アクセス頻度予測に基づく 仮想サーバの計画的オートスケーリング

三宅 悠介[1,a)]　松本 亮介[1,b)]　力武 健次[1,2,c)]　栗林 健太郎[1,d)]

概要：従量課金を主としたクラウドサービスでは，処理性能を保ちつつ必要最小限数の仮想サーバで運用し利用料金を抑制することがWebサービスの運営者の課題である．しかしながら，複雑化したWebアプリケーションにおいて適切なオートスケーリング契機の基準決定は困難である．本報告では，Webアプリケーション構成の複雑さに依存しない指標として一定時間あたりのアクセス頻度であるスループット値を用い，Webサービス運用者が経験的に把握しているアクセス頻度に影響を与える突発的な要因を取り込むことで予測精度を向上させたアクセス頻度予測モデルから，クラウドサービスの利用料金を考慮した仮想サーバ台数を算出する計画的オートスケーリングを提案する．提案手法により，サービス運用から得られる突発的なアクセス頻度の変動要因を予測に反映しながら，仮想サーバごとの負荷が一定に保たれた安定かつ効率的な運用が可能となる．

Scheduled Autoscaling of Virtual Servers by Access Frequency Prediction

Yusuke Miyake[1,a)]　Ryosuke Matsumoto[1,b)]　Kenji Rikitake[1,2,c)]　Kentaro Kuribayashi[1,d)]

Abstract: Web service providers need to run minimal number of virtual servers while maintaining the processing performance, to suppress the usage fee of the cloud computing services with metered billing. Determining appropriate autoscaling indicator, however, is difficult for complicated Web applications. In this report, we propose scheduled autoscaling of the virtual server which calculates the number of virtual servers considering the usage fee from the predicted access frequency, by using the throughput value as an indicator which is unaffected by the Web application complexity. Our method maintains the prediction accuracy by reflecting unsteady factors affecting access frequency based on the Web service providers expertise, and enables stable and efficient operation while keeping steady load of virtual server.

1. はじめに

クラウドサービス [12] が普及を続けている [20]．Webサービスは提供するサービスの成長や利用者の動向に合わせてアクセス頻度が変動するため，サーバ群の処理能力を随時変更可能なクラウドサービスとWebサービス運用は親和性が非常に高い．クラウドサービスは使用する仮想サーバ性能に応じた単価に利用時間を乗じて利用料金が決まる従量課金が主流であるため [1]，処理性能を保ちつつ必要最小限数の仮想サーバで運用し利用料金を抑制することがWebサービスの運営者にとって重要な課題である．

そこでクラウドサービス事業者は変動するサーバ需要に追従するための動的なキャパシティプロビジョニング [2] を実現するオートスケーリング機能 [5] を提供している．

オートスケーリングにはCPU使用率などのリソース変動を契機としたもの，指定した時刻や間隔を契機に行うものがある．リソース変動を契機とする場合，利用者はCPU

[1] GMO ペパボ株式会社 ペパボ研究所
Pepabo R&D Institute, GMO Pepabo, Inc., Tenjin, Chuo ku, Fukuoka 8100001 Japan
[2] 力武健次技術士事務所
Kenji Rikitake Professional Engineer's Office, Toyonaka City, Osaka 5600043 Japan
[a)] miyakey@pepabo.com
[b)] matsumotory@pepabo.com
[c)] kenji.rikitake@acm.org
[d)] antipop@pepabo.com

使用率や超過期間などに閾値を設け，条件を満たした時点で，事前に指定した仮想サーバ台数に調整する．指定した時刻や間隔を契機に行う方式では，利用動向の推移を予想して指定時刻や間隔で調整を行う．これらは契機となるリソース変動指標と Web サービス全体の性能との関係や利用動向の推移が把握できる規模のうちは有効だが，一定以上の規模に成長した Web サービスにおいては Web アプリケーション構成の複雑化に起因する指標と性能間の関係性の複雑化，ならびに利用動向の多様化に起因する影響要因の増加により，これらの指標を経験的に決定することは困難となる．また，オートスケーリングによる仮想サーバ追加指示と起動完了までの時間差 [18] により，特に突発的なリソース不足を契機として仮想サーバを追加するような場合に，起動完了までの間，処理性能が不足する現象が発生しうる．

オートスケーリング契機の基準決定と仮想サーバ起動完了までの時間差の 2 つの課題を解決するためには，オートスケーリング契機の指標の決定が困難である規模の Web サービスにおいて，Web アプリケーション構成の複雑さに依存しない指標を用いて，利用動向の変動要因を加味した上で，クラウドサービスの利用料金を考慮したアクセス頻度予測に基づく仮想サーバの計画的オートスケーリングが必要となる．

提案手法では，1 台あたりの仮想サーバが一定時間に処理可能なアクセス頻度であるスループット値を指標として用いる．加えて，Web サービス運用者が経験的に把握しているアクセス頻度に影響を与える突発的な変動理由を要因として意味付けして学習に加えることで予測精度が向上することに着目し，時系列データを扱うニューラルネットワークのうち，長期間の時系列に対して予測精度の高い LSTM[8] を用いて過去のアクセス頻度並びにアクセス頻度に影響を与える要因である休祝日情報をクラウドサービスの標準的課金単位である 1 時間を粒度とした学習データによって予測モデルを構築する．次に，予測アクセス頻度を元に仮想サーバ台数を算出し，指定時刻によるオートスケーリング機能の設定を更新する．

提案手法の評価は以下の手順で行う．まず Web サービスのうち，アクセス傾向の異なる 2 つのサーバ群に対し，アクセス頻度予測並びに仮想サーバ台数算出を行うシステムを導入し，仮想サーバ台数の調整を行う．はじめにアクセス頻度に影響を与える要因有無による予測精度を比較し，アクセス頻度以外の要因を考慮することで予測精度が向上することを検証する．次に，提案手法によるオートスケーリング実施前後での処理能力と構成台数を比較し，仮想サーバ台数を最小限に抑えつつ，仮想サーバごとの負荷が一定に保たれ安定かつ効率的に運用されることを検証する．

本報告の構成を述べる．2 章ではクラウドサービスを用いた Web サービス運用における処理性能と利用料金の最適化を行う既存手法と課題について述べる．3 章では既存手法の課題を解決するための提案手法について，アクセス頻度とアクセス頻度に影響を与える要因による利用料金を考慮した仮想サーバの計画的オートスケーリングとその実装について述べる．4 章では提案手法の有効性の検証を行い，5 章でまとめとする．

2. クラウドサービスを用いた Web サービス運用における処理性能と利用料金の最適化の課題

クラウドサービスは従量課金が主流であるため，処理性能を保ちつつ必要最小限数の仮想サーバで運用し利用料金を抑制することが Web サービス運用者にとって重要な課題である．Web サービスを安定して運用するため，サーバ需要を見積もり，準備するキャパシティプロビジョニングを仮想化技術と組み合わせることで変動するサーバ需要に追従する動的なキャパシティプロビジョニング [2] が研究されており，クラウドサービス事業者は提供するクラウドサービス上でこれを実現するためオートスケーリング機能 [5] を提供する．

オートスケーリングは CPU 使用率などのリソース変動を契機とする手法と指定時刻や間隔を契機とする手法がある．これらの手法は，リソース変動と Web サービスの性能劣化の関係を把握できるアプリケーション規模や，Web サービス利用動向の推移が明確である場合において簡潔で容易に運用できる．しかしながら，一定以上の規模の Web サービス運用においては契機となるリソース変動や指定時刻を適切に決定することが困難になる．また，仮想サーバ追加指示と起動完了までの時間差に起因する不安定性も課題となる．

2.1 オートスケーリング契機の基準決定に関する課題

リソース変動を契機とするオートスケーリングの場合，Web サービスのサーバ構成は負荷状況の変動に応じて追従する．クラウドサービスの利用者は，「5 分の間に平均 CPU 使用率が 70 %（20 %未満）を超えると 2 つの仮想サーバを追加（削除）する」などリソース使用率や超過期間などに閾値を設け，条件を満たした時点で，オートスケーリングにより事前に指定したサーバ台数に調整される．

この方式では，スケーリングの契機となる適切な指標がクラウドサービスから提供され，Web サービス運用者がそれらの指標と Web サービス全体の性能との関係を把握できる場合において有効である．仮想サーバ上の各リソース使用率を始めとする提供されるべき指標が検討され [6]，クラウドサービスによって提供されており．一部のサードパーティによってミドルウェア関連の指標も提供されている [15]．

しかしながら，多層アーキテクチャ [17] やマイクロサービス化 [10] により複雑化する Web サービスのアプリケーションから得られるこれらの指標と Web サービス全体の性能との関係を把握し，適切な指標と閾値を決定することは困難である．

1 分以上の処理時間を想定するバッチ処理に対して，タスクの実行期限と優先順位をもとにリソース変動を見積もる手法 [13], [14] も提案されているが，リクエストごとの処理時間が短く，バッチ処理と比較してリクエスト種別の多くなるオンライン Web アプリケーションに対しては適用できない．

指定時刻や間隔を契機としたオートスケーリングの場合，計画的に Web サービスのサーバ構成を調整する．クラウドサービスの利用者は，「毎日 20 時に仮想サーバを 2 台追加（削除）する」など指定し，条件を満たした時点で，オートスケーリングにより事前に指定したサーバ台数に調整される．

この方式では，前述のリソース変動と Web サービス全体の性能との関係が把握できない状態であっても，Web サービス利用動向の時間ごとの推移が明確であれば有効である．

しかしながら，Web サービスの規模が大きくなり利用動向の種類が増えると，時間ごとの推移傾向は多くの要因によって構成されるようになり，Web サービス運用者の経験値のみで正しく見積もることは困難である．

そのため過去のサーバ需要傾向に基づき仮想サーバ需要を統計的に見積もる手法が提案されている [9]．この手法では曜日や時刻などの定常的な反復傾向を見出すため，休日などの定期的な要因や季節性の要因は考慮できるが，日本の祝日や Web サービス独自のキャンペーンのような不定期な要因は予測に取り込むことができない．

統計的，機械学習的な手法による予測精度を向上させるため解決するタスクの特性を良く表現した学習データが有効である [3], [21]．また，ネットワークのトラヒック予測は，休日の特性を考慮することにより精度が向上することが知られており [19]，こうした外部の要因がトラヒックの特性となることがわかる．インターネットのトラヒックを構成している Web サービスにおいても外部の要因による予測精度向上の可能性があると考えられる．

2.2 仮想サーバ起動までの時間差に起因する課題

仮想サーバ起動には一定の時間が必要なため，オートスケーリングによる仮想サーバ追加指示と起動完了までの時間差が課題となる [18]．特にリソース変動を契機とする手法では，サーバ需要が直近で不足する状況の場合では，起動完了までの時間差に起因する処理性能の不足が問題となる．特にリソース不足を検知した時点での処理性能と目標とする処理能力に乖離がある場合には Web サービスの不安定性が増大してしまうため，予測的な手法を組み合わせることでこの乖離を縮めておく必要がある．起動完了までの時間差の問題を解決するため，コンテナ技術 [7] を利用する手法もあるが，稼働中のシステムをコンテナ環境に移行するにはコストが高い．

ここまで述べたことから，既存手法の課題として以下の 3 点を挙げることができる．

(1) 一定以上の規模となった Web サービスにおいては，リソース変動や指定時刻といったオートスケーリング契機を Web サービス運用者が経験により適切に決定することは困難である
(2) 従来のオートスケーリング契機決定手法では，一定以上の規模となった Web サービスにおける性能指標選定や非定期な要因を踏まえた予測を行うことができない
(3) 仮想サーバ起動までの時間差により，追従的な構成変更では Web サービスの不安定性が増大する

3. 提案手法

Web サービス運用者の経験値のみでは，リソース変動と Web サービス全体の性能の関係性を把握できず，Web サービス利用動向の推移が明確でない一定以上の規模の Web サービス運用において，2 章で述べたクラウドサービスを用いたオートスケーリングに関する課題を解決するためには，以下の 4 つの要件を満たすことが必要である．

(1) Web アプリケーションの構成の複雑さに依存しない性能指標を用いる．
(2) 複雑な Web サービスの利用動向の変動を把握するために，定常性や非定常な変動要因を加味してオートスケーリング契機を高精度に自動予測する．
(3) 仮想サーバの起動完了までの時間差による不安定性を回避するために，予測的な構成変更を行う．
(4) 利用料金を最適化するために，オートスケーリングはクラウドサービスの課金単位時間を単位に行う．

3.1 アクセス頻度予測モデルの構築

提案手法では，複雑な Web アプリケーション構成のうち Web サーバ機能に限定することで，仮想サーバのスループット，すなわち仮想サーバが一定時間に処理可能なアクセス頻度を Web アプリケーションの構成の複雑さに依存しない性能指標として用いる．仮想サーバに対する個々の処理要求とその処理負荷は様々であるが，一定時間の処理要求を均すことで平均的なスループット，アクセス頻度が求められる．Web サービス運営者は安定したサービス運用が可能なスループットの目安を経験的に把握している．

そこで，提案手法では始めにアクセス頻度を予測し，目安となるスループットを保つことが可能な仮想サーバ台数を算出する．アクセス頻度の予測には時系列データを扱うニューラルネットワークである LSTM[8] を用いる．LSTM は入力と出力の次元数が異なることを許容できる [8] ため，

```
timestamp,count,holiday
2017-03-20 00:00:00,8942,1
2017-03-20 01:00:00,5528,1
2017-03-20 02:00:00,3262,1
2017-03-20 03:00:00,1946,1
2017-03-20 04:00:00,1291,1
2017-03-20 05:00:00,1273,1
2017-03-20 06:00:00,2278,1
2017-03-20 07:00:00,3730,1
2017-03-20 08:00:00,4891,1
```

図 1 アクセス頻度予測モデルの学習データ例
Fig. 1 Data examples of learning access frequency prediction model.

過去のアクセス頻度並びにアクセス頻度に影響を与える非定常な要因をもとに予測モデルを構築する．

非定常な要因には，ネットワークトラヒックに影響を与える休日情報 [19] に加え，日本の祝日や Web サービス独自のキャンペーン期間，利用者向けの一斉プッシュ配信期間といった Web サービス運営者が運用の中で獲得してきた Web サービスのアクセス頻度に影響を与えうる要因を加えることができる．

これらの情報をクラウドサービスの標準的課金単位である 1 時間を粒度とし，Web サービスの利用動向の最短定常性を確認できる単位である 24 時間を入力として 1 時間後のアクセス頻度を正解とする学習データによって訓練を行う．

なお，LSTM は入力ゲートに非線形な活性化関数が利用される [8] ことから，入力データが大きな値の場合に誤差逆伝播によるネットワークの重み更新が進まないため，入力を 0 から 1 までの範囲に正規化し，学習を収束させる．今回はアクセス頻度の値に対して正規化を行う．その他の要因については該当期間を 0 または 1 のフラグとして表現する．学習データの例を**図 1** に示す．

アクセス頻度予測モデルの構築には機械学習のライブラリである TensorFlow[16] をバックエンドとしたニューラルネットワークライブラリであり，LSTM 実装が提供されている Keras[11] を用いる．アクセス頻度予測モデルを Keras を用いて学習する Python コードを**図 2** に示す．

3.2 予測アクセス頻度を用いたサーバ台数算出

アクセス頻度の予測は，予測対象となる 1 時間分のアクセス頻度より前の 24 期分のアクセス頻度並びに非定常の要因を入力として予測を行う．出力のアクセス頻度は正規化されているため，学習時の正規化の際に取得した最小値ならびに最大値を用いて非正規化を行う．2 時間後以降のアクセス頻度を予測するためには予測済みの 1 時間後のアクセス頻度と前 23 期分のアクセス頻度を利用し，必要な期間分だけこれを繰り返す．

```
def run_training():
  X, y, min, max = load_dataset(FLAGS.time_steps)
  input_size = X.shape[2]
  if input_size > 1:
    mat_min = np.zeros([1, FLAGS.time_steps, input_size],
            dtype=np.float32)
    mat_min[:,:,0] = min
    mat_max = np.full((1, FLAGS.time_steps, input_size),
            1.0, dtype=np.float32)
    mat_max[:,:,0] = max
  (X_train, X_test), (y_train, y_test) = split_dataset(X,
          y, 0.1)

  with tf.Session() as sess:
    # train
    K.set_session(sess)
    model = Sequential()
    model.add(Lambda(tf_normalize, arguments={
          'min': min if input_size == 1 else mat_min,
          'max': max if input_size == 1 else mat_max
          }, batch_input_shape=(None, FLAGS.time_steps,
              input_size)))
    model.add(LSTM(FLAGS.hidden_units,
        return_sequences=False))
    model.add(Dense(1))
    model.add(Activation("linear"))
    model.compile(loss="mean_squared_error",
            optimizer="adam")
    model.fit(X_train, normalize(y_train, min, max),
            epochs=FLAGS.epoch, validation_split=0.05)

    # export
    keys_placeholder = tf.placeholder(tf.string,
            shape=[None])
    keys = tf.identity(keys_placeholder)
    tensor_min = tf.fill(tf.shape(model.output), min)
    tensor_max = tf.fill(tf.shape(model.output), max)
    inputs  = {'key': keys_placeholder,
            'recent_access_counts': model.input}
    outputs = {'key': keys, 'access_count': model.output,
            'min': tensor_min, 'max': tensor_max}
    export(sess, inputs, outputs, FLAGS.model_dir)
```

図 2 Keras によるアクセス頻度予測モデル構築の Python を使った実装例
Fig. 2 Implementation examples of building access frequency prediction model using Python and Keras.

次に，求めた予測アクセス頻度を元に仮想サーバ 1 台あたりのアクセス頻度を一定に保つために必要となる仮想サーバ台数を算出する．まず Web サービス運営者が経験的に把握している Web サービスを安定して運用できる目安となるスループット値を T[アクセス頻度/分] と定義する．

次に，以下の式からサーバ台数 S を算出する．

$$S = \begin{cases} P/60/T & (\geq L) \\ L & (\text{otherwise}) \end{cases}$$

ここで P[予測アクセス頻度/時] は非正規化済みの 1 時間

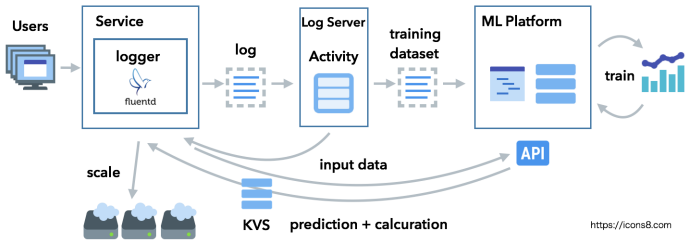

図 3 アクセス頻度予測と仮想サーバ台数算出の処理フロー
Fig. 3 Process flow for predicting access frequency and calculating the number of virtual servers.

あたりの予測アクセス頻度，L はサービス継続性の観点で導入したサーバ台数下限値である．

最後に，求めたサーバ台数を元に指定時間を契機とするオートスケーリングの設定を更新する．ここまでの処理を**図 3** に示す．

4. 実験と考察

提案手法の有効性を検証するため，3 章で構築したアクセス頻度予測モデル並びにサーバ台数算出システムを Web サービスのプロダクション環境に導入した．導入した Web サービスでは，前日分のアクセス頻度と非定常の要因をもとに当日分のアクセス頻度を予測し，時間ごとのサーバ台数を算出，指定時刻にオートスケーリングにより台数の変更を行う．

なお，今回用いた非定常の要因は休日情報にあたる．具体的には導入先の Web サービスにおいて，当日ではなく，翌日が休日か平日かによって特に夜間のアクセス頻度が異なる傾向が確認されており，非定常の祝日にも同様の傾向が見られることからこれを要因として採用した．具体的には該当期間の翌日が休祝日である場合に 0，それ以外を 1 として図 1 の形式でフラグとして学習データならびに予測時の入力に付与した．

また，導入先の Web サービスにはアクセス傾向の異なる 2 つのサーバ群，ここでは A 群と B 群が存在しており，検証はこの 2 群に対して行った．

本研究では，非定常の要因を加味したアクセス頻度予測による仮想サーバの計画的オートスケーリングの有効性を検証するため，以下の 2 点の観点で検証を行った．

(1) 非定常の要因の加味によるアクセス頻度予測の精度
(2) アクセス頻度予測を用いた計画的オートスケーリングによる処理性能と仮想サーバ台数の最適化

4.1 非定常の要因の加味によるアクセス頻度予測精度の評価と考察

本節では，アクセス頻度予測に影響を与える非定常の要因を学習データとして含めるかどうかによって，予測精度に差が見られるか検証する．精度評価指標として平均二乗誤差 [4] を用いた．平均二乗誤差は予測と正解の乖離の程度を表し，数値が小さいほど精度が高いと考えることができる．なお，与えられた学習データは 9:1 の割合で分割し，それぞれ学習，検証データとして利用した．

表 1 非定常の要因の加味によるアクセス頻度予測精度の比較
Table 1 Comparison of prediction access frequency by adding unsteady factors.

	非定常の要因を含まない	非定常の要因を含む
A 群	0.0195460487157	0.0193729232997
B 群	0.043068844825	0.0369464308023

精度比較の結果を**表 1** に示す．非定常の要因を加味することにより，いずれの群においても精度の向上が見られた．特に今回利用した非定常の要因による影響が大きく見られる箇所として翌日が平日である場合の夜間帯のアクセス頻度傾向について，**図 4** に示すように特性を捉え，適切なアクセス頻度の予測が行えていることがわかる．日中のアクセス傾向について要因を加味しなかった場合よりも精度が下がっている箇所が見られる．これは特性が現れない時間帯についても要因のフラグが設定されたことが原因と考えられるが要因の作用する時間帯を絞るなどの調整により精度改善が期待できる．

4.2 アクセス頻度予測を用いた計画的オートスケーリングの評価と考察

本節では，アクセス頻度予測を用いた計画的オートスケーリングによる処理性能と仮想サーバ台数の最適化について考察する．本研究では，オートスケーリングによる余剰な仮想サーバの起動が抑えられたことを確認するため，提案手法の適用前後での時間ごとの仮想サーバ台数を比較した．

比較結果を**図 5** に示す．A 群では 1 日あたりのサーバ起動時間が 504 時間から 159 時間に，B 群では 2,280 時間から 970 時間に減少しており，余剰な仮想サーバの起動が抑えられたことを確認できた．なお，サーバ台数下限値 L を下回った A 群の 1 時から 6 時の期間においては台数の変動が見られない．

次に提案手法により調整されたサーバ台数が適切であるか確認した．提案手法では目安となるスループット値を一定に保つために必要となるサーバ台数を求めているため，提案手法の適用前後でアクセス頻度の分散傾向を比較した．

始めに検証期間中のアクセス頻度の標準偏差を求めることで分散傾向を比較した．比較結果を**図 6** に示す．A 群では約 65.82 から約 100.56 へ，B 群では約 152.88 から約 55.21 へ変化している．

A 群の標準偏差が提案手法適用後に増加しているのは，4.1 で述べた下限値による最低台数運用となった時間帯にアクセス頻度に対してサーバ台数が多く準備されていたことによるスループット値 T の低下が原因と考えられる．下

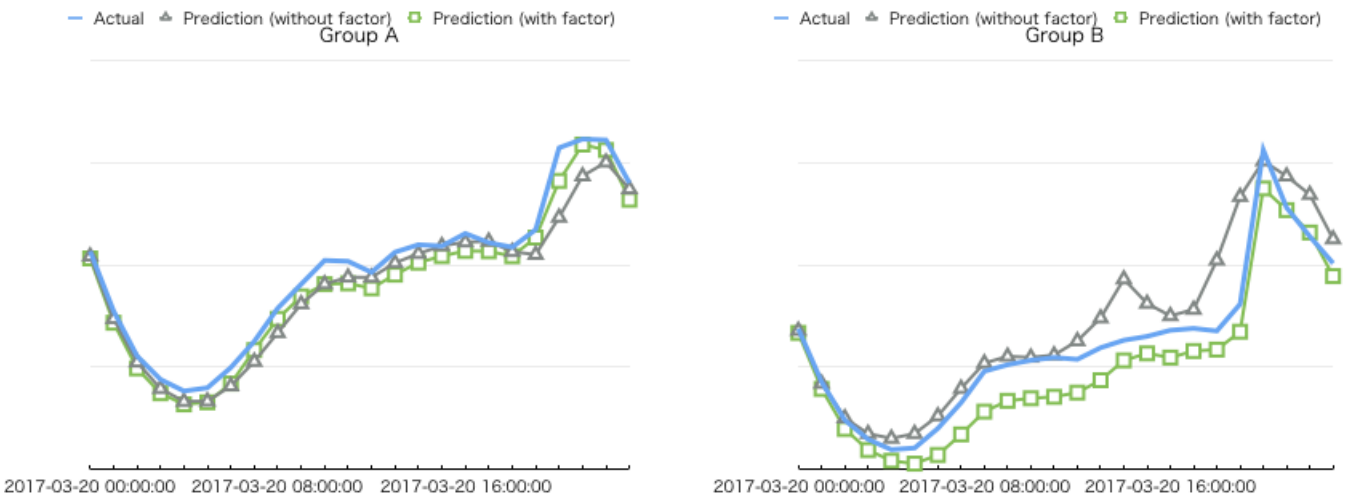

図 4 非定常の要因によるアクセス頻度予測比較

Fig. 4 Comparison of prediction access frequency by unsteady factors.

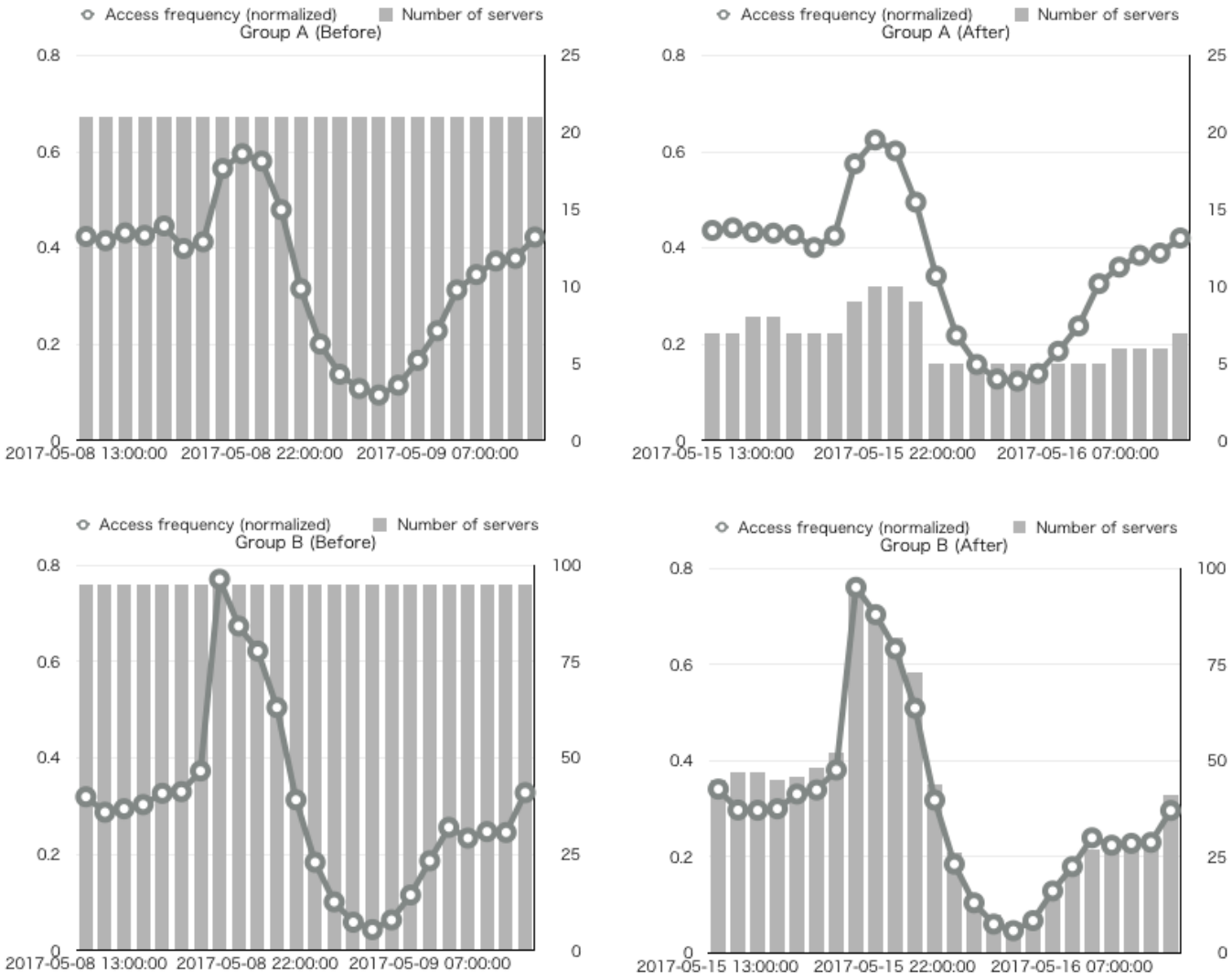

図 5 仮想サーバ台数の推移

Fig. 5 Changes in the number of virtual servers.

限値が適用されなかった B 群では検証期間を通してサーバ 1 台あたりのアクセス頻度が一定に保たれており，最適な仮想サーバ台数で安定した処理性能を保てたことが確認できた．

スループット値 T のゆらぎが見られているが，これは時間ごとの処理種別の比率の偏りが原因と考えられる．アク

セス頻度予測と目安となるスループット値を HTTP リクエストメソッドごとなどの処理種別ごとに分類することで精度向上が可能であると考える．

5. まとめ

本報告では，処理能力を保ちつつ必要最小限数の仮想サーバで運用し利用料金を抑制することが求められる Web サービス運用において，過去のアクセス頻度と非定常の要因から仮想サーバ需要を求める計画的オートスケーリングの手法を提案した．

そして提案手法の有効性を示すために，非定常の要因の有無によるアクセス頻度予測の精度を比較した．加えて Web サービスの異なるアクセス傾向を持つサーバ群に対して提案手法を適用し，適用前後で仮想サーバ台数とアクセス頻度の安定性とを比較した．

結果として，非定常の要因を考慮することで，該当要因に起因するアクセス頻度の変動箇所において変動の特性を捉えることが確認できた．また，提案手法により目安となるスループット値を一定に保つことができるようになり，余剰サーバを削減しながら，安定かつ効率的なサービス運用が行えることも確認できた．

本報告では仮想サーバの起動，停止にまつわる時間差について事前の計画的オートスケーリングにより解決を図ったが，実運用において仮想サーバの起動，停止時間は一斉起動の負荷に起因することから一定ではないこと，Web サービスの最新アプリケーションのデプロイ時間も加わることから必要台数の起動が間に合わないことがあった．よって今後の課題としては仮想サーバ準備までの時間を短縮し，リードタイムによる余剰稼働時間や性能劣化による機会損失を回避する仕組みを検討する必要がある．また，提案手法では，Web サービス運用者の把握していない突発的なアクセス頻度の変動に対しては，リソース変動を契機とするオートスケーリングを組み合わせることで，解決を図ることができる [2] ため，本研究で論じたサーバ起動完了までの時間差を解消できる手法を検討したい．

参考文献

[1] Armbrust, Michael and Fox, Armando and Griffith, Rean and Joseph, Anthony D. and Katz, Randy H. and Konwinski, Andrew and Lee, Gunho and Patterson, David A. and Rabkin, Ariel and Stoica, Ion and Zaharia, Matei, Above the clouds: A Berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley, Feb 2009.

[2] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, Dynamic provisioning of multi-tier internet applications. In 2nd International Conference on Autonomic Computing, Seattle, WA, USA, June 2005.

[3] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva, Learning Deep Features for Scene Recognition using Places Database, In Proc. NIPS, 2014.

[4] Cort J. Willmott, Kenji Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Clim. Res., 30, 7982, 2005.

[5] D. Bellenger, J. Bertram, A. Budina, A. Koschel, B. Pfnder, C. Serowy, I. Astrova, S. G. Grivas, M. Schaaf, Scaling in Cloud Environments, Proceedings of the 15th WSEAS international conference on Computers, Corfu, pp. 145-150 ,2011

[6] H. Lim, S. Babu, J. Chase, and S. Parekh, Automated Control in Cloud Computing: Challenges and Opportunities, In 1st Workshop on Automated Control for Datacenters and Clouds, p.13-18, June 2009.

[7] He S, Guo L, Guo Y, Wu C, Ghanem M, Han R, Elastic application container: A lightweight approach for cloud resource provisioning, Advanced information networking and applications (AINA 2012) IEEE 26th international conference, pp. 15-22, March 2012.

[8] Hochreiter, S., Schmidhuber, J., Long Short-Term Memory, Neural Computation 9 (8), pp. 17351780, 1997

[9] J. Rolia, X. Zhu, M. Arlitt, and A. Andrzejak. Statistical Service Assurances for Applications in Utility Grid Environments. Technical Report HPL-2002-155, HP Labs, 2002.

[10] Johannes Thnes, Microservices, IEEE Software, Volume: 32, Issue: 1, pp. 116-116, Jan.-Feb. 2015

[11] Keras. https://keras.io/

[12] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A view of cloud computing. Commun. ACM 53, 4 (April 2010), 50-58.

[13] M. Mao, J. Li and M. Humphrey. 2010. Cloud Auto-Scaling with Deadline and Budget Constraints. In Proceedings of 11th ACM/IEEE International Conference on Grid Computing. Brussels, Belgium, Oct 25-28, 2010.

[14] M. Mao, M. Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC ’ 11, page 1, NewYork, New York, USA, November 2011.

[15] RightScale. http://rightscale.com

[16] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2015. Software available from tensorflow.org

[17] Xue Liu, Jin Heo, Lui Sha, Modeling 3-Tiered Web Applications, Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, p.307-310, September 27-29, 2005

[18] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez and M. Humphrey, Early Observations on the Performance of

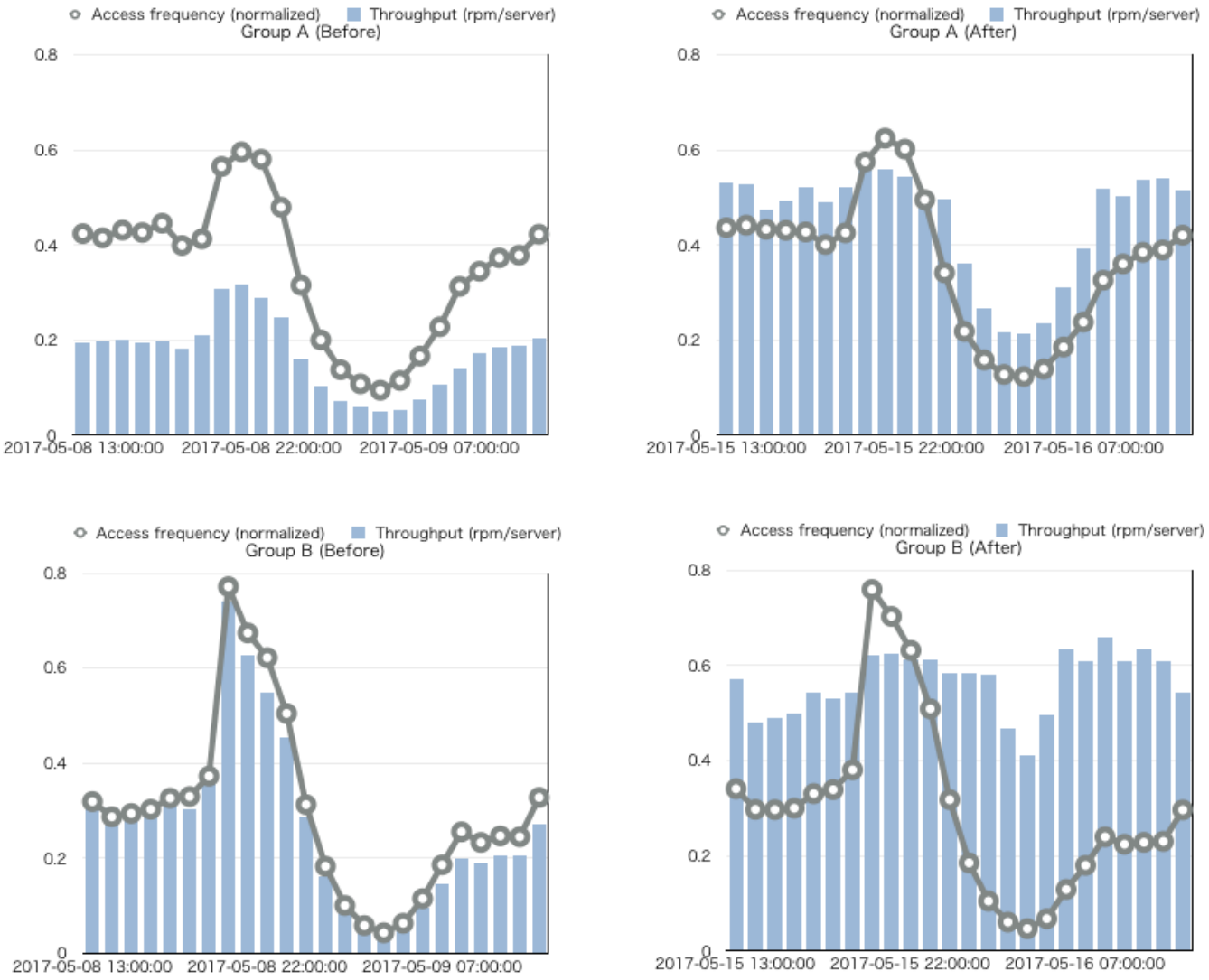

図 6 アクセス頻度の推移

Fig. 6 Changes in the access frequency.

Windows Azure, In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, Chicago, Illinois, June 21, 2010.

[19] 上田 崇介, UEDA Takayuki, 休日の特性を考慮した JPIX ネットワークトラヒックの高精度週間予測法に関する研究, 法政大学大学院紀要（理工学・工学研究科編）, 2015

[20] 総務省, 平成 28 年版　情報通信白書 第 2 節（3）クラウドサービスの利用動向, 2016

[21] 三宅 悠介, 松本 亮介, 力武 健次, 栗林 健太郎, 特徴抽出器の学習と購買履歴を必要としない類似画像による関連商品検索システム, 研究報告インターネットと運用技術（IOT）,2017-IOT-37(4),1-8 (2017-05-18) , 2188-8787