リバースプロキシサーバにおけるメモリと応答時間のパレート最適による keepalive_timeout の設定

坂本 一俊[1]　串田 高幸[1]

概要：リバースプロキシサーバの手動によるパラメータ調整は，管理者の経験に依存している．WEB サーバのキープアライブ 時間を設定する際に，リクエストに対する応答時間やスループットが低下することがある．本論文では，NGINX での keepalive_timeout の値を自動設定することを提案する．keepalive_timout の値が変化するとサーバのメモリの使用量やサーバからの応答時間が変化する．そこで，負荷試験を行いメモリの使用量とサーバからの応答時間を計測して keepalive_timeout の値をパレート最適によって算出する．負荷試験では，Locust を用いて NGINX に負荷をかけた．その際に毎秒 1 ユーザ増やし最大 10 ユーザまで増やす設定にした．その結果リクエスト数は，毎秒 3000 以上 3600 未満のリクエストを行った．評価として，NGINX の設定ファイルに入っていたデフォルトの値と算出した値について比較する．keepalive_timeout を自動設定する実験を行った結果，keepalive_timeout の値は，70 となり，サーバからの応答時間は 0.04ms が短くなった．メモリの使用量は 30KiB 減らすこととなった．

1. はじめに

背景

サーバのパラメータをサーバのメモリの使用量によって動的に設定することが必要である．現状のパラメータ設定は管理者の経験をもとに試行されている [1]．手動によるパラメータ設定は多大な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 [2].

WEB サービス提供者は，サーバのメモリに合わせてパラメータを設定していないため，WEB サービスの利用状況とサーバの CPU やメモリの性能によって望ましい性能が得られない場合がある．また設定の誤りも起きやすく，サーバの主要な障害の原因になっている [3]．クライアントとの通信で HTTP の規約により，1 つの TCP 接続で複数の要求やファイルの送受信を行うことができる．これはキープアライブ接続と呼ばれ，ネットワークの使用率やサーバからの応答時間が短くなることが知られている [4]．例えば，NGINX では，keepalive_timeout というパラメータがある．これはクライアントとサーバ間の接続のタイムアウトを決めるパラメータである．キープアライブ接続中にデータの送受信をしない期間が keepalive_timeout の値を超えると接続が切断される．

課題

課題は，NGINX のデフォルトの keepalive_timeout の値は，管理者の経験に依存して設定していることである．keepalive_timeout の値がより大きいとメモリの使用量が増え，サーバ内のメモリの空き容量を圧迫する．しかしサーバからの応答時間が短くなりクライアントがキープアライブ接続中に WEB サービスにリクエストを送ってからレスポンスを受け取るまでの時間が短くなる．keepalive_timeout の値がより少ないとメモリの使用量が減るのでサーバ内のメモリの空き容量を圧迫しない．しかし応答時間が長くなりクライアントが WEB サービスにリクエストを送ってからレスポンスを受け取るまでの時間が長くなる．またサーバにはメモリの容量に，制限がある．そのため最適な keepalive_timeout の値を動的に変更する必要がある．

各章の概要

第 2 章では関連研究について述べる．第 3 章では提案するシステムについて述べる．第 4 章では提案したシステムの実装と実験方法について述べる．第 5 章では実験の評価と分析を述べる．第 6 章では提案したシステムの議論を述べる．第 7 章では本研究のまとめを行う．

2. 関連研究

初めに，ロングタームエボリューション (LTE) ネットワークでのキープアライブメッセージによって引き起こ

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1

されるシグナリングオーバーヘッドの研究がある [5]．この研究では，スマートフォンの常時接続タイプのアプリケーションから生成されたキープアライブメッセージは，ロングタームエボリューション（LTE）無線リソース制御（RRC）状態間の頻繁な遷移につながり，シグナリングトラフィックの大幅な増加を後押しする可能性があることを課題としている．アップリンクとダウンリンクの両方のデータおよび制御チャネルでのシステムリソース消費の観点から，シグナリング負荷を評価するための一連のメトリックを提案している．この研究では，LTE ネットワークにおけるキープアライブであり，リバースプロキシサーバにおけるキープアライブの最適値の算出ではない．

応答時間ベースの最適な WEB サービスの選択についての研究がある [6]．この研究では，インターネットサービスにおいてパフォーマンスの低いサーバ，待ち時間の長さなどの全体的なサービス品質の低下は，売上損失，ユーザの不満につながる可能性があることを課題としている．隠れマルコフモデルに基づく QoS 計測の新しい手法を提案し，ユーザ要求の実行に最適なパスをさらに提案している．これにより，ユーザが最も信頼性の高い WEB サービスを自動的に選択するのに予測精度を 12%向上した．ここでは WEB サービスのクオリティがサーバにあるとして，隠れマルコフモデルに基づいて QoS 計測する方法を提案しているが，サーバのキープアライブの値は含まれていない．

3. 提案方式

提案方式

課題に対する提案方式として，従来は手動により設定されていた NGINX の keepalive_timeout を負荷試験によってサーバからの応答時間とメモリの使用量を計測しパレート解を算出する．パレート解から 1 つの keepalive_timeout の値を算出して自動設定する手法を提案する．図 1 に提案の概要図を示す．

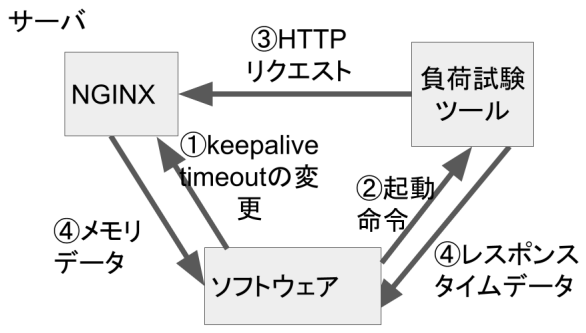

図 1 提案の概要図

図 1 では，サーバの管理者がソフトウェアを起動した際の負荷試験ツールと NGINX で①から④を行い，そこから最適な keepalive_timeout の値を算出する流れを示している．

初めに①の keepalive_timeout の変更では，NGINX の設定ファイル内の keepalive_timeout の値を変更する．これは負荷試験で keepalive_timeout の値ごとのサーバからの応答時間とメモリの使用量を取得する必要があるためである．

2 つ目に②の起動命令では，負荷試験を行う．負荷試験ツールとしてについて Locust を用いて行う．Locust を用いる理由は，負荷試験結果としてリクエストに対する応答時間の平均が得られるためである．Locust は，③の HTTP リクエストのようにリバースプロキシサーバの NGINX に HTTP の GET リクエストを行う．その際のメモリの使用量も計測する．

3 つ目にソフトウェアでパレート解を算出する．その際④のメモリデータとレスポンスタイムデータをソフトウェアが読み込む．それぞれのデータは，keepalive_timeout の値ごとのメモリの使用量の平均と，サーバからの応答時間の平均からパレート解の算出を行っている．パレート解とは，トレードオフの関係にある 2 つの目的を最小化 (最大化) するような実現可能な解の集合から優れた解の集合のことである．選ばれなかった解のことを劣解という．このパレート解を求めることで，keepalive_timeout の優れた値を求めることができる．

パレート解から 1 つの値を算出する方法

パレート解から 1 つの keepalive_timeout の値を算出する方法として，keepalive_timeout の値は，5 から 100 まで 5 ずつの 20 個ある．keepalive_timeout の値ごとにサーバからの応答時間の平均とメモリの使用量の平均の合計を求める．そのときの最小値を keepalive_timeout の値とする．WEB サービスにおいてサーバからの応答時間を短くする必要があるためである．また，NGINX のメモリの使用量を減らす必要があるためである．

ユースケース・シナリオ

本研究のユースケースは，WEB サービスを提供している WEB サーバと，リバースプロキシサーバの NGINX がある．クライアントが WEB サービスにアクセスするとリバースプロキシサーバを経由して WEB サーバにアクセスしている．このリバースプロキシサーバを管理のリバースプロキシサーバにおいて keepalive_timeout の値が小さくサーバからの応答時間が長くなる．また keepalive_timeout の値が大きく NGINX のメモリ使用量の増えることでキープアライブ接続に影響がある場合を例とする．ユースケースについて図 2 に示す．

WEB サーバはここでは HTML・CSS ファイルを配信するサーバである．ここで作成したソフトウェアを使うことで，keepalive_timeout の値が変更され，メモリの使用量やサーバからのリクエストからレスポンスまでの応答時間が

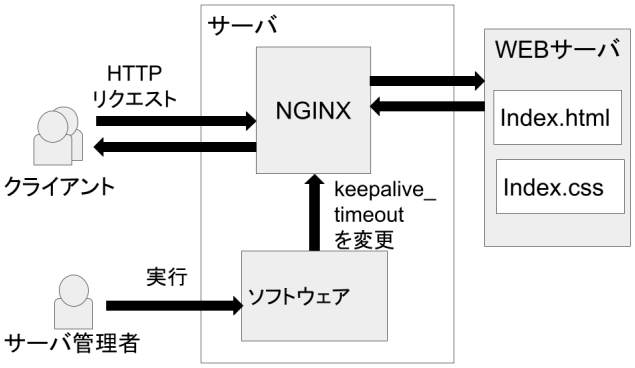

図 2 ユースケースシナリオ

短くすることができる．

4. 実装と実験手法

実装

本研究のソフトウェアのファイルの構成について図 3 に示す．仮想マシン 1，2 は，異なる物理マシンのハイパーバイザー上にある Ubuntu20.04 である．仮想マシン

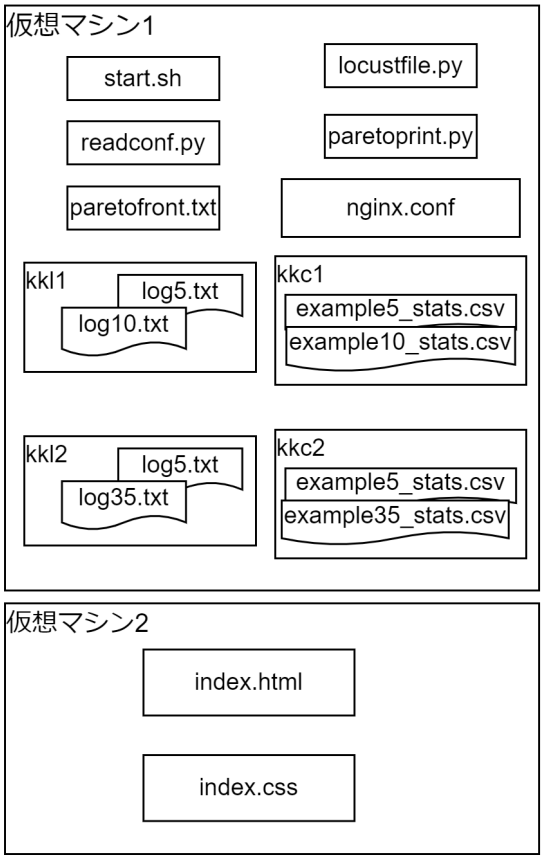

図 3 ファイルの構成

1 には，シェルスクリプトで書かれた start.sh が存在する．このファイルで Python ファイルである readconf.py と paretoprint.py を実行している．readconf.py は NGINX の nginx.conf の書き換えを行っている．paretoprint.py は keepalive_timeout の値ごとのサーバからの応答時間とメモリの使用量からパレート解を paretofront.txt に書き込む．またパレート解から 1 つの値を算出する．

また start.sh で top コマンド locust コマンドを実行する．nginx.conf は NGINX の設定ファイルである．kkl1, kkl2 には負荷試験の際に得られたメモリの使用量を保存している．keepalive_timeout の値が log.txt の間に書き込まれている．kkc1，kkc2 には負荷試験の際に得られたサーバからの応答時間を保存している．kkl1 と同様 exampl_stats.csv の間に keepalive_timeout の値が書き込まれている．locustfile.py には Locust を実行する際にどのような負荷をかけるのか書かれている．

仮想マシン 2 には，WEB サービスとして WEB サイトの Index.html と Index.css が保存されている． それぞれ HTML と CSS で書かれている．作成したソフトウェアのフローチャートについて図 4 に示す．

作成したソフトウェアの start.sh を起動する．start.sh の引数にはクライアントから WEB サービスにアクセスする際の URL を入力する．ここでは，以下の 7 つのことを行う shell scripts になっている．

1 つ目に，readconf.py を起動する．ここで NGINX の設定ファイルの nginx.conf を読み込んで keepalive_timeout の文字列を検索し，値を変更する．

2 つ目に値を変更した NGINX の設定ファイルを nginx -t コマンドで正しい構文であるかどうか設定ファイルのチェックを行う．そして設定ファイルの再読み込みを行う nginx -s reload コマンドを実行する．

3 つ目に負荷試験を行う．locust -f locustfile1.py コマンドで Locust の起動する．Locust は，ユーザが NGINX に GET リクエストを行う. 毎秒 1 ユーザ増やし，最大 10 ユーザとなっている．この値の理由は，これ以上の値を選択すると CPU 使用率が 90%以上になり，正常なリクエストが行えない警告が表示されてしまうためである．Locust は一回の負荷が終わるごとに，example_stat.csv ファイルを生成する．その生成されたファイルの内容は，サーバからの応答時間の平均値である．これは，kkc1 のディレクトリに保存する．また並行して，1 秒ごとに top コマンドを用いてメモリの使用量を表示し，grep で NGINX の含む行のみを抜き出し log.txt に書き込みをしている．以上の 3 つのことを keepalive_timeout の値が 5 から 100 まで 5 ずつ増やして 20 回ループするようになっている．このテキストファイルは，kkl1 に保存する．

4 つ目に paretoprint.py でパレート解を算出する．算出したパレート解を paretofront.txt に書き込む．パレート解を算出する手法として図 5 に示す．

f1 は keepalive_timeout の値ごとのサーバからの応答時

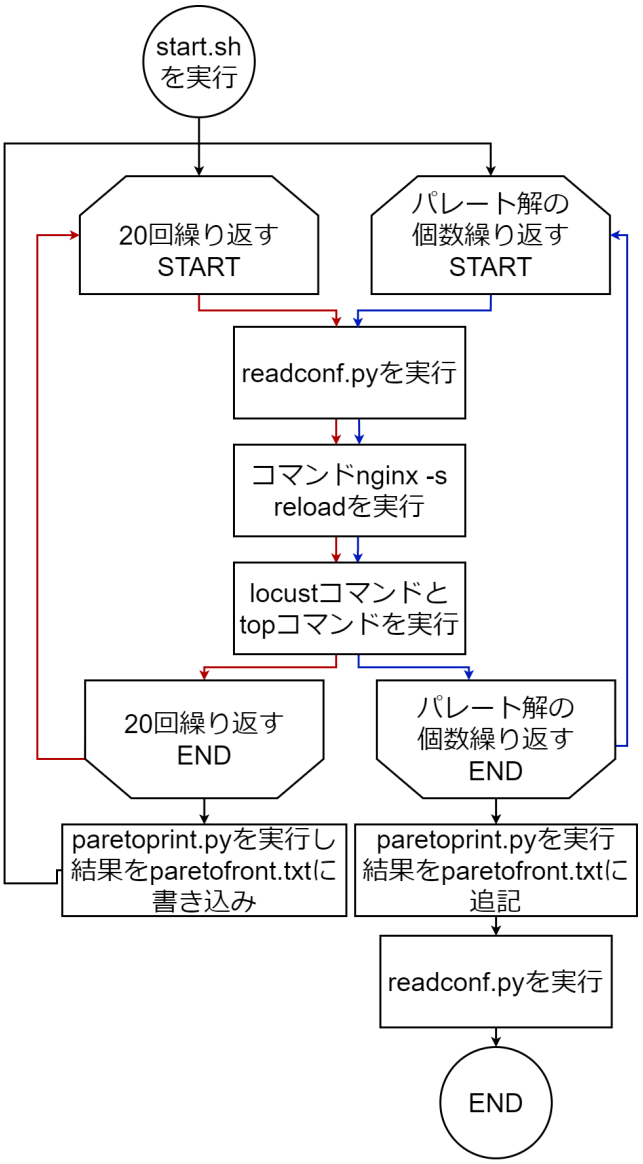

図 4　ソフトウェアのフローチャート

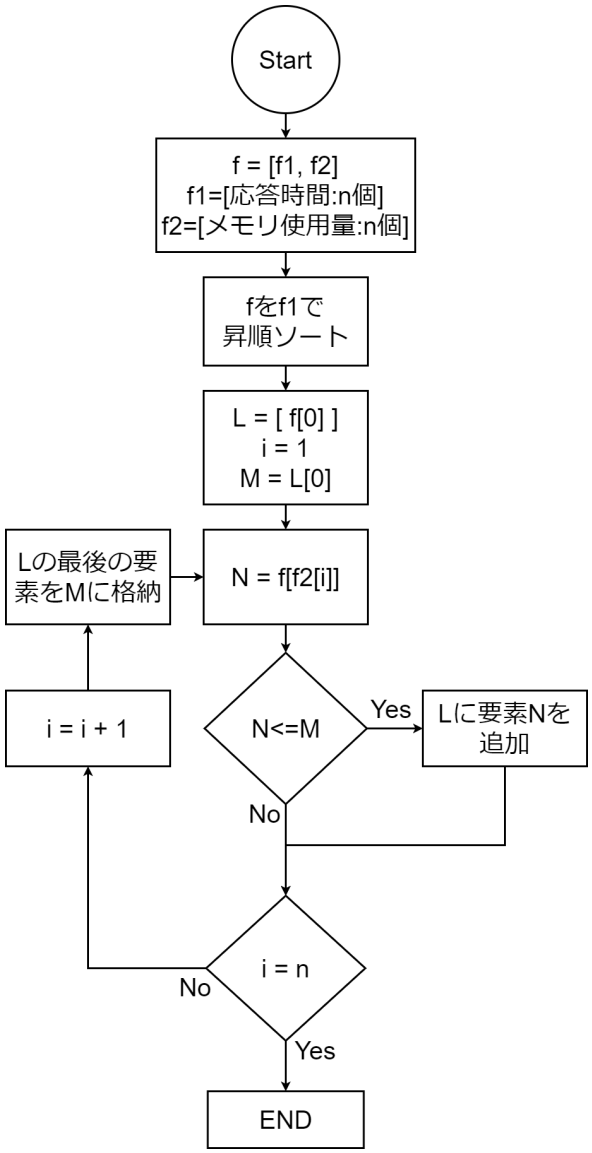

図 5　パレート解の算出手法

間の平均である．f2 は keepalive_timeout の値ごとのメモリの使用量の平均である．f は，f1 と f2 の集合である．L はパレート解のリストである．i は 0 から n までの変数である．M は L の最後の要素が格納される変数である．N は f2 のい番目の値が格納される変数である．

パレート解の算出手法について説明する．実行可能解の集合からメモリの使用量を最小化し応答時間を最小化することでパレート解を求める．実行可能解とは，数理計画問題において実行可能領域上の点の個お t である．データ f は，keepalive_timeout の値ごとのサーバからの応答時間のデータ f1 と，メモリの使用量のデータ f2 である．データ f をサーバからの応答時間 f1 の値で昇順に並べる．並べ変えられたデータ f の 0 番目のメモリの使用量 f2[0] をパレート解 L のリストに追加する．パレート解 L の最後の要素を M とする．これにより応答時間が最小である f[0] のメモリの使用量を M に格納することができる．データ f の i 番目 f2[i] を N とする．その後に N と M で比較をする．N の値が大きければ，i を 1 足して再度 N を定義し比較する．N の値が小さければ，L に N の値を追加する．追加することで応答時間が M よりも長いがメモリの使用量が少ない要素を抽出することができる．i が総要素数 n に達するとパレート解の算出が終了する．

5 つ目に paretofront.txt に書き込まれた値ごとに同じ負荷をかけサーバからの応答時間とメモリの使用量を kkl2, kkc2 のディレクトリに保存する．

6 つ目に paretoprint.py で keepalive_timeout の値ごとのサーバからの応答時間の平均とメモリの使用量の平均の合計が最も小さいパラメータを paretofront.txt の最後の行に書き込む．

7 つ目に readconf.py を起動して 6 つ目で算出した keepalive_timeout の値である paretofront.txt の最後の行を読み込み，設定ファイルの nginx.conf の keepalive_timeout に書き込む．

実験環境

開発ソフトウェアを使って実験したときのハードウェアとソフトウェアの構成について図 6 に示す．

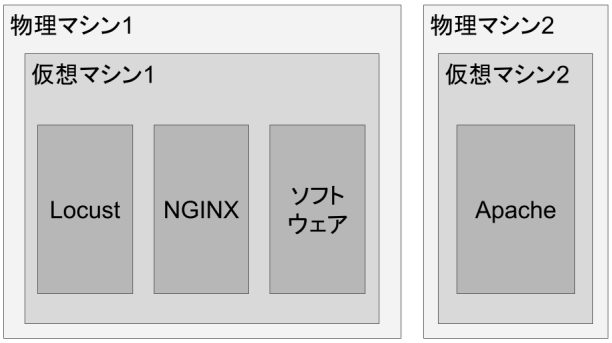

図 6 実験の環境

それぞれの物理マシンそれぞれにハイパーバイザー (VMwere ESXi) 上の仮想マシンを利用する．仮想マシンに Ubuntu20.04 をインストールしている．それぞれのマシンスペックについて，仮想マシン 1 は，CPU が 8 プロセッサー，メモリ 8GB，仮想マシン 2 は，CPU が 1 プロセッサー，メモリ 6GB である．その上で仮想マシン 1 には，Locust・NGINX と本研究で作成したソフトウェアをインストールし，仮想マシン 2 には，Apache をインストールした．Locust のバージョンは，2.5.1 である．NGINX のバージョンは，1.18.0 である．Apache のバージョンは，2.4.41 である．

5. 評価と分析

本研究のソフトウェアを評価する手法として，keepalive_timeout の値のデフォルトである 60s と比較したときのサーバの応答時間とメモリの使用量について評価する．すべての負荷試験において Locust は，ユーザが NGINX に GET リクエストを行う. 毎秒 1 ユーザ増やし，最大 10 ユーザで設定した．その結果 Locust は毎秒 3000 以上 3600 未満のリクエストをした．

keepalive_timeout ごとの結果

keepalive_timeout の値ごとのサーバからの応答時間の平均を図 7 に示す．

応答時間の平均について相関係数が-0.029 となり相関がない結果となった．これは，負荷試験のユーザ数が少ないためである．次にメモリの使用量の平均を図 8 に示す．

メモリの使用量の平均について相関係数は，0.073 となり相関がない結果となった．これは，負荷試験のユーザ数が少ないことと実験で使用した WEB サイトは 133KB と 243KB とサイズが小さいためである．

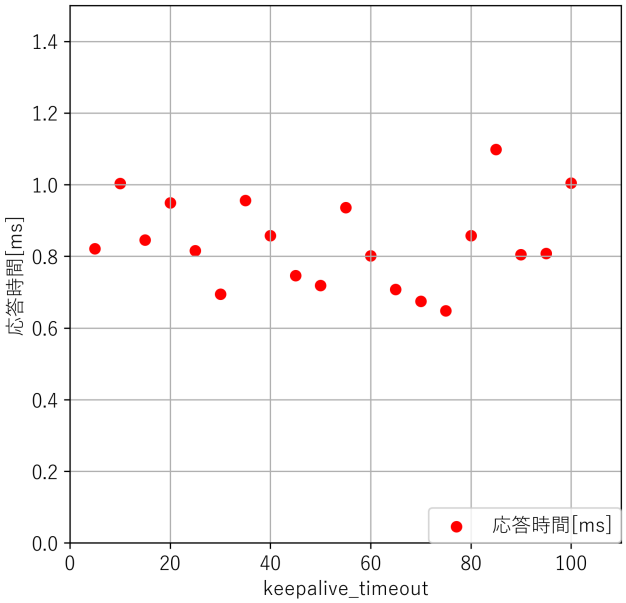

図 7 平均応答時間

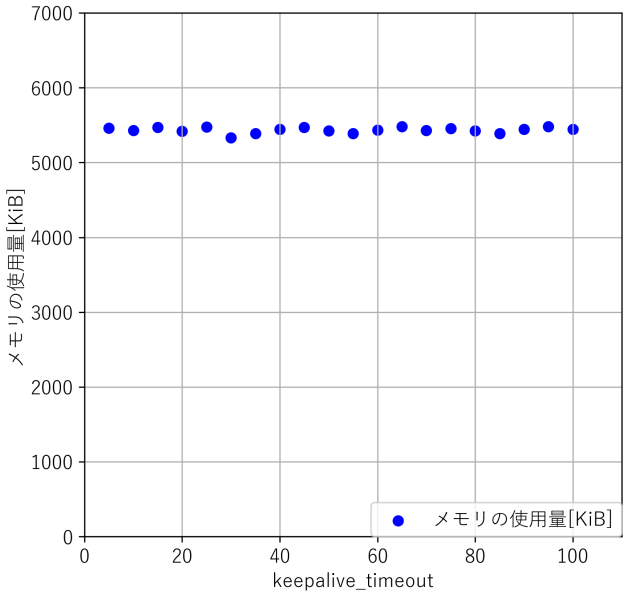

図 8 平均メモリ使用量

パレート解について

ソフトウェアを実行したところ算出されたパレート解は，30s, 70s, 75s であった．実装の結果の散布図を図 9 に示す．

赤い点がパレート解でそれらを結んだ赤い線がパレートフロントである．青い点は，パレート解に含まれなかった，劣解である．この図から劣解と比べてパレート解が劣解と比べてメモリの使用量が低く応答時間が少ないことを示している．

デフォルトとの違いについて

実装の結果から最適な値として 70 が算出された．パレー

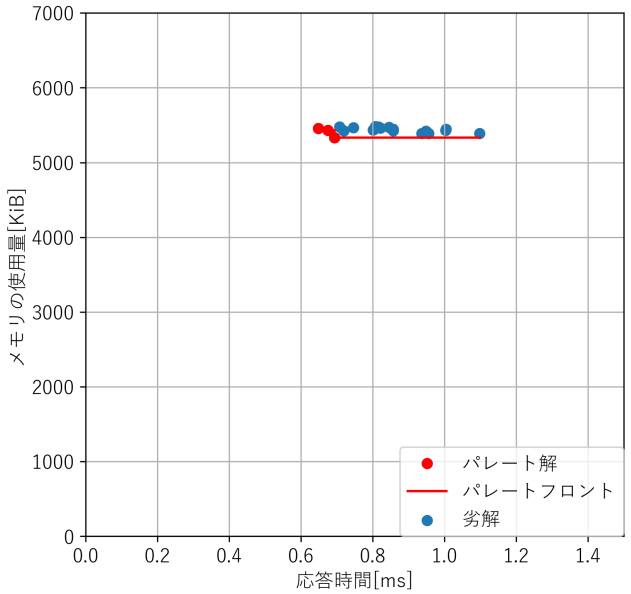

図 9 応答時間とメモリの使用量の散布図

ト解とデフォルトの値 60s のサーバからの応答時間とメモリの使用量の変化を表 1 に表す．表によると最適と算出

表 1 実装の結果

keepalive_timeout	60	70
応答時間 [ms]	0.80	0.84
メモリの使用量 [KiB]	5435	5408

した値 70s とデフォルトの 60s と比べてメモリの使用量が 27KiB 減ることからメモリの使用量は 0.5%減らせることがわかった．サーバからの応答時間は，0.04ms 減ることがわかった．

6. 議論

提案手法では, 負荷試験ツールを用いて keepalive_timeout の値ごとに負荷試験を行い，サーバからの応答時間と top コマンドを使用しメモリの使用量を計測した．計測した結果から，パレート解の算出を行った．パレート解の keepalive_timeout の値ごとに負荷試験を行いサーバからの応答時間とメモリの使用量を計測した．計測した結果から最小値を最適な値とした．

この手法では，2 回負荷をかけなければ一つの最適な keepalive_timeout の値を出すことができなかった．よって keepalive_timeout の値ごとに負荷をかけた際の，サーバからの応答時間との近似曲線とメモリの使用量との近似曲線を求める．それらを目的関数として，1 変数 2 目的の多目的最適化問題を NSGA-II を用いて解く必要がある [7]．また実験環境としてリバースプロキシサーバとして NGINX と Locust を同じマシン内に作成されている．そのため測定方法では本来のアクセスとは異なるため，サーバからの応答時間が正しいか判別ができない．正しく判別するため，違うマシンで物理的に距離が離れている必要がある．

7. おわりに

本研究では，NGINX の keepalive_timeout の最適な値を算出する必要がある．そのため提案方式としてパレート最適を用いて 1 つの keepalive_timeout の値にする手法を用いた．結果は，パラメータの値によってサーバからの応答時間が 0.04ms 短くなった．またメモリの使用量を 27KiB 減らした．本研究のパラメータの設定手法により WEB サーバのメモリの使用量を減らすことができる．

参考文献

[1] Xi, B., Liu, Z., Raghavachari, M., Xia, C. H. and Zhang, L.: A Smart Hill-Climbing Algorithm for Application Server Configuration, *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, New York, NY, USA, Association for Computing Machinery, p. 287–296 (online), DOI: 10.1145/988672.988711 (2004).

[2] Xi, B., Liu, Z., Raghavachari, M., Xia, C. and Zhang, L.: A Smart Hill-Climbing Algorithm for Application Server Configuration, *Thirteenth International World Wide Web Conference Proceedings, WWW2004*, (online), DOI: 10.1145/988672.988711 (2004).

[3] 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, Technical Report UCB/CSD-02-1175, EECS Department, University of California, Berkeley (2002).

[4] Nielsen, H. F., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H. W. and Lilley, C.: Network Performance Effects of HTTP/1.1, CSS1, and PNG, *SIGCOMM Comput. Commun. Rev.*, Vol. 27, No. 4, p. 155–166 (online), DOI: 10.1145/263109.263157 (1997).

[5] Zhang, Z., Zhao, Z., Guan, H., Miao, D. and Tan, Z.: Study of Signaling Overhead Caused by Keep-Alive Messages in LTE Network, *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, pp. 1–5 (online), DOI: 10.1109/VTCFall.2013.6692424 (2013).

[6] Ahmed, W., Wu, Y. and Zheng, W.: Response Time Based Optimal Web Service Selection, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 26, No. 2, pp. 551–561 (online), DOI: 10.1109/TPDS.2013.310 (2015).

[7] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197 (online), DOI: 10.1109/4235.996017 (2002).