AWSAutoScaling を用いたアプリケーションスケーリングの実装と評価

1200285　池上 昌志　【 分散処理 OS 研究室 】

1　はじめに

サーバ負荷に応じて自動的にクラウドサーバのリソースを増減させる仕組みとしてオートスケールがある. オートスケールを用いると必要なサーバリソース量を自動で選択し，最適なサーバリソースでサーバを運用することができる. 先行研究では CPU 使用率を閾値としたスケーリングポリシーでは爆発的なアクセスに対応できないという結果であった. 本研究では CPU 使用率の変化率を閾値としたスケーリングポリシーを AWS-SDK で実装し, 評価する.

2　スケーリングポリシー

本研究では, 二種類の CPU 使用率の変化率を算出してスケーリングしている. 一つ目は過去四分間の CPU 使用率から最大値, 最小値を選び出して算出した変化率を使用する. 二つ目は線形回帰を用いて算出した一分後の CPU 使用率を利用して出した変化率である. これらの変化率と閾値を比較し, 超えていた場合は変化率と閾値を除算した個数分のスケーリングを行う. このとき用いる閾値は, 変化率を算出する際に用いる変化前の CPU 使用率に応じて閾値を変える. これは例えば CPU 使用率が非常に高い状態で安定してしまった場合, 変化率が小さくなりスケールアウトが働かなくなってしまう. これを避けるために閾値を小さくすると, 少しのアクセスでオートスケーリンググループ最大までスケールアウトするため, スケールアウトする直前の CPU 使用率が高いほど閾値を下げる. スケールインの場合は, スケールアウトの逆の状況が起きるため, スケールインする直前の CPU 使用率が低いほど閾値を下げる.

3　評価

3.1　実験方法

仮想マシンが一つ起動しているオートスケーリンググループに Apache Bench を用いて負荷をかけ, 効率的にスケーリングを行えているか評価する. その際, 先行研究のスケーリングに用いられていたを閾値を境界として, 変化率の閾値を変更する. 表 1 に本実験で設定した閾値を示す. これらの閾値の設定理由として, 例えばスケールアウトの場合, 直前の CPU 使用率が 10%以下のときは, スケールアウトを防ぐために閾値を高く設定した. しかし, 急激な CPU 使用率の上昇が起きた場合はスケールアウトを行う. 一方, 直前の CPU 使用率が 30%以上の時は直ちにスケールアウトしなければならない状況なため閾値を低く設定した. スケールインの場合もこれと逆の状況が起こると考えたため表 1 の通り閾値を設定した. 評価の方法として, 5 分毎の仮想マシン数を先行研究と比較する. 負荷モデルとして，先行研究 [1] で用いられていたアクセスデータ（図 1）を用いる.

表 1　スケーリングの条件

CPU 使用率	スケールアウト	スケールイン
10%以下	上昇率が 200%以上	下降率が 10%以上
11~29%	上昇率が 60%以上	下降率が 30%以上
30%以上	上昇率が 20%以上	下降率が 200%以上

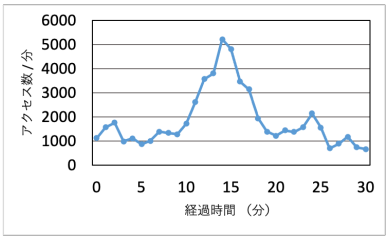

図 1　アクセスデータ

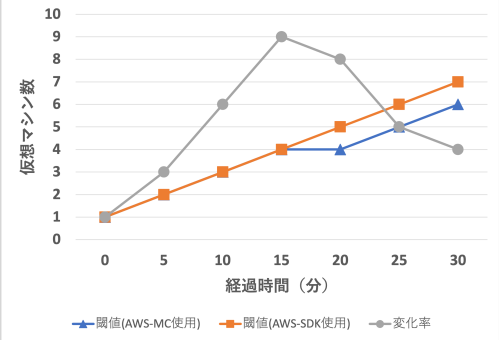

図 2　負荷テスト時の仮想マシン数の遷移

3.2　実験結果

実験結果を図 2 に示す. 結果としてアクセス数の増減にかかわらず単調増加を続けていた先行研究に比べ, 実装したスケーリングポリシーではアクセス数のピークを迎えた後, ホスト数を減少させられている. 以上のことより, 240 行程度のプログラムで記述された提案手法の方が先行研究のものより効率的にスケーリングしていることが確認できた. そして, より高度なスケーリングを行うためには機械学習による CPU 使用率の予測のなどが考えられるが, AWS で実装する場合, 他サービスとの連動や学習データの操作などによりさらに複雑な実装が必要になる.

4　まとめ

本研究では，AWS-SDK を用いて線形回帰による CPU 使用率の予測と変化率を用いたスケーリングポリシーを実装し, 実際の負荷モデルを元に評価した.

参考文献

[1] 土屋 椋, ”パブリッククラウド環境での AutoScaling 機能の評価”，令和元年度高知工科大学学士学位論文 (2020).