令和 5 年度

修士学位論文

クラウド上のコンテナで動作する Web アプリケーションの 性能推定手法の評価

Evaluation of Performance Estimation Methods for Web Applications in Containers on the Cloud

1265089　　荒武 佑磨

指導教員　　松崎 公紀

2024 年 2 月 1 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報学コース

要 旨

クラウド上のコンテナで動作する Web アプリケーションの 性能推定手法の評価

荒武 佑磨

近年，クラウド上でのコンテナベースのアプリケーション運用が増加している．アプリケーションの本番運用においては，アプリケーションの性能を正確に把握し，適切なコンテナインスタンスを選択することが重要である．しかし，クラウドは性質上，その性能が外的要因などによりランダムに変動してしまうため，クラウド上で正確な性能指標を得るのは非常に困難である．Sen らが 2021 年に発表した “Performance Testing for Cloud Computing with Dependent Data Bootstrapping” では，性能変動のある仮想マシン環境において性能指標を得る方法として，ブートストラップ法やブロックブートストラップ法を用いた手法が提案された．特に，ブロックブートストラップ法は，性能変動の内部依存性を考慮することができ，より高い精度で性能予測を行えることが示された．

本論文では，クラウド上で運用される WEB サイトの高精度な性能推定と、負荷テストのコストバランスをとることを目的とし，具体的な負荷テストとその結果に対して，ブロックブートストラップ法とブートストラップ法を適用する．これにより，性能推定の精度を保ちつつ，負荷テストのコストを大幅に削減することが可能であるかを示す．性能評価手法としてブロックブートストラップ法やブートストラップ法を用いた実験では，通常の負荷テストと比較してテストコストが 50%以下になりうることが分かった．

キーワード　クラウド，コンテナ，Web アプリケーション，負荷テスト，ブートストラップ法，ブロックブートストラップ法

Abstract

Evaluation of Performance Estimation Methods for Web Applications in Containers on the Cloud

Yuma Aratake

In recent years,an increasing number of container-based applications are operated on the cloud environments. In the production operation of these applications, accurately understanding of the performance to select the appropriate container instances is crucial. However, due to the nature of the cloud environments, the performance can fluctuate randomly due to external factors, making it extremely difficult to obtain accurate performance metrics on the cloud environments.

In "Performance Testing for Cloud Computing with Dependent Data Bootstrapping," by Sen et al. in 2021, a method using bootstrap and block bootstrap techniques was proposed for obtaining performance metrics in virtual machine environments. In particular, the block bootstrap techniques was shown to be capable of considering the internal dependencies of performance fluctuations, allowing for more accurate performance estimation.

This paper aims to achieve a high-precision performance estimation for websites operated on the cloud environments and to balance the cost of load testing. The author applies the block bootstrap and bootstrap methods to a specific set of load tests. The experiments demonstrates the possibility of significantly reducing the cost of load testing while achiving a similar accuracy of performance estimation, in particular the test cost could be reduced to less than 50%.

key words Cloud, Container, Web Application, Load Testing, Bootstrap, Block Bootstrap

目次

目次

図目次

第 1 章

はじめに

多くの組織が，インフラストラクチャ・アズ・ア・サービス（IaaS）を利用してクラウドベースのソリューションに移行する中，信頼性の高い性能テスト手法が求められている．アプリケーションの性能に関する正確な知識は，適切な仮想マシン・コンテナ設定の選択や効果的なクラウド弾力性ポリシーに不可欠である．しかしながら，ハードウェアリソースの競合や VM スケジューリングのランダム性など，クラウド環境固有の性能不確実性により，正確な負荷テストは困難である．

クラウド環境におけるアプリケーションの性能テスト結果からアプリケーションの性能をより正確に予測する手法について，これまでにいくつかの研究がある [1], [2]．性能評価における課題の一つは，正確な性能評価を行える十分な性能データを得るための．Sen らの研究 [2] は，データ数を増やすリサンプリング手法の一つであるブートストラップ法をこの問題に適用し，単純な手法よりも少ないデータからより正確な性能指標を求められることを示した．続く研究 [1] では，性能テストデータの時間的な内部依存性に着目し，ブロックブートストラップ法を提案し，より正確な性能指標を求められることを示した．

本研究では，Web アプリケーションに対する負荷テスト手法と性能推定手法について，比較検討を行う．本研究の目標は，負荷テスト結果の精度と，クラウドコンピューティング環境におけるテストコストのバランスをとることである．そのために，性能推定結果が十分に正確となるのに十分な負荷テスト量を，確実かつコスト効率良く特定する手法を提供する．実際に Amazon WebServices（AWS） 上での評価実験の結果，本研究で提案する負荷テスト手法を用いることで，最も推定に時間のかかる性能でも一般的な手法に比べ，2 分 1 以下のコストに抑えられる可能性が高いことが分かった．

第 2 章

性能推定精度向上のためのリサンプリング手法

負荷テストのコストを下げつつ推定の精度の高める方法として有力なのは，リサンプリングにより少数の負荷テストデータを水増しするアプローチである．ここでは，ブートストラップ法と，それを内部依存性を含むデータへ改善したブロックブートストラップ法について説明する．

2.1 ブートストラップ法

ブートストラップ法はリサンプリングの代表的手法の一つである．ここでは，性能テストデータの 90%tile 実行時間の 95%信頼区間を決定するのにブートストラップ法を用いる方法を示す．

n 回の繰り返し実行を伴う性能試験から得られた実行時間の集合を S とする．ブートストラップ法では，図 2.1 のように S から実行時間を無作為に抽出して，リサンプリング集合 S^* を構築する．このリサンプリングを x 回繰り返して $\{S_1^*, S_2^*, \ldots, S_x^*\}$ を生成し，これら x 個のリサンプル結果を学習等に用いる．

リサンプリング結果を用いて，性能テストデータの 90%tile 実行時間の 95%信頼区間を求めるには以下の計算を行う．リサンプリングの各結果 S_i^* に対して，その 90%tile を計算したものを θ_i とする．リサンプリングに集合の分布が母集団の分布を適切に近似していると仮定すると，得られた集合 $\{\theta_1, \theta_2, \ldots, \theta_x\}$ は母集団の 90%tile 実行時間 θ の近似と見な

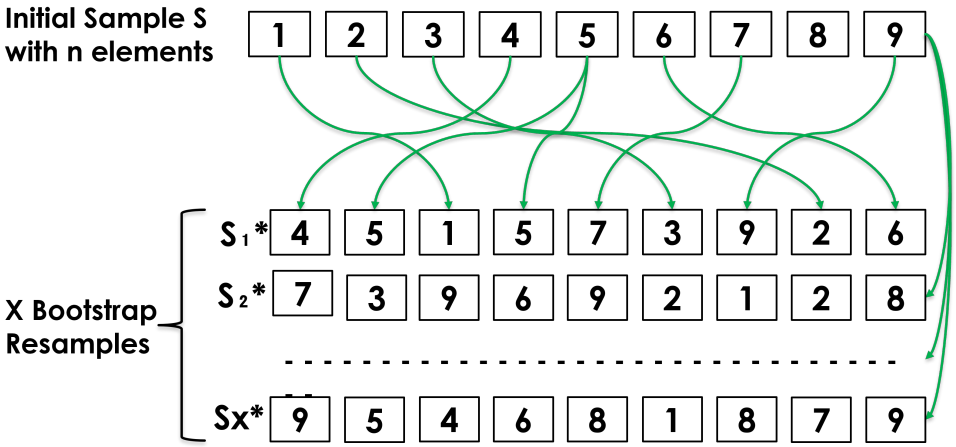

図 2.1　ブートストラップのリサンプリング

せる．したがって，集合 $\{\theta_1, \theta_2, \ldots, \theta_x\}$ の分布の中心 95%信頼区間を，θ の 95%信頼区間とする．

ブートストラップ法が適切に機能するためには，以下の条件が必要である．

- パラメータ x を十分大きくとる．一般に，x は 1000 より大きくする．
- S からのリサンプリングが，S が実際に得られた際の状況に酷似している．

しかしながら，Web アプリケーションの性能テストデータは上記 2 つ目の条件を十分に満たすとは言えない．なぜならば，性能テストデータは通常，内部依存性や季節性を持つ時系列データであるからだ．

2.2　ブロックブートストラップ法

ブロックブートストラップ法は，データの時間的な内部依存性を考慮したリサンプリング手法である．ブロックブートストラップ法では，ブートストラップ法と同様に無作為な選択を用いて S から S^* を構築する．ここで，各選択肢において，一つのデータのみ選択する

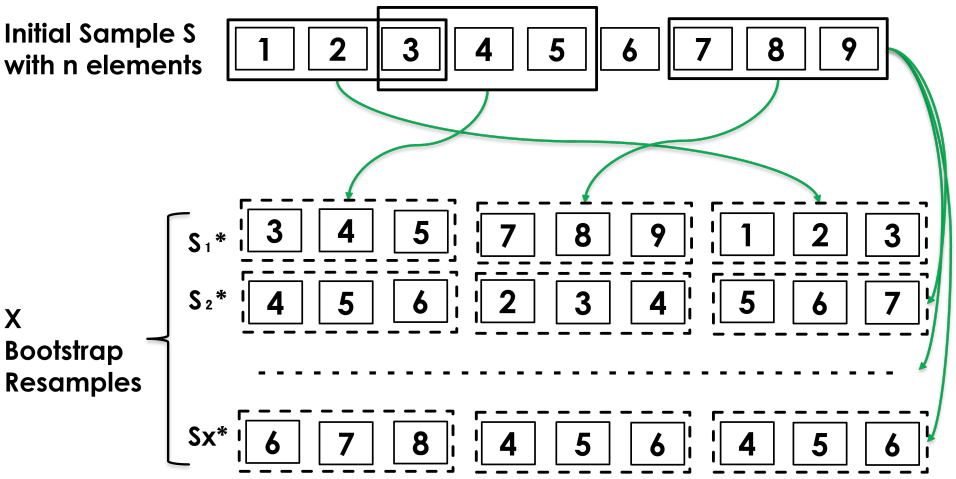

図 2.2　ブロックブートストラップのリサンプリング

のではなく，連続したデータのブロックを選択する．データ内の連続するブロックを選択することで，データ内の内部依存性が保持された形でサンプリングを行うことができる．1 つのブロックが b 個の要素からなるとき，n/b 回のランダムな選択を行って S^* を求める．図 2.2 はブロックブートストラップの手順を示している．基本的なブートストラップと同様に，x 個の再標本を構築して，これらの再標本を用いて信頼区間を求める．

上記手法の説明から明らかなように，ブロックのサイズは重要なパラメータである．ブロックが小さすぎる場合には，内部依存性が正しく再標本化できない可能性がある．逆にブロックが大きすぎる場合には，再標本化によって得られるデータの種類が不足する可能性がある．

第 3 章

性能推定手法

3.1　性能推定手法の概要

1 章の冒頭で述べた，負荷テストに関する問題を解決する手法を紹介する．sen ら [1] の研究で用いられた性能推定手法では，テストコストを削減するために，前述の負荷テストの説明でもあった定期的かつ断続的な負荷テストの実行戦略を採用している．テストの停止条件は，試行回数が多いときには真の平均に近く，試行回数が増えるにつれて真の平均に近くなるという傾向に基づいている [3]．クラウド上の WEB アプリケーションに対する負荷テストでは，この傾向は「クラウドでの多数の実行による平均がすべての潜在的な変動をカバーし，真の平均に近い場合，さらに相当数の実行を追加しても平均の値を大きく変えることはない」と言い換えることができる．この傾向に基づき，本手法ではテストから得られる性能結果が実行回数を大幅に増やしても変化しない場合にテストを停止するしても良いと判断している．本手法では，Politis と White によって開発されたブロックサイズを自動的に選択する方法論を採用している [4]．直感的には，この自動選択は，無視できない自己相関を提供する最小のブロックサイズを使用して「最適」なブロックサイズを見つける [4]．自動選択技術を採用することで,「最適」なブロック・サイズは，ブロック・ブートストラップがオリジナル・サンプルと同程度の内部データ依存性を保持することを可能にする．本手法では，各ブートストラップについて，この自動ブロックサイズ選択が行われ，各性能テストが独自のブロックサイズを使用するようになっている．本研究では，応答時間を入力として実験を行う．これは Web アプリケーションの性能を推定する上で一般的に用いられる性能であるためである．本手法では，一日分の応答時間のデータセットを入力として一日を通し

ての性能と，各時間のデータセットを入力として各時間の性能を推定する．

3.2　性能推定手法の詳細

本研究では，比較のためブートストラップ法とブロックブートストラップ法を用いて性能推定を行う．本節では，ブロックブートストラップ法を用いて，一日を通しての性能の 97%tile を推定する場合の方法について述べる．

負荷テストの結果から得られたデータを 2 週間分の性能推定用のデータと 2 週間分のグランドトゥルース用のデータに分割する．性能推定のデータの内，1 日分を 1 データとしてブロックブートストラップ法を用いてサンプリングを行う．このとき，一日毎の性能の誤差を調べるため，初めは 1 日分と 2 日分のサンプリング結果からそれぞれ 97%tile を求める．得られた 2 つの 97%tile 値の相対誤差を式 3.1 で求め，その誤差があらかじめ設定した最大許容誤差を上回っていた場合，1 日分増やして n 日分と $n+1$ 日分を比較というように 1 日分づつ増やしていき，性能推定に十分な日数を推定する．比較した結果の誤差が事前に定義した最大許容誤差を下回った時点で，性能推定に十分な日数のデータが得られたと判断して，性能推定を停止する．

本研究では，実際にどの程度の精度で推定が行えていたかを，最大許容誤差を下回るまでに利用した日数分の性能推定用のデータと，グランドトゥルース用のデータを用いて以下の式 3.1 に則り計算する．

$$\mathrm{err} = \left| \frac{\mathrm{Perf}_{\mathrm{Metior}} - \mathrm{Perf}_{\mathrm{true}}}{\mathrm{Perf}_{\mathrm{true}}} \right| \times 100\% \tag{3.1}$$

第 4 章

負荷テストの計画と実施

4.1　負荷テストの概要

本研究の目標は，性能推定精度と，クラウド環境におけるテストコストのバランスをとることである．そのために，性能推定結果が正確となる十分な負荷テスト量を，確実かつコスト効率良く特定する手法を提供する．

本研究では，負荷対象の Web アプリケーションとして，ログイン機能や，特定のデータ群の一覧をデータベースから取得して表示するような一般的な機能を持った Web アプリケーションを実装した．

テストシナリオとしては，ユーザがログインを行い，画面遷移を行った後に，登録された TODO の一覧を表示するという一連の流れを想定した．

負荷テスト時にかける負荷量の推移は図 4.1 の通りである．一般的な Web サイトを参考に，主に活動している人が多い時間帯，特に出勤時や帰宅後の負荷量が多くなるように負荷量の増減を設定した．ピーク時で Web アプリケーションからのレスポンスタイムが 3 秒程度になるように同時アクセス数を 200 に設定した．

負荷テストを実施した期間は 4 週間で，頻度としては毎日一時間毎，1 回の負荷テストを行う時間は 3 分間とした．性能を測定する頻度は 1 秒間に一回で，本研究では負荷量を確保するために負荷を掛ける用のコンテナタスクを二つ起動するように設定したため，一回の負荷テストで最大 360 個の性能測定結果が得られる．

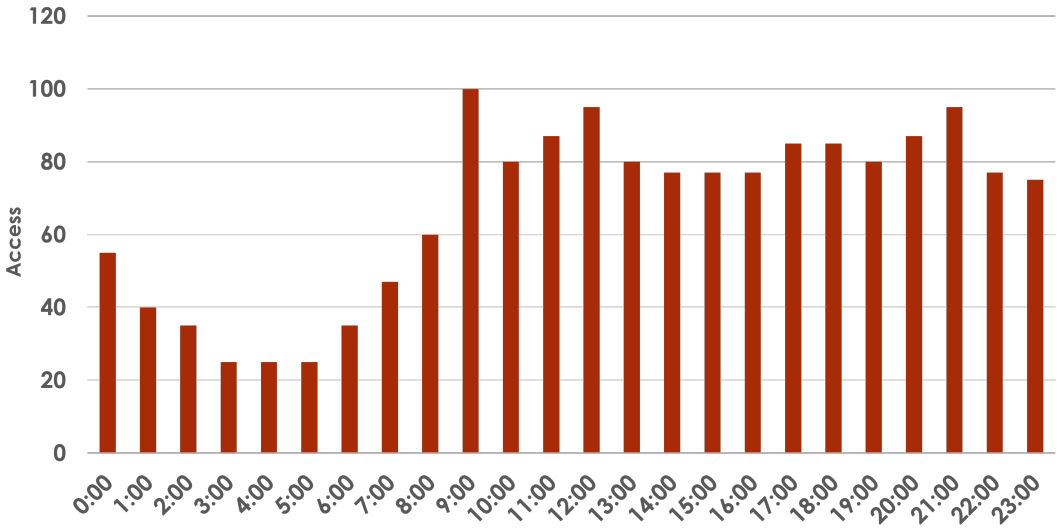

図 4.1　負荷量の推移

4.2　負荷対象 WEB アプリケーション

本研究を行う上で，一般的な WEB アプリケーションとして，ログイン機能や，特定のデータ群の一覧をデータベースから取得して表示するような機能を持った WEB アプリケーションを実装した．今回のテストシナリオで用いた機能のデータフロー図を 4.2 に示す．また，TODO リストを表示する画面を図 4.3 に示す．データベースとの連携が必要になることで CPU 使用率が上昇し，アクセス数に合わせて性能に有意な差が生まれると考えられる．実際に，図 4.1 に示す負荷量の推移に合わせて，Web アプリケーションの性能に有意な差が生まれた．

4.3　負荷テスト環境

負荷テスト対象の環境を図 4.4 に示す．これは負荷テストの対象となる WEB サイトをデプロイした環境を示している．コンテナクラスターには ECS を使用し，コンテナを管理する仮想マシンには Fargate を選択した．WEB サイト用のリレーショナルデータベースとして RDS を使用し、コンテナイメージを push するコンテナレジストリとして ECR を使用

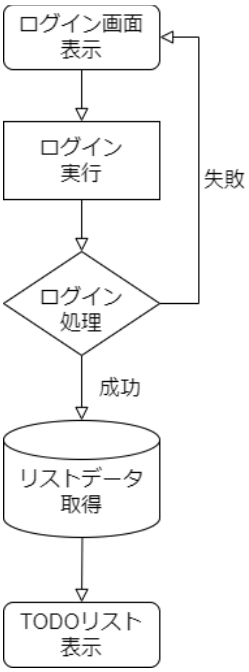

図 4.2　テストシナリオで用いた機能のフローチャート

した。また、コンテナへの安定した接続のためアプリケーションロードバランサー（ALB）を使用した．図 4.4 の下部以外は，コンテナを用いた基本的な本番環境の構成になっている．図 4.4 の下部の Lambda と S3 で連携している部分に関しては，負荷テスト中のコンテナの CPU やメモリのメトリクス情報を取得する為の構成である．

負荷を掛ける環境を図 4.5 に示す．負荷を掛ける環境は AWS から提供されているテンプレートと AWS CloudFormation を用いて構築した．負荷テスト時の動きとしては，あらかじめ設定された時間に，負荷対象環境にアクセスを行い負荷を掛ける ECS タスクが起動する．その後，アクセスを行った結果を Lambda を通じて S3 に保存される．負荷テストの設定は，本環境構築時に用意される専用の WebUI と Jmeter を用いて行った．

4.3　負荷テスト環境

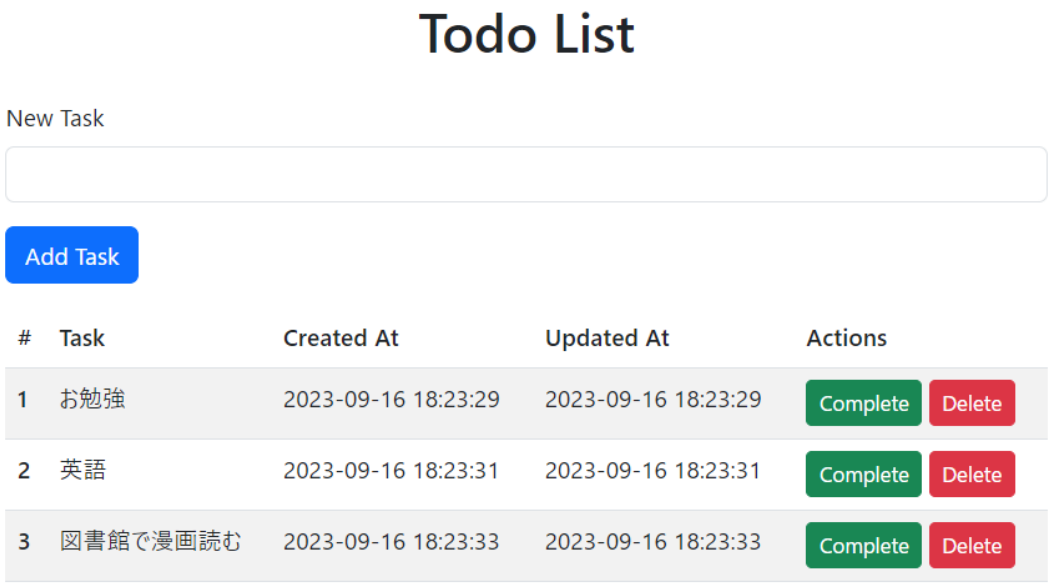

図 4.3　TODO リスト画面

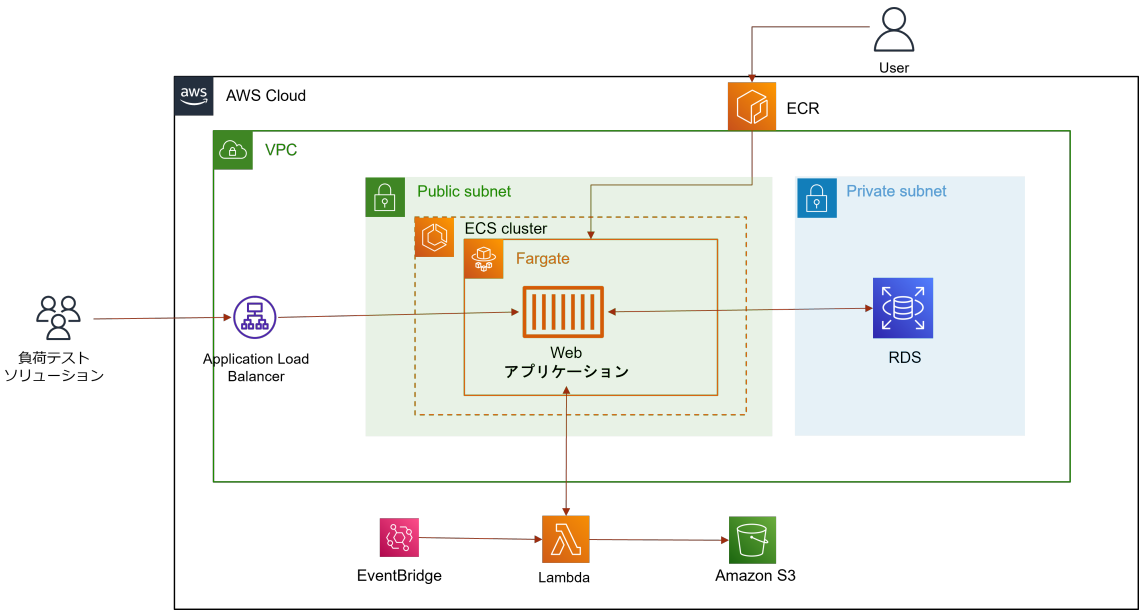

図 4.4　負荷テスト対象の環境

4.3　負荷テスト環境

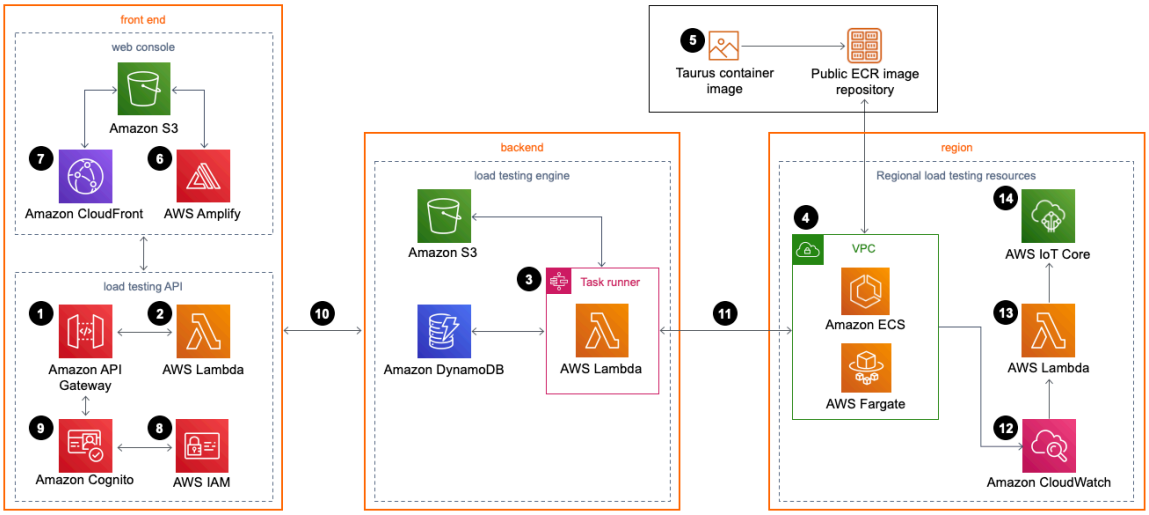

図 4.5　負荷テスト実施環境 [5]

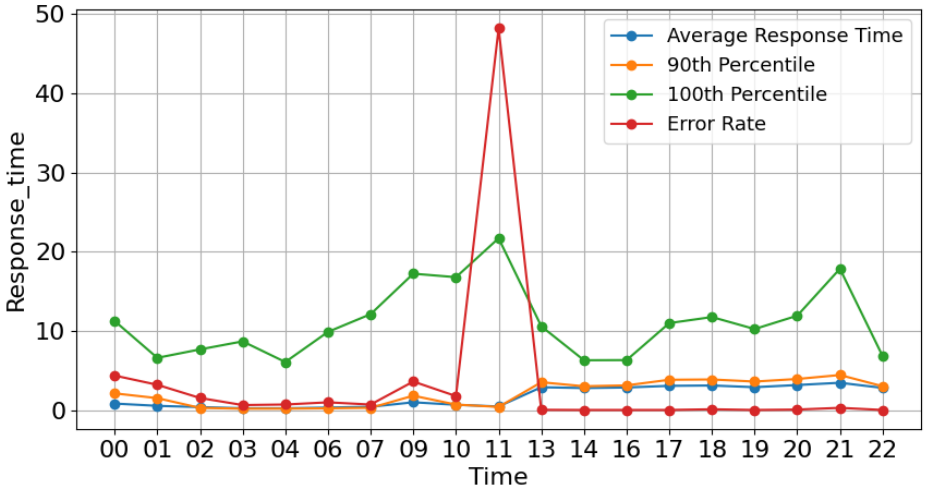

図 4.6　一日の各性能の推移（初日）

4.4　負荷テスト結果

負荷テスト 4 週間の内の初日，2 日目，15 日目，最終日のデータを図 4.6，4.7，4.8，4.9 に示す．前述した負荷量の推移と比較して，実際に負荷量に合わせて性能が変動していることが分かる．しかし，初日や，二日目の特定の時間帯のように負荷テストの開始直後は性能が安定しない時間帯が存在したが，その後は安定して負荷テストが行えていることがわかる．特定の時間の日別の推移を図 4.10，4.11，4.12 に示す．1 日分のデータを示した図 4.8 や図 4.9 と合わせて考えると，負荷が少ない時間帯に比べ，負荷が多い時間帯の方が 90%tile と平均値の差が開いており，性能の変動が大きいことが分かった．

4.4　負荷テスト結果

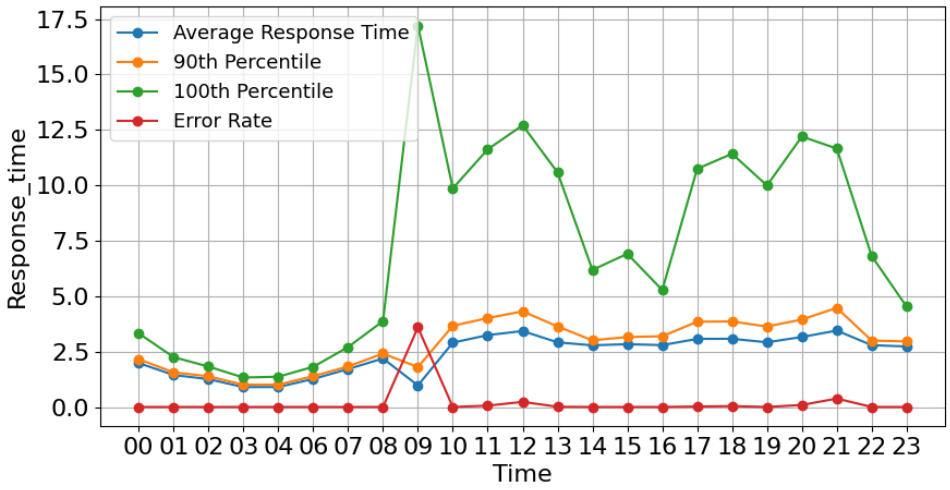

図 4.7　一日の各性能の推移（2 日目）

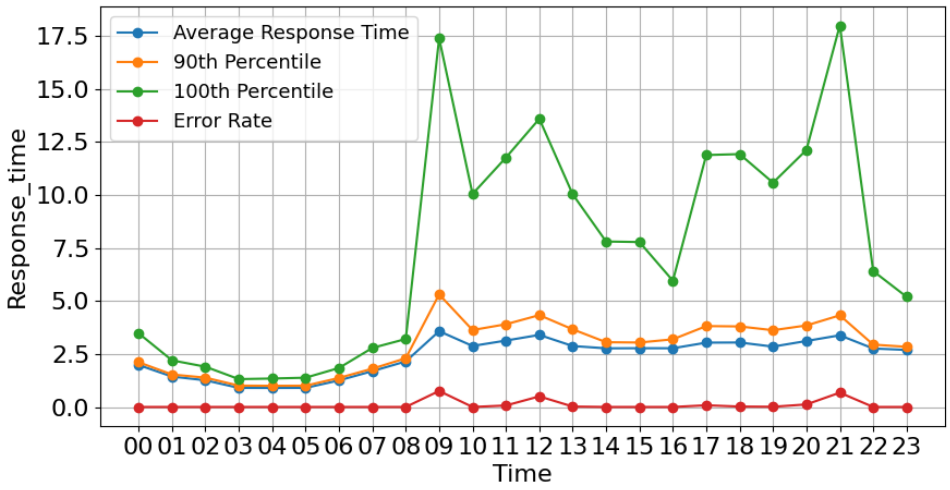

図 4.8　一日の各性能の推移（15 日目）

4.4　負荷テスト結果

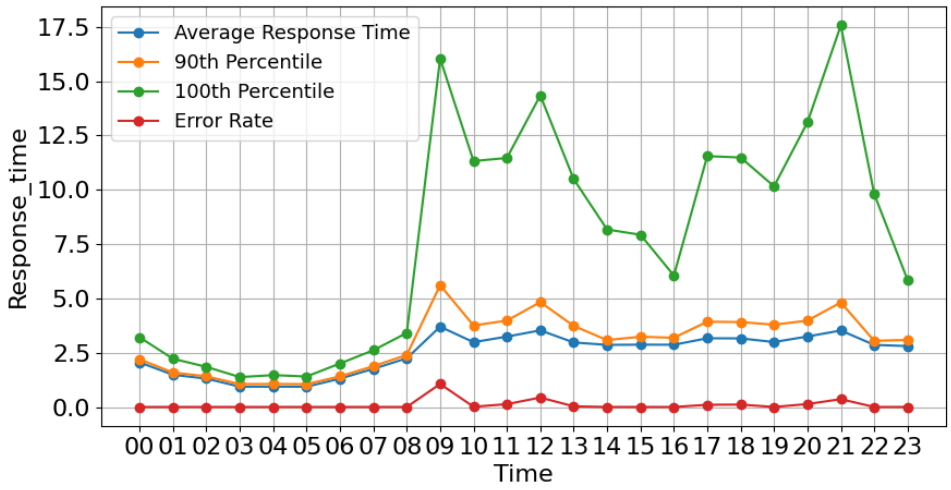

図 4.9　一日の各性能の推移（28 日目）

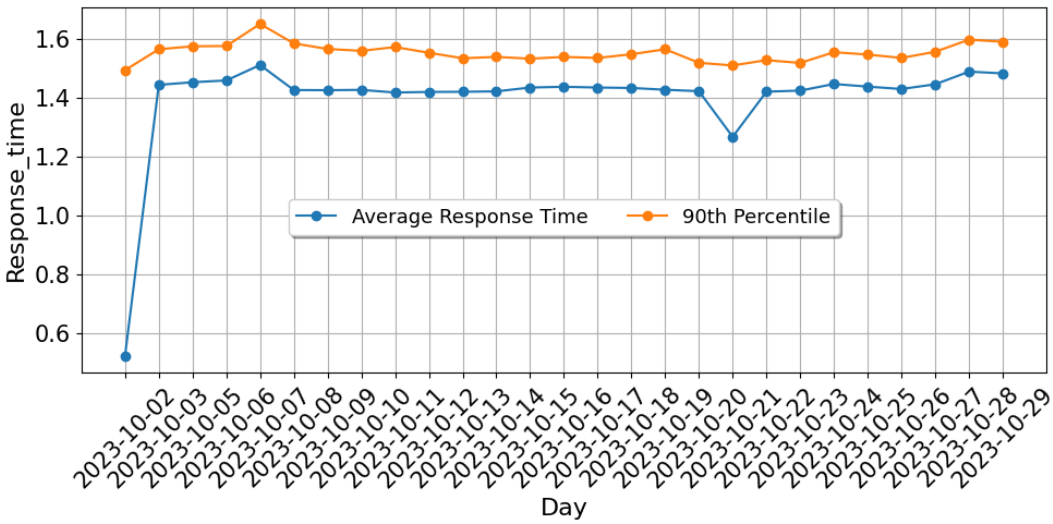

図 4.10　各日の 1 時時点の性能

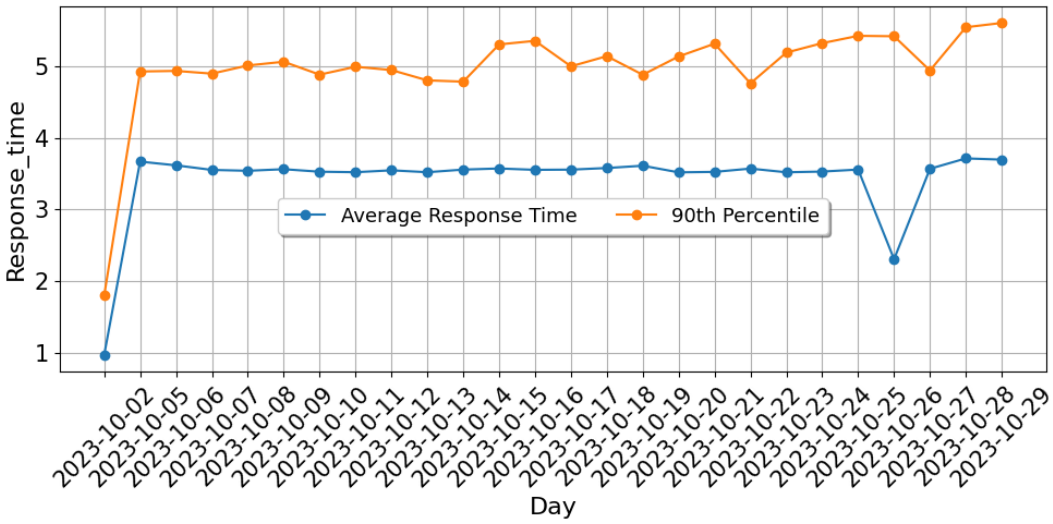

図 4.11　各日の 9 時時点の性能

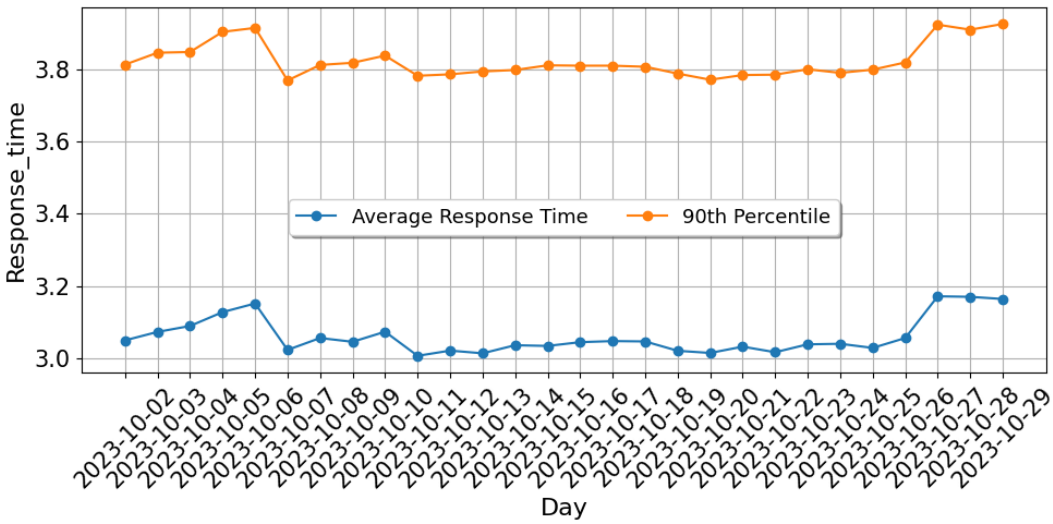

図 4.12　各日の 17 時時点の性能

第 5 章

性能推定手法の評価

5.1　一日を通しての性能推定結果

本研究で推定した性能は，一日を通してのレスポンスタイムの各パーセンタイル値や，各時間におけるレスポンスタイムの各パーセンタイル値とした．これらは一般的に， Web アプリケーションのパフォーマンスの一貫性の把握や，システムのボトルネックの特定，スケーリングを行う前の指標などに用いられる性能である．

図 5.1 はブートストラップ法とブロックブートストラップ法を用いて最大許容誤差 3 パーセントで負荷テストデータの性能評価を行った結果を示している．図 5.1 が示すように，ブロックブートストラップ法が常に高い精度で推定を行えているのに対して，ブートストラップ法では，求めたいパーセンタイルの値が増えるにつれて，推定の精度が低くなっていることが分かる．

グラフの線に沿って記載されている日数は、負荷テストで十分な性能データが取得されたと判断された日数を表している．これを見るとブートストラップ法ではブロックブートストラップ法に比べ、早い段階で十分な負荷テストが実施できていると判断しており，98%tile 以上の推定では，基準精度である 97%を下回っているにも関わらず，十分な負荷テストを行えていると判断してしまっていることが分かる．

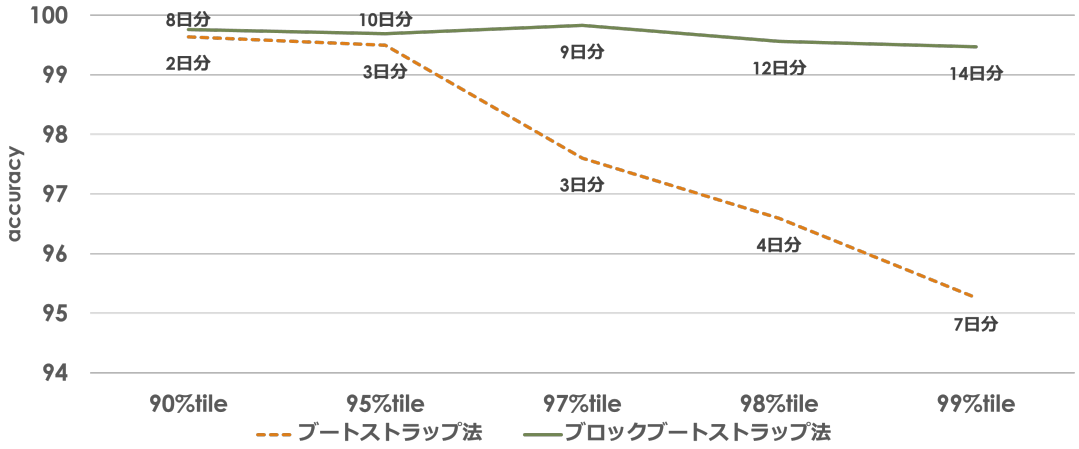

図 5.1　一日を通しての性能の推定結果

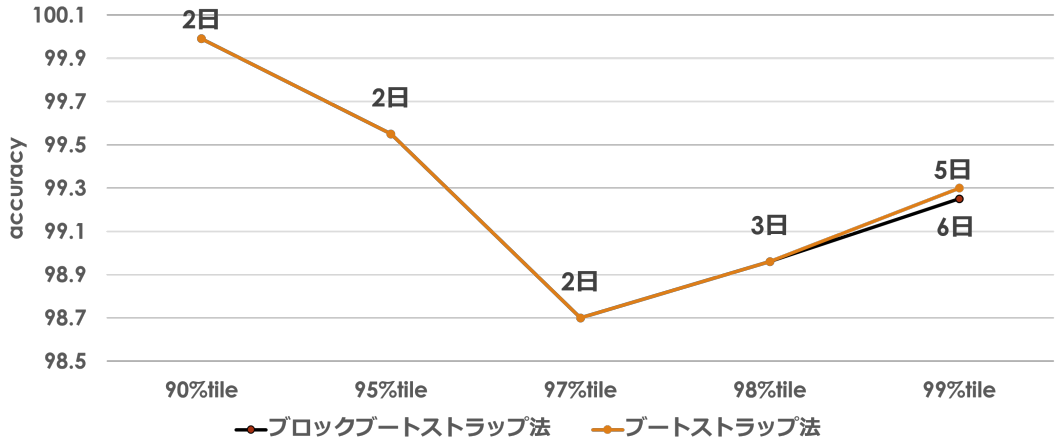

図 5.2　特定の時間帯の性能推定結果（1 時〜2 時）

5.2　特定の時間帯の性能推定結果

図 5.2，図 5.3 は各時間の性能データに対して，ブートストラップ法とブロックブートストラップ法を用いて最大許容誤差 5 パーセントで性能推定を行った結果を示している．図 5.2，図 5.3 が示すように，負荷量が少ない時間帯に比べ，負荷量が多い時間帯の方が性能の推定に必要な日数が多いことが分かった．また，ブロックブートストラップ法とブートスト

5.3　データ数を絞った特定の時間帯の性能推定結果

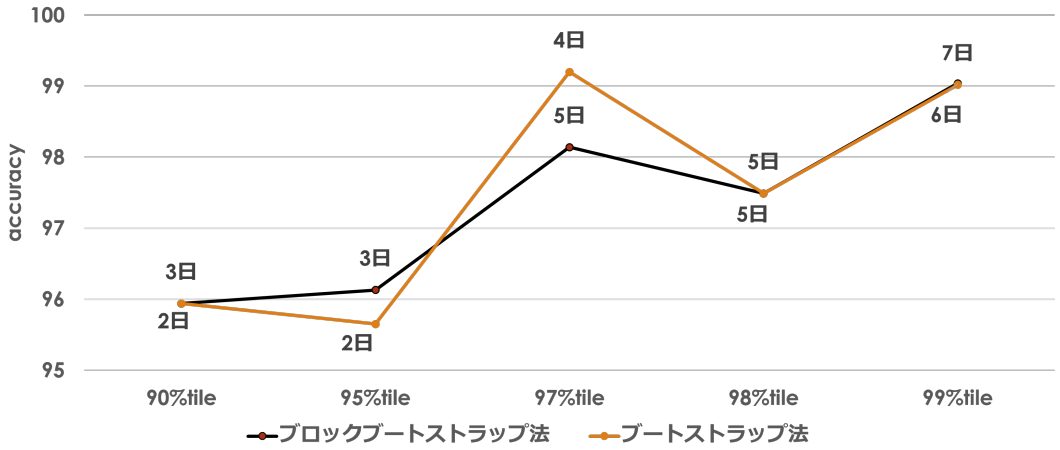

図 5.3　特定の時間帯の性能推定結果（9 時～10 時）

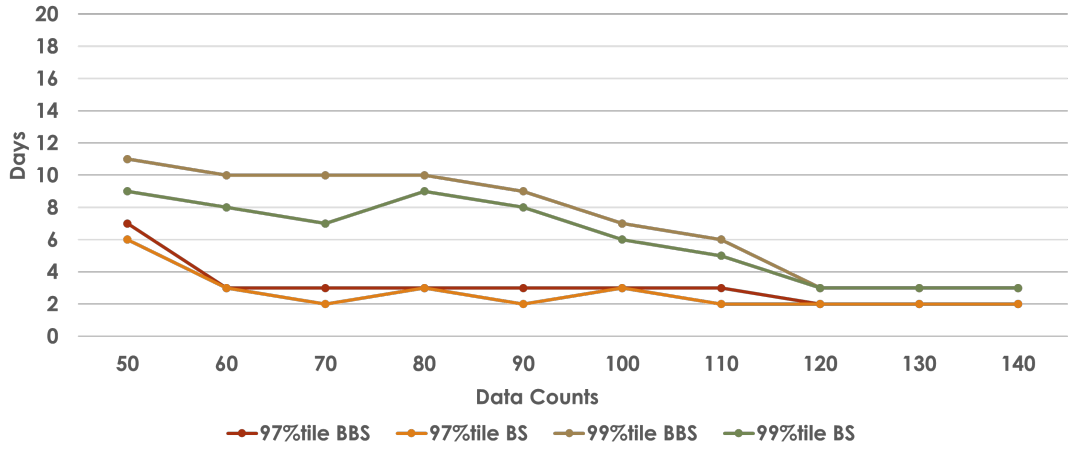

図 5.4　データ数を絞った特定の時間帯の性能推定結果（1 時～2 時）

ラップ法で推定精度に大きな違いは見られなかった．このことから，各時間の精度を推定する場合は，性能の推定に必要な日数が少ないブートストラップ法の方が各時間の性能推定には適しているという事が分かった．

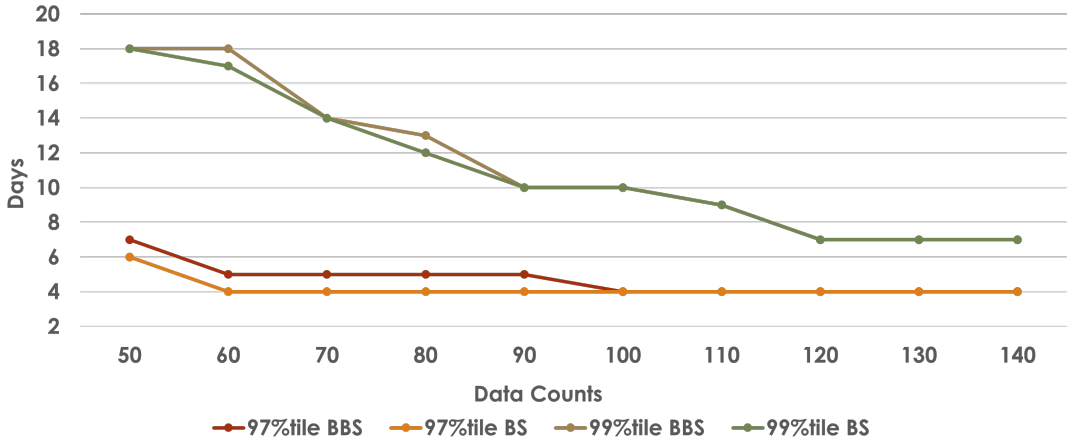

図 5.5　データ数を絞った特定の時間帯の性能推定結果（9 時〜10 時）

5.3　データ数を絞った特定の時間帯の性能推定結果

図 5.4、図 5.5 は，利用するデータ数を絞って各時間の性能推定を行った結果を示している．図 5.4、図 5.5 が示すように，サンプリングに利用するデータ数を絞った場合でも，推定に必要な日数は増加したが，精度を落とすことなく性能推定を行えていることが分かった．また，ブロックブートストラップ法とブートストラップ法で推定精度に大きな違いは見られなかった．

第 6 章

考察

今回の研究では，通常の負荷テストと違い，1 時間ごとに 3 分間だけ負荷テストを実施した．これにより，クラウドのリソース使用量を大幅に抑えることができ，テストコストを大幅に削減することができた．

しかしながらこの設計には問題点もあった．1 時間ごとに負荷テストでのアクセス数を変化させているため，得られたデータの値が急激に変化するような断続性が生まれた．また，1 時間ごとに負荷テスト用のタスクの起動や停止を繰り返しているため，稀に負荷テスト用のタスクが上手く起動せずテストが失敗している時間帯があることが分かった．特に負荷テストの実施を初めて数日間はそういった傾向が顕著にみられたが，日数を重ねるごとに安定してテストを行えるようになっており，最後の 1 週間程度はほとんど失敗なしにテストを行えていた．

本研究では，テストが失敗している時間帯に関してはテストデータに含めずに性能評価を行った．性能推定の結果に関しては，問題なくブロックブートストラップ法が適用できていることから，テストデータの集合に断続性が見られる場合や，1 日ごとのデータ数が違っている場合でも問題なく時間的な内部依存性を考慮することができること分かった．このことから．今回の研究で扱った大幅に負荷テスト時間を削減する手法でも，ブロックブートストラップ法と合わせることで，問題なく性能推定を行えることが分かった．

一日を通しての性能推定では，ブートストラップ法と比較して，ブロックブートストラップ法の方が高い精度で性能推定を行えていることが示されていた．実際に図 5.1 でのグランドトゥルースデータとの相対誤差による精度を見てみると，ブートストラップ法では，基準精度である 97%を下回っているにも関わらず，十分な負荷テストを行えていると判断してし

まっていることがわかる．これでは，負荷テストの誤った早期終了を招いてしまうことが考えられる．そのため，本研究で得られた時間的な内部依存性の高いデータから性能推定を行う場合，ブロックブートストラップ法の方が適していると考えられる．

特定の時間帯の性能推定では，ブートストラップ法とブロックブートストラップ法で推定精度に大きな違いは見られなかった．このことから，今回の負荷テストで得られた時間毎の負荷テストデータにおいては，時間的な内部依存性が低いことが考えられる．

今後の課題の 1 つは，ブロックブートストラップ法で推定を行う際の推定に必要な日数を減らすことである．ブロックブートストラップ法では，ブートストラップ法に比べ十分な精度が得られると判断するまでの日数が多く，ブートストラップ法の精度が基準精度を上回っている 90%tile の日数と比較して，4 倍の日数を必要としてしまっている．このため，推定までの日数を短くすることが課題である．

また，より現実的で複雑な機能を持った Web アプリケーションに対しても同様の手法が適用できるかを調べていくことも重要な今後の課題である．本研究で負荷テストの対象とした Web アプリケーションは，基本的な機能のみを備えたシンプルなものであった．しかし，実際の Web アプリケーションは，複雑な機能を持っていることが多く，そのような複雑な機能を持った Web アプリケーションに対しても同様の手法が適用できるかを調べていくことも今後の重要な課題である．

第 7 章

まとめ

本研究では，クラウド上で運用される Web サイトの性能推定を目的とし，具体的な負荷テストとその結果に対して，ブロックブートストラップ法やブートストラップ法を適用し．負荷テストを早期終了するための適切なタイミングを推定する手法を提案した．

その結果，本研究で得られたような，時間的な内部依存性の高い一日を通しての負荷テストデータに対してはブロックブートストラップの方が日数はかかるが，ブートストラップ法に比べ高い精度で性能推定が行えることが分かった．これにより，99%tile のように値のブレ幅が大きい性能を調べたい場合でも，負荷テストの誤った早期終了を防ぐことが可能であることが示された．

時間毎の性能推定では，ブロックブートストラップ法とブートストラップ法で推定精度に大きな差は見られなかったが，ブロックブートストラップ法の方が推定に日数を多く使っていることが分かった．そのため，時間的な内部依存性の少ない時間毎の性能推定においてはブートストラップ法の方が適していることが示された．

今回の実験において，最も推定に日数を必要とした 1 日を通しての性能推定における 99%tile の推定でも，負荷テストや性能テストにおける十分な日数とされている 4 週間 [2] と比較して，約 2 分の 1 の日数で推定を行えていた．さらに，本研究で行って負荷テストでは，一時間の内 3 分間だけテストを行うという手法を用いているため，一般的な負荷テスト手法と比較して，テストコストを 2 分の 1 以下に削減することが可能であると考えられる．

謝辞

本研究を行うにあたり，指導教員である高知工科大学情報学群の松崎公紀教授には様々なご指導をいただきました．心より感謝申し上げます．また，本論文の副査を引き受けていただいた横山和俊教授，高田喜朗教授に心より感謝申し上げます．本当にありがとうございました．

参考文献

[1] Sen, H, Tianyi, L, Palden, L, Jaewoo, L, In, K. and Wei, W.: Performance Testing for Cloud Computing with Dependent Data Bootstrapping, *2021 IEEE/ACM International Conference on Automated Software Engineering* (2021).

[2] Sen, H, Glenna, M, John, S, Wei, W, Lori, P. and Mary, L.: A Statistics-Based Performance Testing Methodology for Cloud Applications, *In Proc. of ACM Joint Meeting on European Software Engineering Conference. and Symp. on the Foundations of Software Engineering* (2019).

[3] Frederik, M, Cornelis, K, Hendrik, P. and Ludolf, E.: A Modern Introduction to Probability and Statistics, *Springer Science & Business Media* (2005).

[4] Dimitris, N, Politis. and Halbert, W.: Automatic Block-Length Selection for the Dependent Bootstrap, *Econometric Reviews*, 23(1), 53–70 (2004).

[5] Amazon Web Services, Inc.: Distributed Load Testing on AWS, `https://aws.amazon.com/jp/solutions/implementations/distributed-load-testing-on-aws/` (2023).

[6] 荒武 佑磨, 松崎 公紀:クラウド上のコンテナで動作する Web アプリケーションの性能推定手法の評価, **第 *65* 回 プログラミング・シンポジウム** (2024).