2008－OS－108 (5)
2008／4／23

実験計画法を利用したウェブサーバの主要なパラメータ選択手法

杉木 章義† 加藤 和彦‡†

† 科学技術振興機構 CREST
‡ 筑波大学大学院 システム情報工学研究科
E-mail: sugiki@osss.cs.tsukuba.ac.jp, kato@cs.tsukuba.ac.jp

要 旨

近年，ウェブサーバの管理コストが主要な問題の一つとして認識されており，効率的な管理手法が求められている．特に，パラメータ設定作業は多大な経験と時間を必要とし，難易度の高いタスクとなっている．最新のウェブサーバには数十個のパラメータがあり，これらを人手で設定するのは難しい．また，計算機による自動設定の場合であっても，全てのパラメータを対象としたのでは探索空間が膨大となり，現実的な時間で設定完了するのは困難である．本論文では，実験計画法を利用した主要なパラメータ選択手法を提案する．本手法を用いることで少ない実験回数で多数のパラメータを同時に調査し，少数の性能に大きな影響を与えるパラメータに絞り込むことができる．また，性能測定に含まれる誤差を統計的に管理し，効果を判定することができる．Apache ウェブサーバに対して SPECweb2005 の 3 種類のワークロードを与えた場合について調査を行い，32 回の実験で試行した 24 個のパラメータのうち，6–9 個に有意に効果があることが確認された．

Finding Major Parameters of a Web Server with Design of Experiments

Akiyoshi Sugiki† Kazuhiko Kato‡†

† CREST, Japan Science and Technology Agency
‡ Department of Computer Science, University of Tsukuba
E-mail: sugiki@osss.cs.tsukuba.ac.jp, kato@cs.tsukuba.ac.jp

Abstract

The performance of web servers is crucial for today's continual business process. Therefore, the proper management of performance is a major issue in web servers. Current web servers have a large number of parameters, making it impractical to tune all of them manually. In this paper, we present a technique based on design of experiments for finding parameters that influence server performance the most. We call these parameters major parameters. Even with the classical two-level orthogonal design, it can successfully find major parameters with a small number of experiments. We applied this technique to Apache web server under three workloads of SPECweb2005. Experimental results show that our technique can select 6–9 major parameters out of 24 parameters with 32 experiments.

1 はじめに

近年，ウェブサーバの管理コストが主要な問題の一つとして認識されており [1, 2]，効率的な管理手法が求められている．特に，サーバのパラメータ設定作業は多大な経験と時間を必要とし，管理者にとって難易度の高いタスクとなっている [3, 4]．最新のウェブサーバには数十個のパラメータがあり，適切にパラメータ設定を行うかどうかによって，同一ハードウェアでもスループットに数倍以上の開きがある [5]．

現状のパラメータ設定作業は，多くの場合，管理者の経験や勘に頼っている．また，繰り返し実験によってパラメータ設定が行われる場合であっても，アドホックな手法であることが多い．よく行われる手法は，他の全てのパラメータの値を固定し，一つのパラメータの値を変化させて効果を見る方法である [6]．しかしながら，この手法は探索を行う範囲が狭く，正しく効果を判定することができない．

計算機を利用し，探索によってパラメータ設定を行う手法 [3, 4, 7, 8] が提案されているが，全てのパラメータを設定対象としたのでは探索空間が膨大となり，現実的な時間で設定が完了しない．例えば，2 つの値をとるパラメータが 100 個存在した場合，その探索空間は 2^{100} となる．そのため，性能効果が高いと思われる主要なパラメータを選択し，設定することになるが．多くの場合，この選択作業は管理者の経験と勘に大きく依存している．

本論文では，実験計画法 [9] を利用したウェブサーバの主要なパラメータ選択手法を提案する．実験計画法はあらかじめ実験を計画的に構成し，その計画に基づいて実験結果を収集することによって，少ない実験回数で最大の効果を目指す手法である．しかも，その解析結果には統計的な裏付けがあり，誤差を管理しながら効果を判定することができる．本論文では，最初の試みとして古典的な 2 水準の直交表を利用し，ウェブサーバの多数のパラメータの中から，少数の効果のある主要なパラメータを選び出すスクリーニング計画を適用する．本論文の手法は，あくまで主要なパラメータを選び出すだけであり，図 1 に示すように具体的なパラメータ値の決定は探索によって後段階で行われる．

本論文の貢献は以下の 2 つである．

- **実験計画法のウェブサーバへの適用**：本論文は，実験計画法をウェブサーバのパラメータのスクリーニング計画に適用し，その利点と限界を明らかにする．ウェブサーバの性能測定では多くのばらつきが含まれ，一般的に長時間にわたって性能を測定することが推奨されている [10]．一方，実験計画法では誤差を統計的に管理しながら効果を判定することができる．本論文は実験計画法をパラメータ選択に適用する際のノウハウについて明らかにし，一方で計算機での利用を想定せず開発された実験計画法の今後の改良の方向性を示す．
- **標準的なウェブサーバでの効果の確認**：本論文では，広く利用されている Apache ウェブサーバ [11] に対して．一般的に設定した方がよいとされているパラメータに対して検証を行った．本論文の実験はある特定環境での結果であるが，標準的なサーバ構成，ベンチマーク SPECweb2005[12] を使用しており，多くの環境で同じような効果が現れることが期待される．32 回の実験で 24 個のパラメータを調査し，6–9 個に有意に効果があることが確認された．

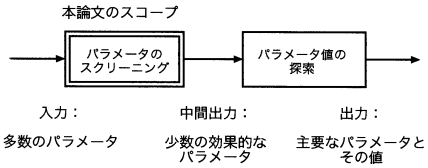

図 1: パラメータ設定のプロセス

2 関連研究

実験計画法のサーバのパラメータ設定への適用は少数の論文で行われ始めているが，いずれも初期段階にとどまり，今後もさまざまな試みが必要である．これらの研究とは，実験計画法という非常に大きな枠組みでは共通しているものの，細部では大きく異なる．特に，本論文はばらつきの管理を重視している点で他の多くの論文とは異なる．

Chow[13] は J2EE サーバのパラメータ設定に実験計画法を利用することを提案している．しかしながら，全てのパラメータ値の組み合わせに対して試行する完全実施要因計画を行っており，本論文の少

ない実験回数で効果のあるパラメータを見つけるスクリーニング計画とは異なっている．また具体的なパラメータ値の推定まで目指しているが，古典的な2水準での推定では限界があることを示している．

Debnath ら [14] は，データベースサーバに対して実験計画法の一手法である Plackett-Burman 法を適用し，性能効果に基づくパラメータのランク付けを行っている．この手法は実験回数の削減を行っているものの，性能測定のばらつきは重視しておらず，分散分析ではなくソートを利用して順位付けを行っている．

Xi ら [4] は Smart Hill-climbing で実験計画法の超ラテン方格を利用している．超ラテン方格を利用し，探索空間を均一に試行することで，局所探索を行う範囲をおおまかに決めている．しかしながら，スループットを直接比較して探索範囲を決めており，誤差については重視していない．

Chung は [6] は Active Harmony で同時に一つのパラメータ値を変化させることでパラメータの優先度付けを行っている．しかし，この手法は実験計画法では好ましくない手法として紹介されており，探索されない空間が大きい．

小谷，須田 [15] は数値計算のパラメータ設定に実験計画法を利用することを目指している．本研究とは対象とするアプリケーションが異なり，異なる分析方法を使用している．

この他にも，実験計画法はハードウェア設計 [16] やソフトウェア工学 [17] などの情報工学分野で利用されている．しかしながら，他の工学分野と比べれば利用は限定的でり，本論文ではウェブサーバの主要なパラメータ選択に適用し，その可能性を探る．

3 実験計画法

3.1 概要

実験計画法は，実験を計画的に構成し，その計画に基づいて実験を行うことで最小の実験回数で最大限の効果を目指す手法である．ウェブサーバのパラメータ設定に適用する場合には，次のような利点がある．

- **効率的な実験回数の削減**：管理者の経験や勘に頼らず，またアドホックな手法に頼ることなく効率的に情報収集することができる．ウェブサーバには多数のパラメータがあり，1回の性能測定に時間がかかるため，実験回数の削減による効果が大きい．
- **測定誤差の管理**：ウェブサーバの性能測定結果には多数のばらつきが含まれるため [10]，一般的に長期間に渡ってデータを測定することが推奨されている．例えば，SPECweb2005[12] は30分間にわたり測定することを要求している．実験計画法はやみくもに実験時間の延長に頼ることなく，誤差を統計的に管理しながら，効果を判定することができる．

3.2 実験手順

実験計画法の具体的手順についてはさまざまな文献 [9] で解説されているが，本論文では関係する部分について説明する．実験計画法は主に，3.2.1 節の実験を計画する段階と 3.2.2 節の実験結果を解析する段階とに分けられる．

3.2.1 実験計画の作成

実験計画法の基本は，全てのパラメータ値の組み合わせに対して実験を行う完全実施要因計画である．伝統的に，実験計画法では一つのパラメータに対して2または3つの値を試行する2水準または3水準実験が好まれており，試行回数はパラメータ数が n 個の場合，2^n または 3^n となる．

完全実施要因計画は実験回数があまりに大きいため，大幅に削減した一部実施要因計画が広く用いられている．この計画を作成するためには，全てのパラメータ値の組み合わせが同数回ずつ出現するという直交性を保ったまま，実験回数を削減する．表1は，2水準3個のパラメータ A，B，C の場合の実験計画の例である．(a) は完全実施要因計画であり，(b) は一部実施要因計画である．パラメータの各列の “1” と “−1” はパラメータの水準を表し，実際には具体的なパラメータ値に置き換える．(a) と (b) を比べると，実験回数が8回から4回に削減されており，各パラメータの水準が同数回出現していることが分かる．実験計画の作成には，直交表と呼ばれるテンプレートがよく利用され，本論文でも直交表を利用する．直交表は $L8(2^7)$ などの名前がついており，L の横の数字は実験回数の8回，カッコの中

表 1: 実験計画の比較

(a) 完全実施要因計画

No.	A	B	C
1	-1	-1	-1
2	-1	-1	1
3	-1	1	-1
4	-1	1	1
5	1	-1	-1
6	1	-1	1
7	1	1	-1
8	1	1	1

(b) 一部実施要因計画

No.	A	B	C
1	-1	-1	-1
4	-1	1	1
6	1	-1	1
7	1	1	-1

の数字は2水準のパラメータが最大7個まで調査可能であることを示している．

一部実施要因計画は実験回数が削減できる反面，列同士の交絡関係が発生するという欠点がある．例えば，表1(b)では，パラメータCの効果とパラメータA，Bの交互作用の効果が交絡しており，両者どちらの効果であるか判別することはできない．交絡は実験回数を削減するほど増加し，どこまで交絡するかは，計画のresolutionに依存する．一部実施要因計画は実験回数が削減できる反面，効果が大きいと思われる要因同士が交絡しないように注意深く割り付けを行うなど工夫が必要となる．

3.2.2 ANOVAによる解析

実験計画では，分散分析（ANOVA: ANalysis Of VAriance）による解析がよく行われる．ここでは，パラメータA，Bの2個の場合について説明するが，一般の場合も同様である．実験計画法では，測定結果 y_{ij} を以下でモデル化する（i,j はパラメータの水準を表す）．

$$y_{ij} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ij} \quad (1)$$

μ はパラメータによらない成分であり，α_i，β_j はそれぞれのパラメータの値による効果を示す．また，$(\alpha\beta)_{ij}$ はパラメータ間の交互作用による効果を示す．ε_{ij} は測定の誤差を表し，正規分布 $N(0,\sigma^2)$ に従うとする．

式1は測定結果を各成分に分解したモデルとなっており，同様にばらつきも以下のように分解できることが知られている．

$$S_T = S_A + S_B + S_{A\times B} + S_E \quad (2)$$

S_T は全体の変動であり，S_A，S_B，$S_{A\times B}$ はパラメータの主効果，交互作用による変動を示す．S_E は誤差による変動である．これらは平方和と呼ばれ，例えば S_T は以下で計算できる（$\overline{y}$ は測定平均）．

$$S_T = \sum_i \sum_j (y_{ij} - \overline{y})^2 \quad (3)$$

実験計画法では，S_A，S_B，$S_{A\times B}$ が誤差変動 S_E に比べてどの程度大きいかを見ることで効果を判定する．例えば，パラメータAの効果であれば，

$$F = (S_A/\phi_A)/(S_E/\phi_E) \quad (4)$$

の F 値を見ることで判定する．ϕ_A，ϕ_E はパラメータA，誤差要因それぞれの自由度である．この F 値は F 分布に従うため，検定により効果を判定することができる．確率 p 値を計算し，有意水準0.05または0.01以下であれば，効果ありと判断する．

4 実験

実験計画法を利用してウェブサーバの主要なパラメータが選択可能であるか実験を行う．まず，一部実施要因計画によって実験回数を削減しても，完全実施の場合と同様に主要なパラメータが選択可能であることを確認する．次に，実際に標準ベンチマークでのスクリーニング計画に適用し，どのパラメータが性能向上に有効であるか確認する．

4.1 実験環境

実験には，デュアルXeon 3.60GHz CPU，メモリ2GB，36GBのSCSIディスクで構成された計算機10台を使用した．これらの計算機から，ウェブサーバに1台，バックエンドエミュレータに1台，クライアントに8台を割り当てた．全ての計算機は1000BASE–Tで単一のスイッチに接続されている．オペレーティングシステムはFedora 8（Linux 2.6.23）を使用した．ウェブサーバにはApache 2.2.6[11]を使用し，キャッシュモジュールである `mod_cache` を有効にした．また，PHP 4.4.8，Fast CGI 2.4.1（`mod_fastcgi` 2.4.6），PHPのキャッシュモジュールであるAPC 3.0.15を使用した．

ワークロードには標準ベンチマークの一つであるSPECweb2005 1.00[12]を使用した．Banking，Ecommerce，Suppportの3種類のワークロードを使用し，PHP版を使用した．SPECweb2005の規約では30分間にわたり測定することになっているが，

今回は主要なパラメータ選択が目的であり，誤差は分散分析により管理できることから測定時間は1分とした．今回は，SPECweb2005のスコア値を性能指標として最大化する．スコア値は応答時間が基準を満たす最大の接続数に対応し，本論文では二分探索でそのパラメータ値の組み合わせでの最大スコア値を探す．そのため，1回のパラメータの組み合わせでのスコア値の測定に30分程度を要する．実験に使用した計算機のディスク容量の制約から，最大スコアは1000で打ち切りとなる．

直交計画の生成や分析には，統計解析ソフトウェアJMP 7[18]のスクリーニング計画を使用した．

4.2 対象パラメータ

本論文で試行を行うパラメータを表2に示す．広範囲のパラメータを対象とし，ウェブサーバ上のApache，APCのパラメータ（Apache，APC），バックエンド・エミュレータ上のApacheのパラメータ（besim），Linuxのカーネル・パラメータ（Proc）を試行する．各パラメータの“−1”，“1”はパラメータの水準を表し，全て2水準で実験を行う．

Apacheのパラメータは，公式のチューニング・ノート[19]などで設定が推奨されているパラメータの中から選択した．MCacheSize，shm_sizeはそれぞれmod_cache，APCのキャッシュサイズである（単位はMB）．Besimは MaxClients と KeepAliveTimeout のみ対象とする．各パラメータの水準値は，設定可能だと思われる範囲の中からおおまかに最小値と最大値付近となるように選択した．

カーネルのパラメータはSPECweb2005のサイト上の報告から，頻繁に変更されているものを選択した．低水準側はLinuxのデフォルト値とし，高水準側はSPECweb2005のサイト上で多く報告されている値とした．カーネルのパラメータ，特にTCP/IP関連のパラメータは相互の計算機同士が関係すると考えられることから，全ての計算機を同時に同じ値へと変更する．この他のパラメータについては特に指定がない限り，デフォルト値を使用する．

4.3 実験結果：完全実施との比較

まず，本論文で使用する一部実施要因計画と完全実施要因計画と比較し，実験回数を削減しても主要なパラメータを選択できることを示す．

表2: 対象パラメータ

パラメータ	-1	+1
Apache.AllowOverride	Off	On
Apache.Modules	最小	全て
Apache.HostnameLookups	Off	On
Apache.Logging	Off	On
Apache.MaxClients	150	700
Apache.KeepAliveTimeout	1	15
Apache.Timeout	30	300
Apache.MaxRequestsPerChild	0	10
Apache.MCacheSize	0	512
APC.shm_size	0	512
Besim.MaxClients	150	700
Besim.KeepAliveTimeout	1	15
Proc.rp_filter	Off	On
Proc.tcp_timestamps	Off	On
Proc.tcp_max_tw_buckets	180000	3000000
Proc.rmem	131071	10000000
Proc.wmem	135168	10000000
Proc.optmem_max	20480	10000000
Proc.tcp_rmem	4096	30000000
	87380	30000000
	174760	30000000
Proc.tcp_wmem	4096	30000000
	16384	30000000
	131072	30000000
Proc.somaxconn	128	40960
Proc.netdev_max_backlog	300	300000
Proc.min_free_kbytes	16382	200000

表3に，同じパラメータの組に対して完全実施要因計画と一部実施要因計画 $L8(2^7)$ を実施した結果を示す．ワークロードはBankingを使用している．Apache，BesimそれぞれのMaxClientsとKeepAliveTimeoutを対象とし，計4個で実験を行う．交互作用は効果が大きいと思われるApache内のMaxClientsとKeepAliveTimeout間，ApacheとBesim間のMaxClientsを取り上げる．この2つ以外の交互作用については，残差にプーリングする．

表3上で＊で示されているパラメータは効果ありと判定されたパラメータであり，完全実施，一部実施共に同一のパラメータを選択している．効果があるとされているのは，ApacheのMaxClients，KeepAliveTimeoutと，この2者間の交互作用である．一方，Besimはほとんど性能に影響しない．完全実施の方が実験回数が多いため p 値が小さく，推定精度がわずかによいが，選択されるパラメータという点は全く変わらない．

図2に表3の解析で使用したApacheのMaxClientsとKeepAliveTimeoutを変化させた場合のSPECwebのスコア値を示す．図2を見るとMaxClientsは大きい方がよく，KeepAliveTimeoutは小さい方がよいことが分かる．この2つのパラメータ間には交互

表 3: 完全実施要因計画との比較

	完全実施要因計画				一部実施要因計画			
	自由度	平方和	F	p	自由度	平方和	F	p
Apache.MaxClients (MC)	1	1438200.6	380001.6	$< .0001*$	1	720600.13	640533.4	0.0008*
Apache.KeepAliveTimeout (KA)	1	387817.6	104269.2	$< .0001*$	1	194376.13	172778.8	0.0015*
Besim.MaxClients (MC)	1	3.1	0.8092	0.3918	1	0.13	0.1111	0.7952
Besim.KeepAliveTimeout (KA)	1	7.6	1.9982	0.1911	1	3.13	2.7778	0.3440
Apache.MC * Apache.KA	1	83376.6	22029.77	$< .0001*$	1	37950.13	33733.44	0.0035*
Apache.MC * Besim.MC	1	0.065	0.0165	0.9006	1	0.13	0.1111	0.7952
誤差	9	34.1			1	1.13		

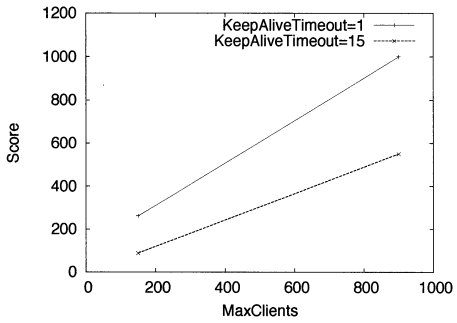

図 2: パラメータの交互作用

作用があると判定されており，KeepAliveTimeout を小さくした方が，MaxClients を変化させた場合のスコア値の増加が大きくなる．

4.4 実験結果：スクリーニング

Apache ウェブサーバに対して，SPECweb2005 のワークロードを与えた場合に，スクリーニング計画を適用した結果を表 4 に示す．ここでは，L32(2^{31}) の計画を使用し，3 種類のワークロード全てについて調査している．ここでは主効果のみ取り上げ，交互作用については調査しないものとする．表の読み方は表 3 とほぼ同様であり，最大化と書かれた列は効果があるパラメータについてスコア値を最大化するパラメータの値を示している．表 5 にそれぞれのワークロードでの最小スコアと最大スコアを示す．

表 4 を見ると，MaxClients，KeepAliveTimeout の効果がどのワークロードでも非常に大きく，続いて Logging，Timeout，MaxRequestsPerChild に効果があることが分かる．MaxClients と KeepAlive Timeout の効果が高いことは以前から知られており，我々が以前行っていた研究 [5] の結果と一致する．リクエストのタイムアウト処理に関係する Timeout は意外にも性能に関係し，MaxRequestsPerChild はプロセスの生成や破棄のコストが関係することから当然といえる．さらに，Banking と Ecommerce では，rp_filter を効果ありとして検出している．rp_filter は送信元の IP アドレスの詐称を防ぐ機構であり，受信ごとに処理を行うことから性能に影響したと思われる．ただし，ベンチマークでは有効であるが，セキュリティの面から実システムでは解除しない方が望ましい．Support では F 値が 2 を超えるパラメータが複数存在することから，効果の小さいパラメータの影響を誤差にまとめる残差のプーリングをさらに行った．追加で rp_filter，somaxconn，net_dev_max_backlog，min_free_kbytes に効果があることが分かった．

SPECweb2005 の 3 種類のワークロードの結果を比較すると，どのパラメータに効果があるかという点ではワークロードの変化によってほとんど変わらないことが分かる．一方で，パラメータの具体的な値はワークロードによって変わる可能性がある．とはいいながらも，Support は他の 2 種類のワークロードと少し異なる傾向を示しており，追加で 3 つのパラメータに効果がある．Support はファイルのダウンロードが主体であり，トランザクション主体の他の 2 種類のワークロードとは異なる．これは，実験中の我々の経験と一致する．また，今回検出できなかったからといって効果が必ずしもない訳ではない．誤差と効果が判別できないだけであり，さらに実験回数を追加するなどによって，確かめることができる．

表 4: SPECweb2005 での実施結果

パラメータ	自由度	Banking		Ecommerce		Support		最大化
		F	p	F	p	F	p	
Apache.AllowOverride	1	0.2038	0.6653	0.0842	0.7801	0.7414	0.4177	
Apache.Modules	1	0.0639	0.8077	0.1533	0.7070	0.5923	0.4667	
Apache.HostnameLookups	1	0.0609	0.8122	0.1035	0.7571	0.0976	0.7638	
Apache.Logging	1	6.1050	0.0428*	10.9156	0.0131*	511.1325	< .0001*	Off
Apache.MaxClients	1	110.5082	< .0001*	150.8482	< .0001*	3922.260	< .0001*	+
Apache.KeepAliveTimeout	1	39.5631	0.0004*	63.9317	< .0001*	1386.707	< .0001*	−
Besim.MaxClients	1	0.0359	0.8550	0.1373	0.7219	0.0168	0.9006	
Besim.KeepAliveTimeout	1	0.0565	0.8189	0.1819	0.6825	0.0795	0.7861	
Apache.MCacheSize	1	0.0483	0.8324	0.1174	0.7419	0.4195	0.5378	
APC.shm_size	1	0.0537	0.8234	0.1272	0.7319	0.3094	0.5954	
Apache.Timeout	1	6.7143	0.0359*	7.6478	0.0279*	8.0782	0.0250*	+
Apache.MaxRequestsPerChild	1	7.1581	0.0317*	7.8816	0.0262*	9.7128	0.0169*	Off
Proc.rp_filter	1	6.3301	0.0400*	6.3541	0.0398*	4.8495	0.0635	Off
Proc.tcp_timestamps	1	0.1651	0.6967	0.0968	0.7647	0.7414	0.4177	
Proc.tcp_max_tw_buckets	1	0.1461	0.7136	0.0968	0.7647	0.5022	0.5015	
Proc.rmem	1	0.1904	0.6757	0.1702	0.6923	0.0488	0.8314	
Proc.wmem	1	0.1904	0.6757	0.1479	0.7120	0.0363	0.8543	
Proc.optmem_max	1	0.1261	0.7330	0.1879	0.6777	0.0131	0.9122	
Proc.tcp_rmem	1	0.1198	0.7394	0.2066	0.6632	0.1394	0.7199	
Proc.tcp_wmem	1	0.1461	0.7136	0.1479	0.7120	0.2161	0.6562	
Proc.tcp_mem	1	0.1984	0.6695	0.1533	0.7070	0.1176	0.7417	
Proc.somaxconn	1	0.1878	0.6778	0.0925	0.7698	3.3166	0.1114	+
Proc.netdev_max_backlog	1	0.1904	0.6757	0.1150	0.7445	3.0983	0.1218	+
Proc.min_free_kbytes	1	0.1801	0.6841	0.1057	0.7546	2.7848	0.1391	−

表 5: パラメータ値によるスコアの変化

	最小スコア	最大スコア
Banking	86	999
Ecommerce	82	999
Support	86	991

5 議論

実験計画法はウェブサーバのパラメータ設定作業を大きく改善する反面，古典的手法とサーバ環境との間にはギャップがある．まず，実験計画法では 2 水準または 3 水準での実験が好まれる．しかも，3 水準は実験回数が増加し，パラメータの交互作用の計算が複雑になることから 2 水準の方が好まれることが多い．一方で，サーバ環境では工業プロセスに比べればまだ実験回数を増加させやすく，計算機による解析が容易であることから，多水準実験の方がよいと思われる．例えば，MaxClients では値を大きくすることによって一度スループットが増加し，その後過負荷となり低下すると考えられることから 2 次以上で近似した方がよいと思われる．

また，サーバ環境のパラメータ設定では試行を行うパラメータ値自体を決めるのが難しい．メモリサイズを設定するパラメータであればハードウェアの制約から決めることができるが，MaxClients などの物理的な数値から離れたパラメータであれば，経験がなければ推測することはできない [20]．試行するパラメータ値によってパラメータの効果が十分出ないこともあり，何らかの工夫が必要である．

6 まとめと今後の予定

本論文では，実験計画法を利用したウェブサーバの主要なパラメータ選択手法を提案した．最新のウェブサーバでは数十から数百のパラメータがあり，現実的な時間で設定を完了するために，この中から性能に大きな影響を与える少数のパラメータを選び出し，調整を行うことが必要である．本論文では最初の試みとして，古典的な直交計画をウェブサーバの主要なパラメータ選択に適用した．本手法を用いると，少ない実験回数で多数のパラメータの効果を同時に調査することができる．

Apache ウェブサーバに SPECweb2005 の 3 つのワークロードを与えた場合で実験を行ったところ，32 回の実験で 24 個のパラメータのうち 6–9 個に有意に効果があると判定された．

今後は，実験計画法を基盤としてさまざまな方向に展開することが考えられる．まず，アプリケーションサーバなど他のサーバに対しても適用し，有効性を検証する．実験計画法は汎用性の高い手法であることから，同じように有効であることが期待される．また，今回は既存のサーバに対して適用したが，堅牢なサーバの設計手法としても活用することができる．最後に，古典的な実験計画法は工業プロセスの改善など頻繁に実験を行えない状況を想定しており，現在のサーバ環境との間に大きな隔たりがある．今後は，クラスタ環境を利用したパラメータ設定手法など，現在のサーバ環境に応じた手法へと改良していく．

謝辞

本研究は科学技術振興機構 CREST「自律連合型基盤システムの構築」の支援を受けている．

参考文献

[1] Patterson, D., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., Oppenheimer, D., Sastry, N., Tetzlaff, W., Traupman, J. and Treuhaft, N.: Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies, *UC Berkeley Computer Science Technical Report UCB//CSD-02-1175* (2002).

[2] Ganek, A. G. and Corbi, T. A.: The Dawning of the Autonomic Computing Era, *IBM Systems Journal*, Vol. 42, No. 1, pp. 5–18 (2003).

[3] Zheng, W., Bianchini, R. and Nguyen, T. D.: Automatic Configuration of Internet Services, *ACM EuroSys'07*, pp. 219–229 (2007).

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

[5] Sugiki, A., Kono, K. and Iwasaki, H.: Tuning Mechanisms for Two Major Parameters of Apache Web Servers, *Software – Practice and Experience* (2007). DOI 10.1002/spe.861.

[6] Chung, I.-H. and Hollingsworth, J. K.: Using Information from Prior Runs to Improve Automated Tuning Systems, *Proc. ACM/IEEE SC'04*, p. 30 (2004).

[7] Chung, I.-H. and Hollingsworth, J. K.: Automated Cluster-Based Web Service Performance Tuning, *IEEE HPDC'04*, pp. 36–44 (2004).

[8] Diao, Y., Eskesen, F., Froehlich, S., Hellerstein, J. L., Spainhower, L. and Surendra, M.: Genereic Online Optimization of Multiple Configuration Parameters with Application to a Database Server., *Proc. 14th IFIP/IEEE Int'l Workshop on Distributed Systems: Operations and Management*, pp. 3–15 (2003).

[9] 山田秀：実験計画法 –方法編–，日科技連 (2004).

[10] Osogami, T. and Kato, S.: Optimizaing System Configurations Quickly by Guessing at the Performance, *ACM SIGMETRICS'07*, pp. 145–156 (2007).

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

[12] Standard Performance Evaluation Corporation: The SPECweb2005 benchmark (2005). `http://www.spec.org/web2005/`.

[13] Chow, K.: J2EE Performance Optimization, Part 3 – Design of Experiments for Performance Tuning. `http://www.intel.com/cd/ids/developer/asmo-na/eng/182743.html`.

[14] Debnath, B. K., Skarie, J., Lilja, D. J. and Mokbel, M. F.: SARD: A Statistical Approach for Ranking Database Tuning Parameters, *Int'l Workshop on Self-Managing Database Systems (colocated with ICDE 2008)* (2008).

[15] 小谷和正，須田礼仁：汎用的なソフトウェア自動チューニング機構のための実験計画法の応用の検討，情報処理学会研究会報告 (2006-HPC-107(33)) (2006).

[16] Joshi, A., Yi, J. J., Jr., R. H. B., Eechhout, L., John, L. and Lilja, D.: Evaluating the Efficacy of Statistical Simulation for Design Space Exploration, *IEEE ISPASS'06*, pp. 70–79 (2006).

[17] Dunietz, I. S., Ehrlich, W. K., Szablak, B. D., Mallows, C. L. and Iannino, A.: Applying Design of Experiments to Software Testing: Experience Report, *Int'l Conf. on Software Engineering'97*, pp. 205–215 (1997).

[18] SAS: JMP 7. `http://www.jmp.com/`.

[19] The Apache Software Foundation: Performance Notes - Apache Tuning (2002). `http://www.apache.org/`.

[20] Diao, Y., Gandhi, N., Hellerstein, J., Parekh, S. and Tilbury, D. M.: Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics With Application to the Apache web server, *IEEE/IFIP NOMS'02*, pp. 219–234 (2002).