4A-2

AWSのインスタンスタイプ選択における指標の検討

永田健人† 小板隆浩 †
† 同志社大学 理工学部 情報システムデザイン学科

1. はじめに

近年，計算資源のクラウド化が進んでいる．特にAmazon Web Services(AWS)[1]は，数十万社を超えるユーザがおり，仮想サーバやストレージ，データベースを用意したプラットフォームを提供している．AWSの中でも仮想サーバ，計算資源を提供しているサービスがElastic Compute Cloud(EC2)である．EC2では，初期費用なく時間単位での支払いが可能であるため必要な分だけ稼働させることで費用を抑えることができる．時間単位での費用は，CPUやメモリなどのあらかじめ決められたスペックを示すインスタンスタイプによって決まる．また，仮想サーバは5分程度で起動できるため，急なスケールアウトにも優れている．

このようなメリットによりスタートアップ企業がウェブアプリケーションのサーバとしてEC2を選択している．しかし，どのインスタンスタイプを選ぶべきかは経験的にしか知ることができないため，期待する応答時間に対して過剰な費用が発生する，もしくは期待する応答時間を満たさないインスタンスタイプを選ぶ可能性がある．そのため，EC2の導入に際して時間もしくは費用がかかってしまうという問題が発生する．

本研究は，EC2のインスタンス選択を容易にし，導入にかかる費用や時間を少なくすることを目的とする．具体的には，インスタンスタイプごとにウェブアプリケーションの負荷テストを行い，平均応答時間とドルあたりのリクエスト処理数を評価する．平均応答時間より求める応答時間を満たすかどうかを比較し，ドルあたりのリクエスト処理数より処理に対しての費用を比較する．

A Design of Type Index for AWS Instances
Kento Nagata†, Takahiro Koita†
†Graduate School of Informatics System Design, Doshisha University

2. 負荷テスト方法

本研究で負荷対象となるアプリケーションは，WordPress[2]とRuby on Rails[3]である．WordPressは，Contents Management System(CMS)として数多くのサイトが採用しているアプリケーションである．またRuby on Railsは，Rubyで書かれているウェブアプリケーションフレームワークであり，アジャイル開発において開発コストが低いこともありスタートアップ企業において採用が急増している．これらの背景よりこの2つのアプリケーションを負荷対象として採用した．

負荷テストために使用したツールは，Apach JMeter[4]であり，ポアソン到着モデルやRailsにおけるCSRF対策の回避として要する正規表現抽出を含むテストプランを容易に作成可能である．

まず，本研究で評価するインスタンスタイプは，t1.micro，m1.small，m1.medium，m1.largeの4つである．ある同時利用ユーザ数での試行時間は1時間とし負荷テストを行う．さらに各ユーザは，平均4000ミリ秒でHTTPリクエストを送信する．ある同時利用ユーザ数において平均応答時間が8000ミリ秒を超えるものに関しては，ユーザがエラーだと見なす時間として，試行する同時利用ユーザ数の限度を決定した．

次に，各アプリケーションでの負荷テストプランを述べる．WordPressにおいてはテンプレート記事に対してリクエストを送る．すべてのユーザがある記事を表示しようとしている想定である．次にRuby on Railsにおいては，Facebookのように投稿やいいね，コメントなどが可能なパスワード認証によるユーザ登録型のアプリケーションを開発した．これは，Ruby on Railsで開発される会員制のサービスの典型例である．ユーザはログインをした後，投稿やいいね，コメント，フォローそしてストリーム表示のいずれか

Copyright ©2014 Information Processing Society of Japan.
All Rights Reserved.

をランダムに平均4秒に一度リクエストを行う．これらの負荷テストの結果を次章に述べる．

3. 負荷テスト結果

WordPressとRailsアプリケーションのドルあたりのリクエスト処理数を図1と図2に示す．どちらの結果も４つすべてのインスタンスタイプで同時利用ユーザの増加とともないドルあたりのリクエスト処理数は増加傾向にある．WordPressにおいて，同時利用ユーザ数別ドルあたりのリクエスト処理数が最大となるインスタンスタイプは，同時利用ユーザ数が1から10までがt1.micro，11から25がm1.small，26から47がm1.medium，48から128までがm1.largeとなった．次にRailsアプリケーションにおいて，同時利用ユーザ数別ドルあたりのリクエスト処理数は，同時利用ユーザ数が1から86ではt1.micro，m1.small，m1.medium，m1.largeの順に高い傾向にある．

まず，開発者は想定している同時利用ユーザ数の応答時間を比較し，求めるインスタンスタイプを検討可能である．さらに，想定している同時利用ユーザ数でのドルあたりのリクエスト処理数が最大となるインスタンスタイプを選択することにより，費用を低くすることが可能である．つまり，ドルあたりのリクエスト処理数が最大となるインスタンスタイプが提案可能なインスタンスタイプである．

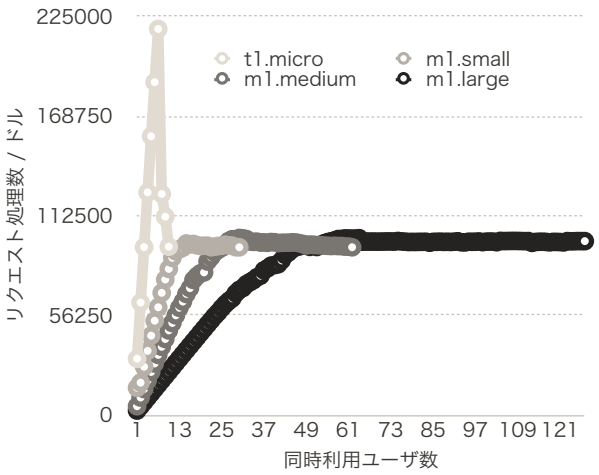

図1 WordPressにおける同時利用ユーザ数別ドルあたりのリクエスト処理数

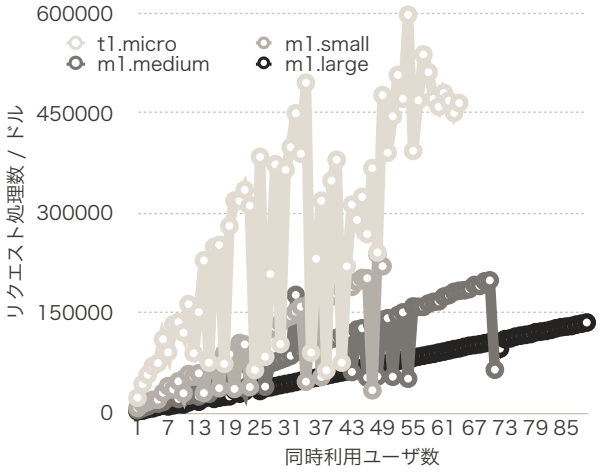

図2 Railsアプリケーションにおける同時利用ユーザ数別ドルあたりのリクエスト処理数

4. まとめ

負荷テストの結果より，開発者が想定している同時利用ユーザ数や応答時間などよりインスタンスタイプが提案可能となることが示した．インスタンスタイプを簡単に選ぶことが可能になれば，サービス開発から公開までの時間が短くなり，開発者のアイディアをサービスとしてユーザに届けやすくなる．今後の課題としては，今回実験には用いなかったインスタンスタイプでの評価もしていきたい．さらに，拡大していくサービスに対してどのようにサーバを増やしスケールアウトしていくのかという方法についても検討していきたい．

参考文献

[1] Amazon Web Services, http://aws.amazon.com
[2] WordPress, http://wordpress.org
[3] Ruby on Rails, http://rubyonrails.org
[4] Apache JMeter, http://jmeter.apache.org

Copyright ©2014 Information Processing Society of Japan.
All Rights Reserved.