間を用いたサーバ接続数設定の場合，サーバが高負荷になったことを検出するために，平均値の増加を基準として用いる．しかし，サーバが高負荷となる以外の要因，例えば他のパラメータの自動設定の結果によって，平均応答時間が変動することがある．実際，図 2 では我々が提案している keep-alive 時間の設定機構 [6] が背後で動いており，keep-alive 時間の設定によってクライアントの振る舞いが変わり，12 分付近で平均応答時間が増加している．一方，平均応答時間のばらつきは keep-alive 時間の設定によって変わらない．

3 サーバ接続数の自動設定

3.1 サーバ接続数の設定方式

本論文で提案するサーバ接続数設定機構は，サーバ接続数を段階的に増加させ，接続数の増加に対して応答時間の標準偏差が一定かどうかを確かめる．サーバ接続数の増加によってサーバが高負荷となり，標準偏差が一定から急激に増加する直前のサーバ接続数に設定する．一定であるかどうかを検出すればよいので，特定の標準偏差の値に依存しない．そのため，ワークロードが変わっても，この手法を用いることができる．

応答時間の標準偏差がサーバ接続数の増加に対して一定であるかどうかの検出は，図 3 上のある範囲に回帰直線を当てはめ，回帰直線の傾きを見ることで検出することができる．なぜなら，標準偏差が一定ではなくなる地点を検出するだけであり，標準偏差の正確な変動を検出する必要はないからである．

図 3 のサーバ接続数と応答時間の標準偏差に対して，回帰直線で近似したものを図中に示す．サーバ接続数 100 ごとに回帰直線を計算し，当てはめている．図 3 をみると，直線の近似でも応答時間の標準偏差の傾向をよく表していることが分かる．回帰直線の傾きが増加し始めるサーバ接続数は，それぞれのワークロードでの適切なサーバ接続数と一致する．SPECWeb 標準では，サーバ接続数 800 から回帰直

```
 1: // win: 回帰直線計算のためのウィンドウ
 2: typedef enum {ASCEND, DESCEND} dir_t;
 3: void mc_control(dir_t dir) {
 4:  while (true) {
 5:    // (1) 回帰直線の計算
 6:    x = now();     // 経過時間 [ms]
 7:    y = get_sd(); // 応答時間の標準偏差 [ms^2]
 8:    grad = regression(win, x, y);
 9:    // (2) 回帰直線の傾きの判定
10:    if (dir == ASCEND) {
11:      if (grad > threshold) break;
12:      maxclients += adj;
13:    } else {
14:      if (grad < threshold) break;
15:      maxclients -= adj;
16:    }
17:    reset_sd(); // 標準偏差の計算を初期化
18:    sleep(1000);
19:  }
20:  maxclients = regression_center(&win);
21: }
```

図 4: 自動設定機構の擬似コード

線の傾きが大きくなり，SPECWeb 大では，サーバ接続数 200 から回帰直線の傾きが大きくなる．

3.2 設定手法の詳細

本機構では，3.1 節の手法を忠実に実現する．サーバ接続数を段階的に増加させながら，サーバ接続数と応答時間の標準偏差の対応関係を回帰直線によって求める．回帰直線の傾きはサーバの負荷が低い間はほぼ一定であるが，サーバが高負荷となると一定ではなくなり，傾きが大きくなる．本機構はこれを検出し，サーバ接続数を設定する．

図 4 に，サーバ接続数の設定機構の疑似コードを示す．まずは，サーバ接続数をはじめて設定する場合について説明する．$dir = ASCEND$ の場合である．サーバ接続数を増加させながら（12 行目），過去一定範囲に回帰直線を当てはめ（5–8 行），その傾きをもとに判定を行う（11 行）．

(1) 回帰直線の計算：回帰直線はスライディングウィンドウを用いて，計算する．スライディングウィンドウとは，現在の時刻から過去一定期間のサーバ接続数と応答時間の標準偏差の組を記録しておくための配列である．ウィンドウ内の値をもとに回帰直線の傾きを計算する．図 4 では，8 行目で新たな時刻の測定結果を加え，回帰直線の傾きを計算している．現在，1 秒ごとにウィンドウに追加するものとし，ウィンドウの大きさは 3 分としている．

3.1 節では，サーバ接続数と応答時間の標準偏差の組に対して回帰直線を当てはめた．しかし，X 軸をサーバ接続数とした場合には，サーバ接続数が変化しない場合，回帰直線の傾きを計算することができない．そのため，設定手法の開始直後，スライディングウィンドウが満たさるまで，回帰直線が正しく計算できないという現象が発生する．この対策のため，実際の設定手法では X 軸を経過時間とし，その代わりサーバ接続数を一定の速度で変化させている．サーバ接続数が変わらない安定状態から回帰直線の傾きが計算できるため，直ちに設定を開始できる．また，X 軸を経過時間とすることで，3.3 節の安定状態からの再設定開始基準にも利用できる．

(2) 回帰直線の傾きの判定: 新しいサーバ接続数の決定のために，回帰直線の傾きと閾値を比較する．閾値 *threshold* は 0.005 としている．傾きが閾値を超えると，サーバ接続数を確定する．しかし，傾きが閾値を越えた時点のサーバ接続数は，応答時間の標準偏差が大きくなっているので負荷が高い状態となっている．そのため，サーバ接続数を回帰直線の傾きが閾値を越える前の状態まで戻す必要がある．ここでは，回帰直線を計算するためのウィンドウ中央にあたるサーバ接続数に戻して，安定状態となる（20 行目）．

現在のサーバ接続数が適切なサーバ接続数に比べて大きく，サーバ接続数を現在の値に比べて小さな値に設定しなければならない場合がある．この場合，逆の動作となる．$dir = DESCEND$ の場合である．サーバ接続数を段階的に小さくし，傾きが閾値以下となったところにサーバ接続数を設定する．

3.3 サーバ接続数の再設定

サーバ接続数が一度適切な値に決まった後，ワークロードが突然大きく変化し，適切なサーバ接続数の値が変わることがある．そのため，ワークロードが変わるたびにサーバ接続数の再設定を行う．

ワークロードの変化の検出も，回帰直線の傾きを利用する．ワークロードが突然変化すると，応答時

```
 1: void on_stable(void) {
 2:   while (true) {
 3:     // 回帰直線の傾きの計算 (3.2 節と同様)
 4:     x = now();
 5:     y = get_sd();
 6:     grad = regression(win, x, y);
 7:     // 再設定開始の判定
 8:     if (abs(grad) > threshold) {
 9:       if (grad > 0) mc_control(DESCEND);
10:       else          mc_control(ASCEND);
11:     }
12:     reset_sd(); // 標準偏差の計算を初期化
13:     sleep(1000);
14:   }
15: }
```

図 5: サーバ接続数再設定のための疑似コード

間の標準偏差もこれまでの値に比べて大きく変化し，回帰直線の傾きも一時的に変化する．本機構はこの一時的な変化を検出し，サーバ接続数の再設定を行う．

図 5 に疑似コードを示す．まず，3.2 節と同様に回帰直線の傾きを計算する．回帰直線の傾きの絶対値が，3.2 節で用いた閾値を超えると，再設定を開始する．適切なサーバ接続数が現在の値に比べて大きいか小さいかは，回帰直線の傾きの正負で判断することができる．傾きが正の場合は，応答時間の標準偏差が急に大きくなったので，サーバ接続数を小さくするように再設定する．逆に，傾きが負の場合は，応答時間の標準偏差が小さくなったので，サーバ接続数を大きくするように設定する．

4 実装

提案機構を Linux 上の Apache 2.0.54 を対象に実装を行った．サーバ接続数の計算に必要な応答時間の計測は，Apache が呼び出す `poll()` システムコール間の経過時間を計測することで行う．Apache では，各クライアントからの接続ごとにサーバ上のプロセスを割り当て，各プロセスは I/O 入出力を監視するシステムコール `poll()` により，クライアントからのリクエストの受信を監視している．クライアントからリクエストが送信されると，`poll()` から Apache に処理が戻り，その要求処理を行い，再び `poll()` により次のリクエストを待つ．したがって，`poll()` 間の経過時間を計測すれば，サーバの応答時間を計測することができる．なお，時間の計測には Intel 製 CPU が提供する `rdtsc` 命令を用いた．`rdtsc` 命令は計算機の起動時刻からの経過時間をクロック単位で取得することができる．

サーバ接続数の設定は，`fork()`，`exit()` を書き換えることで実現できる．現在のプロセス数をカウントし，現在のプロセス数が提案機構で計算したサーバ接続数を超えそうな場合には，実際の `fork()` システムコールを呼び出さず，`fork()` が `EAGAIN` エラーを返したように模倣する．Apache はそのまま動作を続け，10 秒後に再試行を行うため，問題はない．

本機構は環境変数 `LD_PRELOAD` の機能を利用し，実行時に OS とサーバ間に挿入するライブラリとして実装されている．ユーザが作成したライブラリを環境変数 `LD_PRELOAD` に指定しておくと，OS やアプリケーションを改変することなしに，標準ライブラリ関数（とそれに対応するシステムコール）への呼出しをフックし，置き換えることができる．そのため，本実装は Apache と Linux を変更することなしに，利用可能である．

5 実験

提案機構を用いた自動設定の結果が，試行錯誤で求めた手動設定の結果に近いことを示すため，実験を行った．異なる 2 つのワークロードを与え，それぞれ適切にサーバ接続数を設定することを確認した．また，ワークロードを突然変化させ，適切に再設定されることを確認した．

5.1 実験環境

サーバ計算機は CPU が Pentium4 2.8GHz，主記憶 512MB，SCSI 接続 7200 回転の HDD，33MHz，32 ビット PCI バスの PC を用いた．クライアント計算機はサーバ計算機と同じ構成の PC を 16 台用いた．サーバ計算機とクライアント計算機は 1000 Base-T で 1 台のスイッチに接続されている．

ウェブサーバは Apache 2.0.54 を用いた．Apache のコンパイルでは，デフォルト設定であるプロセスのみを使用する `prefork MPM` を用いた．Apache のパラメータは `KeepAliveTimeout` を 1 秒とし，サーバ接続数 `MaxClients` を除くその他すべてのパラメータはデフォルト値を用いた．OS は Linux 2.4.20 を使用しており，カーネルのパラメータは変更していない．

表 1: 実験で用いる負荷

負荷	説明	ファイル・サイズ分布				総ファイル・サイズ
		≤1KB	≤10KB	≤100KB	≤1MB	
(a) SPECWeb 標準	SPECWeb99 標準の負荷	35%	50%	14%	1%	3.6GB
(b) SPECWeb 大	SPECWeb99 標準を平均ファイル・サイズが大きくなるように修正したもの	1%	14%	50%	35%	3.6GB

5.2 ワークロード

ワークロード生成は SPECWeb99[7] にユーザ思考時間を導入したものを用いた．SPECWeb99 はウェブサーバの標準的なベンチマークである．しかし，サーバの性能測定を目的とするため，思考期間に相当するクライアントの振舞いは考慮されていない．思考期間を模倣するため，Pareto 分布にしたがって要求生成するようにした．一般に，思考期間は Pareto 関数で近似するのがよいことが知られている [8]．Pareto 関数の係数は，文献 [8] の値を用いた．なお，リクエストが連続する場合には SPECWeb99 の規約に従う．今回，動的ページに対する要求は行わず，静的ページに対する要求のみとする．

実験では，表 1 の 2 つの負荷を用いる．最初の **(a) SPECWeb 標準**は，SPECWeb99 の規約で定められている標準設定である．**(b) SPECWeb 大**は，画像を多く含むサイトなどを想定した，平均ファイル・サイズが大きくなるように修正した負荷である．

5.3 手動設定と自動設定の比較

図 6 に，手動設定の結果と提案機構による自動設定の結果を示す．2 つのワークロードを与えた場合の，スループットと平均応答時間について示す．手動設定の結果は，SPECWeb 標準ではサーバ接続数 800，SPECWeb 大では 200 の結果である．

図 6 をみると，提案機構による結果は手動設定の結果に十分近い．本機構はサーバ接続数をそれぞれ 795，175 に設定しており，サーバ接続数からみた差は −6.3%，−12.5%である．ワークロードごとに個別にみれば，SPECWeb 標準ではスループット，平均応答時間の差はそれぞれ +2.4%，−1.5%である．SPECWeb 大ではスループット，平均応答時間の差はそれぞれ −14.1%，−7.2%である．

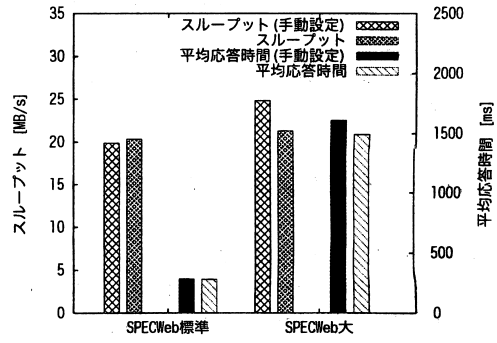


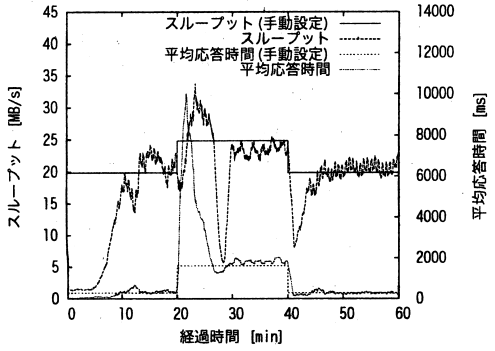
図 6: 手動設定と自動設定の比較

5.4 ワークロードの変化に対する適応

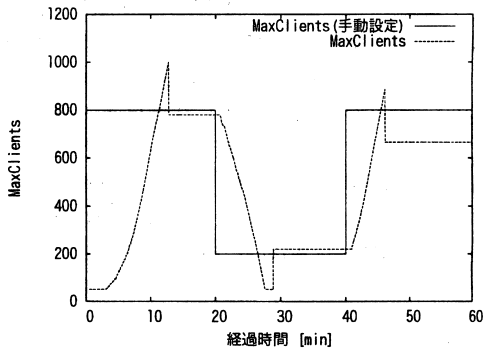
図 7 に本機構が時間経過にしたがってサーバ接続数を自動設定する様子を示す．途中，ワークロードを 2 度変化させており，その時刻は 20 分，40 分である．ワークロードは 0–20 分は SPECWeb 標準，20–40 分は SPECWeb 大を用いており，40–60 分は再び SPECWeb 標準に戻している．参考のため，それぞれのワークロードでの手動設定の最適値を図中に示す．

図 7 を見ると，ワークロードの変化に応じて，それぞれ適切にサーバ接続数が自動設定されていることが分かる．自動設定の結果は手動設定の結果に近く，約 10 分程度で適応している．

時間経過にしたがって順にみていくと，3 分間のウォームアップ時間の後，3 分後に自動設定を開始し，サーバ接続数を徐々に増加させる．12 分後に回帰直線の傾きが一定ではなくなったことを検出し，サーバ接続数を傾きが大きくなる直前のものに戻す．そして，自動設定を完了し，安定状態となる．20 分にワークロードが突然変化すると，本機構は直ちにこれを検出し，サーバ接続数を徐々に減少させる．途中，27–28 分でサーバ接続数が一定となっているが，これはサーバ接続数の下限を 50 としているためである．約 29 分にサーバは再設定を完了し，安定状態となる．40 分にワークロードは再び変化し，



(a) スループット・平均応答時間



(b) サーバ接続数

図 7: ワークロードの変化に対する適応

本機構は約 46 分に再設定を完了する．

6 関連研究

6.1 ウェブサーバのサーバ接続数自動設定

ウェブサーバのサーバ接続数を対象とした研究として，[1, 2, 9] がある．Liu ら [1] は，最適なサーバ接続数付近で平均応答時間が下に凸となることを報告し，平均応答時間が下に凸となるサーバ接続数をニュートン法とファジイ制御を用いて探索し，Apache のサーバ接続数を設定している．また，ボトルネックとなる CPU 使用率をもとに，サーバ接続数を設定する機構も同時に評価している．これに対して，本論文では応答時間の分散をもとにサーバ接続数を設定する機構を提案している．文献 [1] の実験でも，応答時間の標準偏差が最適なサーバ接続数付近で小さくなっており，本機構もこれらの手法と同様に有効であることが期待される．

Diao ら [2] は，Apache を対象に CPU とメモリで構成されるモデルを作成し，サーバ接続数と keep-alive 時間の 2 つのパラメータを設定している．CPU とメモリの使用量がパラメータの線形結合で求められると仮定し，人工的なワークロードをサーバに与え，モデルにあらわれる係数を求めている．CPU とメモリの消費量はワークロードに依存して決まると考えられ，負荷が変わるとどの程度正しく動作するかについては議論されていない．

Menascé ら [9] はサーバを待ち行列でモデル化し，管理者が与えた QoS の性能目標をもとにサーバ接続数の設定を行う．モデルにさまざまなサーバ接続数をパラメータとして与え，モデルが予測したスループットと応答時間が最もよいものを選択する．この手法はサーバ接続数を設定するために，実際のサーバ上でサーバ接続数を変え，その効果を測定する必要がない反面，パラメータ設定の結果はモデルの精度に大きく依存する．そのため，モデルの精度向上を目指した研究が広く行われている [10]．

6.2 汎用的なパラメータ設定手法

サーバ接続数のみを対象としない汎用的なパラメータ設定手法として，Active Harmony[3]，Smart Hill-climbing[4] などがある．これらの手法はシンプレックス法などの発見的探索手法を用いて，さまざまなパラメータの組み合わせを試行する．試行の結果，最もよいパラメータの組み合わせを選択する．

このアプローチは個別のパラメータに対する知識を全く必要としない，多数のパラメータを同時に設定できるという反面，発見的手法であるため，極小解に陥ったり，解が得られないことがある．

本提案機構もサーバが高負荷になると応答時間の分散が大きくなるという知識以外，特定のワークロードやサーバの知識を必要としない．サーバの接続数以外でも高負荷時に応答時間の分散が大きくなるパラメータであれば，適用可能であると期待できる．

6.3 アドミッションコントロール

サーバに接続するクライアント数を制限するということに関連して，ウェブサーバを対象としたアド

ミッションコントロールに関する研究が数多く行われている [11, 12, 13, 14, 15, 16]. アドミッションコントロールでは，クライアントからの接続の受け入れの可否を，あらかじめ与えられた QoS の性能目標をもとに決定する．外部から与えられた性能目標にもとづいて制御を行うため，サーバの性能やワークロードの変化に応じて，サーバ接続数を自動的に決定する本研究とは本質的に異なる．

7 まとめ

本論文では，応答時間の分散を利用したウェブサーバの接続数の自動設定手法を提案した．サーバは高負荷になると，応答時間の分散が急激に大きくなるという現象を示す．本機構は応答時間の分散が大きくなったことを検出し，サーバ接続数を自動設定する．

実験では，2 つの異なるワークロードに対して，応答時間の増加を抑えながら，スループットを最大にするサーバ接続数に自動設定することができた．また，ワークロードの突然の変化に対しても適応することができた．

謝辞

本研究の一部は，科学技術振興機構 CREST「ディペンダブル情報処理基盤」による支援を受けている．

参考文献

[1] Liu, X., Sha, L., Diao, Y., Froehlich, S., Hellerstein, J. L. and Parekh, S. S.: Online Response Time Optimization of Apache Web Server, *Proc. 11th IEEE Int'l Workshop on QoS*, pp. 461–478 (2003).

[2] Diao, Y., Gandhi, N., Hellerstein, J., Parekh, S. and Tilbury, D.: Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics With Application to the Apache Web Server, *Proc. 7th IEEE/IFIP Symp. on Integrated Network Management*, pp. 219–234 (2001).

[3] Chung, I.-H. and Hollingsworth, J. K.: Automated Cluster-Based Web Service Performance Tuning, *Proc. 13th IEEE Int'l Symp. on High Performance Distributed Computing*, pp. 36–44 (2004).

[4] Xi, B., Liu, Z., Raghavachari, M., Xia, C. H. and Zhang, L.: A Smart Hill-Climbing Algorithm for Application Server Configuration, *Proc. WWW Conf. 2004*, pp. 287–296 (2004).

[5] The Apache Software Foundation: Apache HTTP Server (1995). `http://www.apache.org/`.

[6] 杉木章義, 河野健二, 岩崎英哉: リクエスト間隔を考慮したウェブサーバの keep-alive 時間の自動設定, 日本ソフトウェア科学会 第 22 回大会論文集 (2005).

[7] Standard Performance Evaluation Corporation: The SPECweb99 benchmark (1999). `http://www.spec.org/osg/web99/`.

[8] Barford, P. and Crovella, M.: Generating Representative Web Workloads for Network and Server Performance Evaluation, *Proc. ACM SIGMETRICS'98*, pp. 151–160 (1998).

[9] Menascé, D. A. and Bennani, M. N.: On the Use of Performance Models to Design Self-Managing Computer Systems, *Proc. 2003 Computer Measurement Group Conf.* (2003).

[10] Robertsson, A., Wittenmark, B., Kihl, M., Andersson, M.: Design and Evaluation of Load Control in Web Server Systems, *Proc. 2004 American Control Conf.*, pp. 1980–1985 (2004).

[11] Chen, X., Chen, H. and Mohapatra, P.: An Admission Control Scheme for Predictable Server Response Time for Web Access, *Proc. WWW Conf. 2001*, pp. 545–554 (2001).

[12] Cherkasova, L. and Phaal, P.: Session-Based Admission Control: A Mechanism for Improving the Performance of an Overloaded Web Server, *Technical Report HPL-98-119* (1998).

[13] Abdelzaher, T. F., Shin, K. G. and Bhatti, N.: Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 13, No. 1, pp. 80–96 (2002).

[14] Elnikety, S., Nahum, E., Tracey, J. and Zwaenepoel, W.: A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites, *Proc. WWW Conf. 2004*, pp. 276–286 (2004).

[15] Kamra, A., Misra, V. and Nahum, E. M.: Yaksha: A Self-Tuning Controller for Managing the Performance of 3-Tiered Web Sites, *Proc. Int'l Workshop on QoS*, pp. 47–56 (2004).

[16] Karlsson, M., Zhu, X. and Karamanolis, C.: An Adaptive Optimal Controller for Non-Intrusive Performance Differentiation in Computing Services, *Proc. Int'l Conf. on Control and Automation 2005*, pp. 709–714 (2005).