# Nginx の worker_connections 設定のための負荷試験の自動化

栗原 尚希[1]　大野 有樹[2]　串田 高幸[1]

**概要**：Nginx の設定ファイルである nginx.conf ファイルの作成は，Nginx をリバースプロキシの OSS として使用する際に必要不可欠な工程である．しかしこの工程にかかる時間は，熟練者と初学者で差がある．本稿では，設定項目の worker_connections をデフォルト値から変更する必要性の有無の判別を自動化する．それによる nginx.conf 作成時間の短縮を目標にする．負荷試験の際の評価基準はエラーの発生の有無とした．負荷試験の結果，worker_connections がデフォルト値の場合，秒間リクエスト量が 40.7 を超えるとエラーが発生した．

## 1. はじめに

### 背景

Nginx を使用したリバースプロキシの構築工程には，設定ファイルである nginx.conf の作成がある．nginx.conf はディレクティブと言われる設定項目とその設定値で構成される．ディレクティブは基本ディレクティブとブロックディレクティブの 2 つに分類できる．基本ディレクティブば「 設定項目 設定値; 」形式を意味し，ブロックディレクティブは「 設定項目 A 設定項目 B 設定値 b; 」形式のように波括弧で囲った一連の設定項目とその設定値を意味する．ブロックディレクティブが波括弧内に他のディレクティブが定義できる場合，コンテキストと呼ぶ．これには events，http，server，location ディレクティブが該当する．コンテキスト外の構成ファイルに置かれたディレクティブは，メインコンテキストとみなされ，events と http ディレクティブはメインコンテキストに属し，server は http に，location は server に属す[*1]．ソースコード 1 は nginx.conf ファイルのサンプルである．

この nginx.conf の作成には労力がかかる．なぜならユーザの nginx への理解が必要なためである．nginx.conf を構成するコンテキストは 100 以上存在し，nginx に必要とされる動作をさせるにはそれに該当するコンテキストをユーザが探し出し，適切な設定値を記述する必要がある．コンテキストの説明は英語で書かれており，プロキシ関連の専門用語も複数使われている．さらにコンテキストに設定する値の決定にも，プロキシやサーバの知識が必要となっている．

本稿で取り上げている worker_connections は events に属するディレクティブである．worker_connections は worker プロセスによって開かれる最大同時接続数を設定するためのコンテキストである．worker_connections の項目を nginx.conf に書き込んでいない状態では，最大同時接続数はデフォルト値の 512 となる[*2]．

**ソースコード 1**　nginx.conf のサンプル

```
1  worker_processes 1;
2  events {
3      worker_connections 1024;
4  }
5  http {
6      server {
7          listen 80;
8           server_name localhost;
9
10         location / {
11             root html;
12             index index.html index.html;
13          }
14
15         error_page 500 502 503 504 /50x.html;
16          location = /50x.html {
17             root html;
18          }
19     }
20 }
```

[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
[2] 東京工科大学院バイオ・情報メディア研究科
〒 192–0982 東京都八王子市片倉町 1404-1

### 課題

nginx.conf ファイルの worker_connections の値を決定するための過程には，負荷試験ソフトの構築や worker_connections の変更，その上での負荷試験実行の作業が必要である．この作業は Nginx サーバ構築のための，nginx.conf ファイル作成の負荷の一つとなってる．

Nginx のリバースプロキシサーバを構築する際，worker_connections のデフォルト値である 512 は，リバースプロキシサーバとしての処理を行うためのリソースとして不十分である[*3]．不十分となりうる状況の一つとしてクライアントからのリクエスト数が増加した場合があげられる．図 1 は worker_connections 不足によるエラーを示している．Nginx はクライアントからのリクエスト増加によりワーカープロセスの同時接続数が上限に達するとそれ以上のリクエストに対する処理が行えない．worker_connections の値がデフォルト値である 512 でワーカープロセス接続数が 512 を超えるとき，worker_connections が不足する．このとき，Nginx を起動しているリバースプロキシサーバで”worker_connections are not enough”のエラーとなる．このエラーが発生すると，クライアントから来たリクエストの正常な処理が行えず，クライアントはサーバにリクエストしたデータ，例えば WordPress サーバにある WEB サイトのデータを得ることが出来ない．

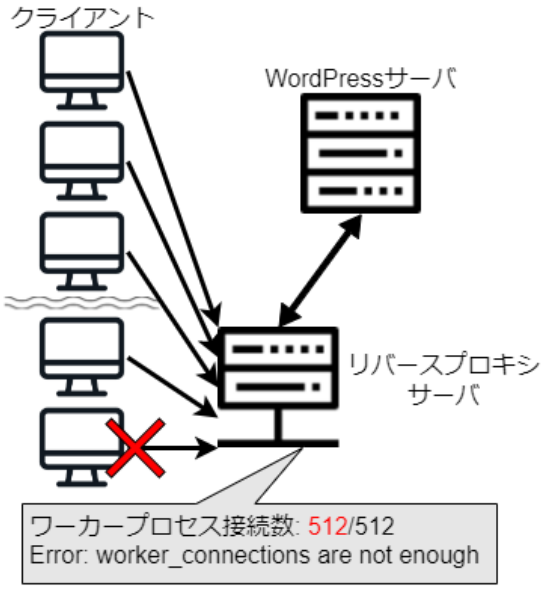


**図 1** worker_connections 不足によるエラー

[*1] IT 生涯学習 Tech Home nginx 設定ファイルの概要 https://gakumon.tech/nginx/nginx_conf_overview.html(2022 年 12 月 16 日)
[*2] Core functionality - Nginx.org http://nginx.org/en/docs/ngx_core_module.html#worker_connections(2023 年 1 月 10 日)
[*3] Avoiding the Top 10 NGINX Configuration Mistakes https://www.nginx.com/blog/avoiding-top-10-nginx-configuration-mistakes/(2023 年 1 月 11 日)

### 各章の概要

第 2 章では関連研究について述べる．第 3 章では本稿の提案方式を説明する. 第 4 章では実装及び実験方法を述べる. 第 5 章では提案方式の評価と分析を行う. 第 6 章では本稿の提案方式についての議論を行う. 第 7 章では本稿のまとめと成果を述べる.

## 2. 関連研究

Nginx ベースの WEB サーバのチューニングを行っている研究がある [1]．これは WEB サーバのスレッドやプロセスの値，キャッシュサイズの設定を，最適化・分析をしている．これにより，WEB サーバの総合的なパフォーマンスが向上した．本稿ではこの関連研究で行われた分析作業の部分的な自動化を行った．

## 3. 提案

### 提案方式

本稿は課題に対する提案方式として，Nginx の pod 立ち上げ作業からリバースプロキシサーバの負荷試験までの一連作業の自動化を行う．ここで作成する提案ソフトウェアは，ユーザが負荷試験を行う際に試験したい worker_connections の設定値を，実行時に入力できるものとする．提案方式の概要を図 2 に示す．

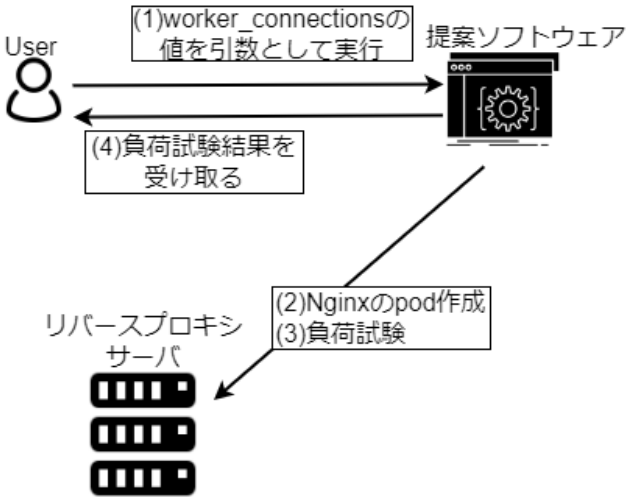


**図 2** 提案の概要図

図 2 の流れを具体的に説明する．
( 1 ) worker_connections の設定値を引数として実行
( 2 ) K3s をインストールし引数をもとに Nginx の pod 作成
( 3 ) Nginx で構築したサーバに Locust で負荷試験
( 4 ) Locust で試験された結果をユーザに返答
まず始めに (1) でユーザは試験したい worker_connections の設定値を引数として，提案のソフトウェアを実行する．次に (2) でユーザがリバースプロキシサーバとして使用したいサーバ上に，提案ソフトウェアが K3s をインストー

ルする．K3s がインストールされることで kubectl を用いて K3s サーバー上に (1) の実行時の引数を基にした Nginx の pod を提案ソフトウェアが作成する．(3) で構築されたリバースプロキシサーバに，Locust での負荷試験が行われる．(4) で Locust で試験された結果がユーザに返却される．

図 2 の動作をすることで，Nginx の pod 作成と worker_connections の設定値を決定するための負荷試験，さらに別の設定値に変更する作業が行われる．これにより worker_connections を変更する必要性の判断材料である負荷試験のデータが得られるため，課題であるこれらの工程にかかる時間を短縮することが出来る．

### ユースケース・シナリオ

ユースケースは個人が所有するブログ用の WordPress を想定している．データを保管する WEB サーバを複数所有し一纏めの WEB サーバとして使用する場合や，クライアントからのリクエスト処理を分散させたい場合は，リバースプロキシサーバを設置することが方法の一つとなる．本稿ではサーバ初学者がリバースプロキシサーバを組み込んだ WordPress を構築することを想定している．ユースケースについての図 3 に示す．

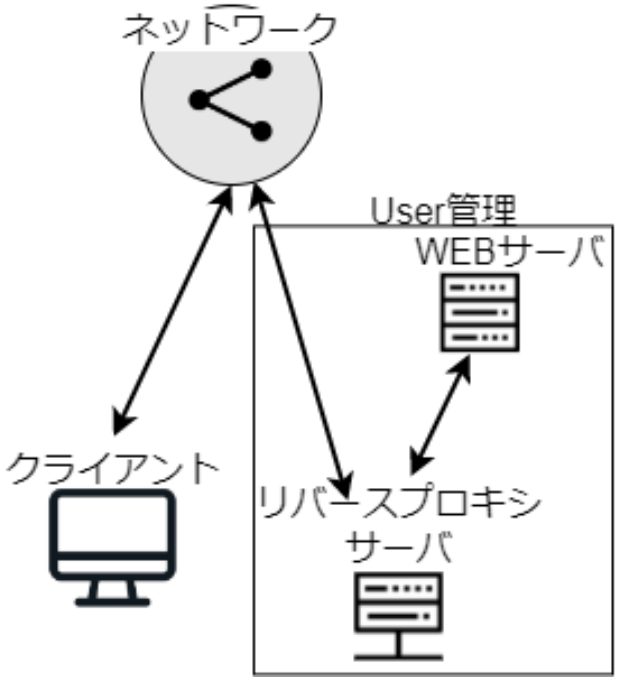


**図 3** ユースケースの図

Nginx ベースのリバースプロキシサーバを構築には，設定ファイルである nginx.conf ファイルの作成作業が含まれている．この作業は，使用する Nginx の設定項目の選別や，設定項目に設定する値そのものの決定が主であり，サーバ・ネットワークの知識や 100 以上存在する設定項目への理解が必要不可欠である．この作業をサーバ構築経験の乏しいユーザが行う場合，熟練者よりも時間がかかることが予想される．そのため，初学者と熟練者の作業時間の差を縮める提案が求められる．

## 4. 実装

提案手法を基にしたソフトウェアを新たに作成する．

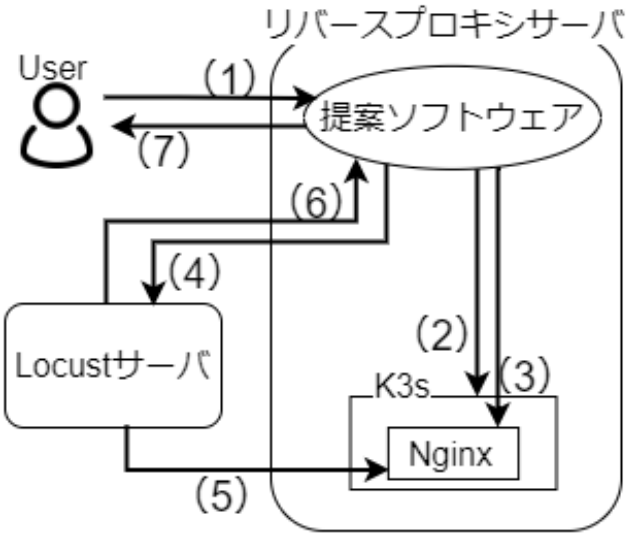


**図 4** 実装の図

図 4 は実装の図を示している．サーバは Locust サーバとリバースプロキシサーバの 2 つがある．リバースプロキシサーバは提案のソフトウェアを配置し，K3s に Nginx を起動する．以下に実装の流れを説明する．ユーザーは (1) でリバースプロキシサーバーにある提案のソフトウェアを Nginx の設定値の 1 つである worker_connections の値を引数として実行する．(2) で提案のソフトウェアは K3s をインストールする．(3) で提案ソフトウェアは kubectl コマンドを用いて Nginx の Pod 作成をする．このとき，Nginx の設定値の 1 つである worker_connections の値を nginx.conf に代入して Nginx を起動している．(4) で提案のソフトウェアは K3s に Nginx の Pod が起動したことを確認して Locust サーバに負荷試験の要求をする．(5) で Locust サーバは Nginx に負荷試験を実施する．(6) で Locust サーバは提案ソフトウェアへ負荷試験された結果を送信する．最後に (7) で提案のソフトウェアは Locust サーバからの負荷試験の結果をユーザーに返答する．

## 5. 実験と分析

### 実験環境

実験環境として以下の環境の Ubuntu22.4 の仮想マシンを 3 台構築して使用した．各仮想マシンの性能は全て同じである．

**OS:** Ubuntu-22.04

**vCPU:** 2 コア
**RAM:** 4GB
**HDD:** 30GB

実行環境の構成を図 5 にあらわす．ハードウェアの上にハイパーバイザーとして ESXi を使用している．ハイパーバイザーの上に VM として Locust サーバと K3s サーバ，WEB サーバがある．Locust サーバには負荷試験ツールとして Locust を実行している．K3s サーバにはリバースプロキシサーバとして Nginx を実行している．WEB サーバには WEB サイトとして WordPress を実行している．worker_connections はデフォルト値である 512 に設定し，


**図 5** 実験環境の構成

Locust のリクエスト数は 1 秒あたりに 10 ずつ増加させ，最大数は 300 とする．

### 実験結果と分析

負荷試験を行った結果のログデータから，クライアントのリクエスト数に対して worker_connections に設定した値が不足しているか可視化するために，ログデータの抽出とグラフ化を行う．抽出するデータはリクエスト数，秒間リクエスト量，秒間応答失敗量である．抽出したデータのグラフ化は Python ライブラリの matplotlib を用いて行う．

図 6 は，経過時間と抽出したデータの関係をあらわす．X 軸 Time Stamp は経過時間をあらわし，単位に s(second) をとる．Y 軸 Requests and Failures per second と赤の折れ線グラフは秒間リクエスト量をあらわす．同軸と青の折れ線グラフは秒間応答失敗量をあらわす．

図 6 のグラフから，秒間応答失敗量は秒間リクエスト量が 40.7 を上回った時点から急激に上昇している．負荷試験時に設定した最大リクエストに達してからは，秒間応答失敗量は急激な減少傾向にあるが 0 にはならずに一定で安定している．そのため，worker_connectionsn の設定値であるワーカープロセスの最大同時接続数の不足が考えられる．

## 6. 議論

本稿での提案手法では負荷試験ツールを使用して，Nginx を用いたリバースプロキシサーバの負荷試験を実施した．

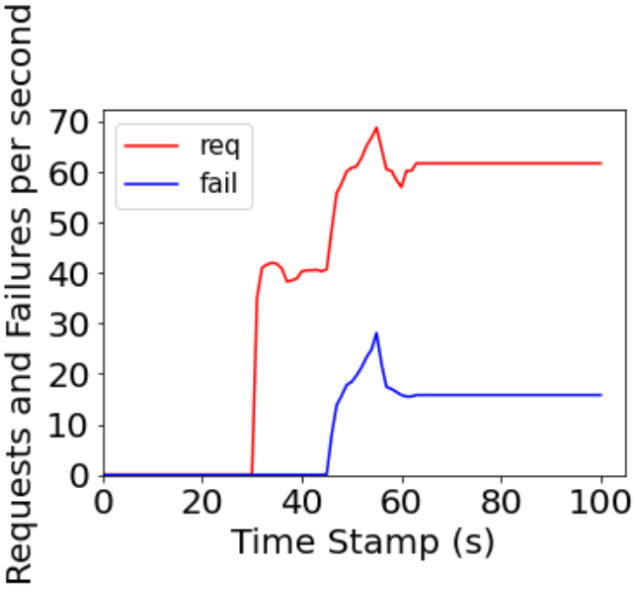


**図 6** 負荷試験結果のグラフ

実験の結果として，worker_connections をデフォルト値のままでは，秒間リクエスト量が 40.7 を超えた時点でリクエスト処理として正常に WEB サーバからデータを受け取ることが出来ない状態となった．本稿ではバックエンドサーバとして最低限の情報量の WordPress サーバを使用しているが，やり取りを行うデータ量が増加すれば処理に使用するリソースが増える．そのため worker_connections の最適値を求める場合は，リバースプロキシサーバ自体の性能を考慮することに加え，処理するデータ量も考慮する必要がある．

本稿で変更した worker_connections 以外にも，変更することで Nginx サーバのパフォーマンスに関係のある設定項目が存在する．具体例として関連研究であげた研究 [1] で設定している設定項目の内の，client_max_body_size がある．client_max_body_size はクライアントリクエストボディの最大許容サイズを指定するための項目である．デフォルト値は 1 に設定されており，リクエストのサイズが 1 MB を超えると，Request Entity Too Large エラーがクライアントに返される[*4]．つまりバックエンドサーバから受け取り，クライアントへ送るデータサイズが 1 MB 以上の場合はこの設定値を変更する必要がある．この設定値はやり取りするデータサイズから求めることができる．

## 7. おわりに

本研究の課題は，worker_connections をデフォルト値から変更するか否かの判断にかかる時間であった．提案手法はこれを判断するために行う負荷試験の自動化であった．実験の結果から，worker_connections をデフォルト値

[*4] Module ngx_http_core_module - Nginx.org http://nginx.org/en/docs/http/ngx_http_core_module.html#client_max_body_size(2023 年 1 月 19 日)

のままでは，リクエスト数を増加させると秒間リクエスト量が 40.7 を超えた時点から”worker_connections are not enough”のエラーが発生した．

## 参考文献

[1] Wang, J. and Kai, Z.: Performance Analysis and Optimization of Nginx-based Web Server, *Journal of Physics: Conference Series*, Vol. 1955, No. 1, p. 012033 (online), DOI: 10.1088/1742-6596/1955/1/012033 (2021).