様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２５年４月２２日現在

機関番号：１２１０２
研究種目：若手研究（Ｂ）
研究期間：2010～2012
課題番号：２２７０００２３
研究課題名（和文）　ロバストなサーバソフトウェアの運用と設計開発：タグチメソッドを利用したアプローチ
研究課題名（英文）　A Study on Robust Design and Management of Server Software - Based on a Taguchi-Method Approach
研究代表者
杉木　章義（SUGIKI AKIYOSHI）
筑波大学・システム情報系・助教
研究者番号：５０５３６８２８

研究成果の概要（和文）：本研究では，インターネットサービスのサーバソフトウェアの分野に実験計画法の一手法であるタグチメソッドを適用し，環境の変化に対してロバストなサーバソフトウェアの運用管理手法，およびそれらをフィードバックした設計開発手法の構築を目指した．より具体的には，Apache と Hadoop を対象にロバスト性を高めるチューニング手法に関する研究を実施し，またその作業を支援するためのスクリプティング環境の整備も行った．

研究成果の概要（英文）：In this study, we have explored the possibility to achieve robustness in design and management of Internet server software. We adopted Taguchi-method, one of the designs of experiments techniques to achieve firm management of the server software against changes in the environments. We also sought a design strategy of the software to easily achieve robustness in the management steps. We conducted experiments with Apache web server and Hadoop clusters by using our tuning-aid scripting environments, which were built as a part of this research.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２０１０年度	1,000,000	300,000	1,300,000
２０１１年度	600,000	180,000	780,000
２０１２年度	600,000	180,000	780,000
総　計	2,200,000	660,000	2,860,000

研究分野：ソフトウエア
科研費の分科・細目：ソフトウエア
キーワード：並列分散システム，オペレーティングシステム，サーバ管理技術

１．研究開始当初の背景

ウェブサーバをはじめとするインターネットサーバでは，運用管理にかかるコストが大きな問題となっている．特に，サーバソフトウェアのパラメータ設定は性能や安定性に大きく影響し，困難性の高いタスクとなっている．その一方で，いまだに多くの場面で経験や勘に頼ったパラメータ設定が行われている．これらの手法は多大な時間と労力を要するのと同時に，系統的に確立された手法ではないため，潜在的な問題を内包している．この問題は，負荷が低い時には現れず，サー

バの負荷が急に増加した場合に性能低下や障害となって表面化する．

我々は，サーバソフトウェアのパラメータ設定手法の研究を進めており，この分野における最大の問題は性能測定に含まれる誤差や不安定性の管理であることが知見から得られている．特に，測定誤差に関しては長時間にわたり測定することで誤差を打ち消しているのが現状であり，SPECweb2005 では 30 分にわたる測定が義務づけられている．

実験計画法は，その名前が示す通り，予め実験を計画的に構成することで．少ない実験回数で最大の判別効果を目指した手法である．他の研究では，実験回数の削減を主な目的としているが，本研究では統計的な誤差の管理に着目している．タグチメソッドは実験計画法の中でも，製品の劣化や気温などの外的要因の影響を抑えながらパラメータ値を決定する場合に用いられる手法である．我々が管理できない外的要因をノイズとして捉え，S/N 比を向上させるように，パラメータ値を調整しておくことでロバスト性の向上を図る．本研究ではインターネットサーバのロバスト性向上にタグチメソッドを利用する．

実験計画法はいくつかの文脈で用いられ始めているが，タグチメソッドを利用し，サーバソフトウェアのロバスト性の向上を目指した例はまだないと言える．インターネットサーバへの実験計画法の適用は，Chow，Debnath，Xi ら，Chung らによって始められているが，実験回数の削減を主な目的としており，本研究のロバスト性の向上と目標が異なっている．また，タグチメソッドの適用も行われていない．

２．研究の目的

本研究では，実験計画法，特にその中でもタグチメソッドを適用したサーバソフトウェアのロバストな運用方法を目標とした．本研究におけるロバスト性とは，管理者が管理できない環境の変化にも関わらず，安定したサービス提供を目標とするという意味である．また，その成果をフィードバックしたサーバ設計手法も目標とした．

タグチメソッドは，他の工学分野やハードウェア分野などで品質改善のための手法として広く用いられており，サーバソフトウェアの運用管理にこの手法を適用することで，インターネットサービスにロバスト性をもたらすことができる．しかも，経験や勘に頼らない系統的な手法であり，アドホックになりがちな管理運用に新たな視点をもたらすことができる．

また我々は，元々サーバソフトウェアやミドルウェア開発を得意としており，運用管理での経験をふまえて，ソフトウェア設計開発に成果をフィードバックすることができる．

タグチメソッドは，日本国内において田口らによって開発され，世界中に広まった手法であり，国内でサーバソフトウェアへと適用分野を開拓することは非常に意義がある．

３．研究の方法

本研究は 3 年間にわたり，下記のステップに従って研究を実施した．

(1) ウェブサーバを使用した予備実験
本格的な実験に着手する前に，Apache ウェブサーバと比較的簡易でコントロールしやすいワークロードを使用して予備実験を行った．

(2) ロバスト性の規定
タグチメソッドなどのロバスト性を高める手法を適用するには，サーバの運用におけるロバスト性を規定することが必要である．本研究では，これを実施した．

(3) Hadoop を使用した本格実験
予備実験の成果を受けて，近年，データセンター環境で広く使用されている Hadoop を使用した本格実験に着手した．

(4) チューニング環境の整備
本研究のようなチューニング作業に関する研究を効率的に進めるためのスクリプティング環境を整備した．単にスクリプティング整備しただけではなく，実際に本環境を使用した Hadoop と Apache を使用したチューニングを行った．

４．研究成果

本章では，上記の研究の方法に対応し，それぞれで得られた研究成果を順に説明する．

(1) ウェブサーバを使用した予備実験

まず，アプローチの有効性の確認として，Apacheウェブサーバ1台で構成されたサーバを対象に，MaxClients，KeepAliveTimeoutなどの少数のパラメータに対してタグチメソッドを適用し，ロバスト性が向上するかどうか確認を行った．ベンチマークとしては，振る舞いが解釈しやすい少数のファイルにアクセスする単純なワークロードを人工的に生成し，使用した．その結果，限定的な環境ではあるが，ワークロードの変動に対して，応答時間が変化しにくいパラメータ値の組み合わせを見つけることができた．直感的な実験結果の解釈として，ロバスト性を実現するパラメータ値は，それぞれのワークロードにおける最適値の中間値に決まることが多く，パラメータ値を中庸に設定した方がロバスト性には有利であることが確認された．

また，Apacheウェブサーバを使用した実験を行った一方で，HadoopクラスタとHadoopに付属する標準サンプルプログラムをベンチマ

ークとして使用した予備実験も実施した．こちらも，パラメータ値がHadoopの性能に大きな影響を与えることが確認された．

(2) ロバスト性の規定

本研究では，まず一般的なロバスト性の定義を，外的な環境の変化にもかかわらず，Service Level Agreement の範囲内に収まることと規定した．より具体的には，それぞれのワークロードに応じて，応答時間が規定時間内に収まるように最小化する，スループットが規定以上となるように最大化するなどの目標設定を行った．

次に，サーバのロバスト性に影響を与える一方で，管理者がコントロールできない要因の分析を行った．これらの要因として様々なものが考えられるが，本研究では，ユーザの志向の変化にともなうワークロードの変化のみに焦点を当て，研究を実施した．これは，ワークロードの変化がロバスト性を損なう主要な要因となりやすく，また，ウェブサーバのコンテンツの変更や，ユーザの嗜好の変化によって時間とともに大きく変化しやすいからである．

(3) Hadoop を使用した本格実験

本研究課題の中核であるロバスト性の実現に関しては，Hadoop を使用し，本格的な実験を行った． Hadoop 付属プログラムを使用した予備実験により，一定の感触は得られているが，本格実験では標準ベンチマークとして広く知られている Gridmix2 を使用し，実験を実施した．まず，26 種類のパラメータに対してスクリーニング実験を実施し，6 つのパラメータで統計的に有意な効果が得られた（表 1）．次に，応答曲面法（RSM）を使用し，6 種類のワークロードごとで，これらの有意なパラメータの最適な値が異なることを確認した（図 1）．この実験では，6 種類のワークロードのうち，Combiner ワークロードのみが他のものと最適値が異なることが確認された．これは当初の想定から，少し意外な結果であった．また，Gridmix2 は標準的なベンチマークとして期待したにもかかわらず，Reduce タスクの方が全体の処理時間の多くを占める reduce-heavy なベンチマークであることが分かった．

以上の対策として，この他にももう一つの標準ベンチマークとして知られている HiBench を使用した実験も実施した．

(4) チューニング環境の整備

スクリプティング環境の整備では，従来，さまざまなスクリプトを組み合わせて行っていたチューニング作業を，統一的な環境で効率的に行えるようにした． 本環境では，チューニング対象のサーバや，ワークロードの生

表 1：Hadoop における各パラメータの性能効果（ANOVA 解析結果）

Parameter	Low Value	High Value	Stream Sort	Java Sort	Combiner	Web Data Scan	Monster Query	Web Data Sort
Env.Server	false	true	0.9816	0.9614	0.1548	0.2205	0.9142	0.9451
Env.UseParallelGC	false	true	0.5747	0.7356	0.0738*	0.4603	0.7375	0.5417
Env.PreferIPv4Stack	false	true	0.6087	0.6233	0.0527*	0.3630	0.8074	0.7476
Env.NAMENODE_MEM	500	1000	0.3026	0.4194	0.9779	0.3560	0.7337	0.5561
Env.SECONDARY_MEM	500	1000	0.6533	0.8332	0.6388	0.0774*	0.2365	0.5135
Env.DATANODE_MEM	500	1000	0.3865	0.4612	0.7027	0.3243	0.6375	0.8010
Env.BALANCER_MEM	500	1000	0.9360	0.3586	0.7688	0.6115	0.9728	0.3000
Env.JOBTRACKER_MEM	500	1000	0.1191	0.9156	0.6943	0.6359	0.5357	0.7814
fs.inmemory.size.mb	75	200	0.7480	0.4614	0.9292	0.7525	0.4549	0.8737
io.sort.factor	10	100	0.3199	0.6073	0.9095	0.1955	0.3001	0.8776
io.sort.mb	100	200	0.6587	0.5682	0.7275	0.4436	0.3023	0.9698
io.file.buffer.size	4096	131072	0.5339	0.9266	0.5740	0.7783	0.6194	0.8546
dfs.namenode.handler.count	10	40	0.7617	0.9862	0.0969*	0.7665	0.4884	0.4169
dfs.datanode.handler.count	10	40	0.6101	0.6492	0.5445	0.7892	0.1455	0.5790
dfs.datanode.max.xcievers	256	4096	0.7555	0.4745	0.3669	0.4476	0.5105	0.3735
mapred.compress.map.output	false	true	0.3774	0.2659	0.9338	0.7902	0.9771	0.6390
mapred.map.tasks.speculative.execution	false	true	0.2255	0.8657	0.3360	0.9113	0.4492	0.7195
mapred.reduce.tasks.speculative.execution	false	true	0.5998	0.9458	0.1499	0.1525	0.3203	0.6636
mapred.job.reuse.jvm.num.tasks	-1	1	0.1372	0.3766	0.2508	0.3032	0.5130	0.3180
mapred.tasktracker.map.tasks.maximum	1	4	0.7750	0.7057	0.0008*	0.3820	0.0157*	0.5187
mapred.tasktracker.reduce.tasks.maximum	1	4	<.0001*	<.0001*	<.0001*	<.0001*	<.0001*	<.0001*
mapred.job.handler.count	10	60	0.8030	0.9341	0.9132	0.2899	0.1079	0.3825
mapred.map.child.java.opts	-Xmx256m	-Xmx512m	0.8636	0.2177	0.4293	0.4757	0.5208	0.3774
mapred.reduce.child.java.opts	-Xmx256m	-Xmx512m	0.2526	0.9392	0.7936	0.4217	0.5443	0.3579
tasktracker.http.threads	40	60	0.5144	0.3050	0.0696	0.1708	0.5103	0.9084
mapred.reduce.parallel.copies	10	20	0.7092	0.4595	0.4329	0.5034	0.7418	0.6996

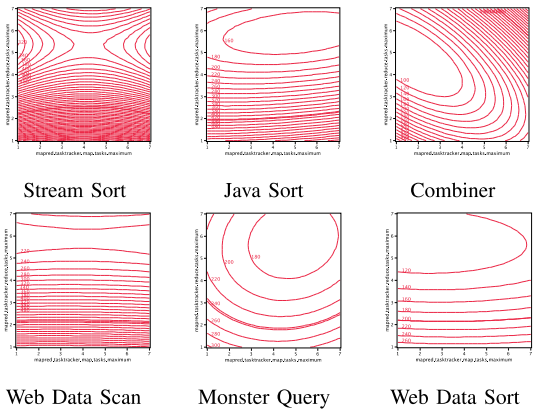

図 1：Map/Reduce タスク数を変化させた場合の処理時間の応答局面

```
1: /* Apache ウェブサーバの設定  */
2: val apache = pms(0).appm.add("Apache")
3: apache.maxClients = 512
4: apache.keepAliveTimeout = 15
5: apache.restart
6:
7: /* SPECweb ベンチマークの設定  */
8: val specweb = pms(1).appm.add("SPECwebPrime")
9: val clients = pms.drop(2).take(8).dmap{
10:    _.appm.add("SPECwebClient")
11: }
12: specweb.testType = specweb.Banking
13: specweb.sessions = 500
14: specweb.webserver = apache
15: specweb.clients = clients
16:
17: /* SPECweb クライアントの一斉起動 */
18: clients.dforeach(_.start)
19:
20: /* SPECweb ベンチマークの実行  */
21: val result = specweb.start
```

図 2：Apache チューニングの記述例

成に使用するベンチマーククライアントをスクリプティング環境上の言語オブジェクトとし，複数の計算機を同時に操作する実験であっても，透過的に一つの環境上でスクリプトとして記述可能とした．そのため従来と比べて，飛躍的に実験における苦労が軽減されている（図2）．

本研究課題を円滑に進めるために導入したチューニングのためのスクリプティング環境では，ApacheとHadoopの2つのサーバソフトウェアを対象に評価を行った．Apacheでは SPECweb2005のBanking，Ecommerce，Supportの3つのワークロードを使用し，Hadoopでは標準付属のWord Count，Sort，Grepの3つのプログラムを使用し，実験を行った．その結果，少量のスクリプト記述で，効率的にチューニングの過程を記述できることを確かめた．また，本環境では，手動でのチューニングだけではなく，シンプレックス法を使用した自動チューニングにも対応しており，実際に自動的にチューニングが行われることを実験により，確かめた（図3）．

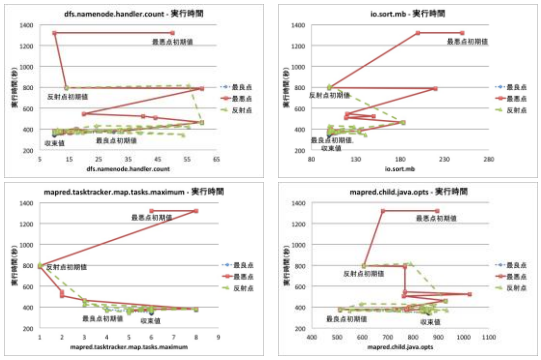

図 3：自動チューニングの過程

本研究は，ロバストなサーバソフトウェアの実現を目標に，サーバソフトウェアのチューニングに関する研究を実施した．本研究の特徴は，性能測定の際に発生する誤差を当然のものとしてとらえ，統計的な管理手法を使用しながらチューニングを実施することを目標とした点にある． また，単に性能の最高点を目指すのではなく，ロバスト性の向上に着目している点も特徴である．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 4 件）

[1] A. Sugiki and K. Kato, “Towards Robust Tuning of Hadoop by using Design of Experiments Approach”, IEEE PRDC 2012, Fast Abstract, 2 pages, 2012. 査読有

[2] 相川 拓也，杉木 章義，加藤和彦，“サーバ／チューニング記述のためのスクリプティング環境”，情報処理学会：論文誌（ACS），5(5)，pp. 138-151，2012 年．査読有

[3] A. Sugiki and K. Kato, “An Extensible Cloud Platform Inspired by Operating Systems”, IEEE UCC 2011, pp. 306-311, Dec. 2011. 査読有

[4] A. Sugiki, K. Kato, Y. Ishii, H. Taniguchi, N. Hirooka, “Kumoi: A High-Level Scripting Environment for Collective Virtual Machines”, IEEE ICPADS 2010, pp. 322-329, Dec. 2010. 査読有

〔学会発表〕（計 2 件）

[1] 相川 拓也，杉木 章義，加藤 和彦，“クラウド環境におけるサーバパラメータ調整のためのスクリプティング環境”，SWoPP 鹿児島 2011，2011 年 7 月 27 日（かごしま県民交流センター）

[2] 杉木 章義，加藤 和彦，“国産クラウド基盤ソフトウェア Kumoi の開発の現状と今後について”，情報処理学会 ComSys 2010，ポスター発表，2010 年 11 月 29 日（大阪大学中之島センター）

６．研究組織
(1)研究代表者

杉木　章義（SUGIKI AKIYOSHI）
筑波大学・システム情報系・助教
研究者番号：50536828