[ポスター発表] 研究報告

Kaburaya AutoScaler: 多環境での運用性を考慮した自律適応型オートスケーリング制御系

三宅 悠介[1,a)]　栗林 健太郎[1,b)]

Kaburaya AutoScaler: Autonomous autoscaling control system considering operability in multiple environments

情報システムの運用において，処理性能を保ちつつ必要最小限数のサーバを用いることで運用コストを制御することは，情報システムの開発運用者にとって重要な課題である．そこで，開発運用者は変動するサーバ需要に追従するためにオートスケーリング機能 [1] を情報システムへ導入する．この機能は，開発運用者が予め定めた条件を元に情報システムとこれを構成するサーバの状況を監視する．そうして，条件を満たせば，開発運用者が予め定めた条件に従いサーバ数を増減させる．このようなオートスケーリング機能は，仮想サーバを提供するクラウドサービスや，コンテナの運用基盤において標準的な機能となっている．

オートスケーリングを実行するための仕組みが整う一方で，開発運用者は情報システムの処理性能を保つ必要最小限のサーバ数を，経験と地道なチューニングを通して個別に求めている．この運用の負担は，情報システムごとに負荷に対する最適なサーバ数が異なることに起因する．そのため，開発運用者は情報システムごとにこれを構成するサーバの処理性能を計測し，要求される処理量に対して必要なサーバ数を求める．情報システムに長期間変更がなければ，開発運用者は経験的にそれらの処理性能を把握していくが，継続的に変更される場合はその限りではない．また，管理対象の情報システムの増加に従い，経験による把握は一層困難となる．そこで，継続してサーバの処理性能を把握するため，計測の自動化が行われる．パラメタ値を変化させながら負荷試験を行い最良の性能を得る値を探索的に求めることで変化する環境への追従性は向上する．しかし，負荷試験は本番環境への影響を避けて，評価用の環境で事前に実施されるため評価環境と本番環境の誤差に対処できない．そこで，フィードバック制御を用いて要求される処理量に対して必要なサーバ数を実行時に求める方式が提案されている [2]．この方式では，情報システムの変更に対する追従性が高く，評価環境との誤差も発生しない．ただし，制御量であるサーバ台数の決定は，処理性能の過不足のみから探索的に求めるため時間がかかる．処理性能が不足すると情報システムの不安定性が増大することから，サーバ台数は即時に定まることが望ましい．

また，オートスケーリング機能を安定して運用するために，開発運用者は実行時の時間差も考慮してサーバ数を求める必要がある．この時間差は「入力の遅れ」と「出力の遅れ」に分類できる．入力の遅れは，負荷上昇からサーバ台数を見積もるまでの時間差を指す．出力の遅れは，サーバ台数の変更指示から起動までの時間差を指す．これらの遅れにともない処理能力の増減も遅延するため，情報システムの不安定性が増大する．情報システムに対する負荷を予測して事前にサーバ数を変更しておく方式 [3] によって，これらの影響を回避できるが，必要なサーバ数の算出にはサーバの処理性能を把握しておく必要があり，前述と同様の課題が発生する．遅れを最小化する手法 [4], [5] でも完全に取り除けないため，依然として対策が必要である．

継続的に変更されうる複数の情報システムに対し，運用の負担を抑えながらオートスケーリング機能を安定して運用するためには，以下の要件が必要となる．

(1) 情報システムを構成するサーバの処理性能を自動で把握できる
(2) これを不必要な負荷をかけることなく実行時に継続的に把握できる
(3) 把握した処理性能から情報システムの処理性能を保つ必要最小限のサーバ数を求める
(4) オートスケーリング実行時の遅れが考慮されている

本研究では，上記の要件を満たすサーバの処理性能を実行時に自動かつ継続的に推定し，オートスケーリングの遅れも考慮した最適なサーバ数を算出する制御系を提案する．提案手法では，サーバの処理性能として単位時間あたりの

[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 810-0001 Japan
[a)] miyakey@pepabo.com
[b)] antipop@pepabo.com

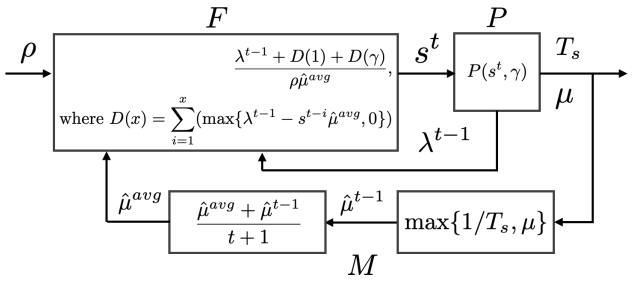

図 1　提案手法による制御系

Fig. 1　A conrol system of proposed method.

処理数の上限を用いる．ここで，処理数の上限が得られない低負荷時には単位時間あたりのレスポンスタイムの平均から処理数の上限を推定する．本研究では，要求を直列に処理するサーバを前提にレスポンスタイムの逆数からこれを導く．レスポンスタイムは高負荷時に長くなるため，この逆数である処理数の上限は低く推定される．提案手法では単位時間あたりの実際の処理数と推定処理数のうち大きいものを利用することで，情報システムに不必要な負荷をかけることなく実行時に安定してサーバの処理性能を把握できる．また，オートスケーリングの遅れを踏まえた負荷状況を予測し，これを処理可能なサーバ数を見積もることで遅れに起因する情報システムの不安定な期間を短縮する．

提案手法を用いた制御系を**図 1** に示す．F は制御器であり，推定したサーバの処理性能と情報システムの負荷状況から単位時間ごとにサーバ台数を算出する．P は情報システムである．F で求めたサーバ台数 s で運用され，平均レスポンスタイム T_s，処理数 μ を計測して F に渡す．M の経路にて，これらの計測値から推定したサーバの処理性能を全期間で平均した $\hat{\mu}^{avg}$ を得る．F では式 $(\lambda + D(1) + D(\gamma))/\rho\hat{\mu}^{avg}$ よりサーバ数を算出する．λ は実際の単位時間あたりの要求処理数，$D(1)$ と $D(\gamma)$ は入力と出力の遅れから想定される未処理の要求処理数を表す．式 $D(x)$ では各単位期間でのサーバ台数と $\hat{\mu}^{avg}$ の積を情報システムの処理能力と見なし，これの λ に対する不足分を x 単位期間，総和する．なお ρ は開発運用者が指定する利用率であり，処理数の上限に対する安全係数として働く．

本研究では提案手法の有効性を評価するためシミュレーション環境で予備評価を行なった．対象の情報システムは一般的な待ち行列モデルである M/M/S モデルに従うものとした．単位時間は 1 秒とし，負荷を変化させながら 500 ステップを計測した．出力の遅れは Web アプリケーションコンテナの起動時間 [5] を参考に 6 秒とした．なお，処理性能の推定とコンテナ起動の時間は負荷によらず一定とした．測定結果を**図 2** に示す．1 段目は情報システムに対するリクエスト数の推移である．2 段目の実線はサーバ処理性能の推定結果であり．リクエスト数に左右されず安定して推定できることがわかる．なお，緑の破線は全サーバの平均から求まるためサーバ台数の増加に従い分散が減少

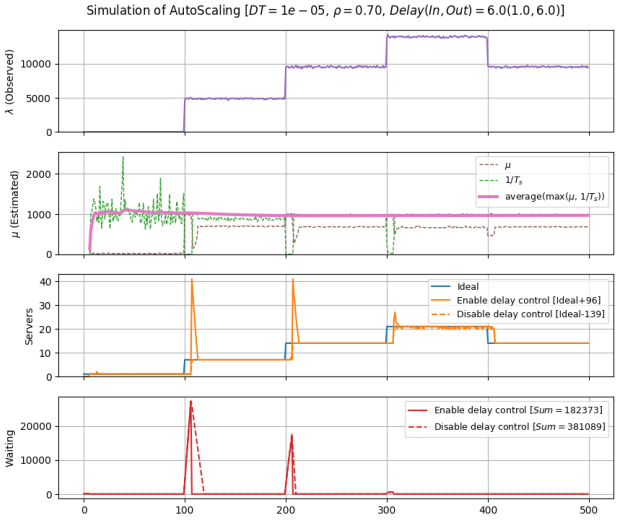

図 2　提案手法の評価シミュレーション

Fig. 2　A simulation of proposed method.

している．3 段目はサーバ数の推移，4 段目は未処理の要求数の推移である．比較のため遅れ対策をしない場合の推移も各々に破線で示した．青線の理想サーバ数に追従しながら，リクエスト数の増加時には遅れ対策によってサーバ起動後，直ちに未処理の要求が解消されたことが確認できる．

本研究では継続的に変更されうる複数の情報システムに対し，多環境での運用を考慮した自律適応型オートスケーリング制御系を提案した．シミュレーションによる予備評価では一般的な待ち行列モデルに対する提案手法の有効性を示した．実用化に向けて並列処理を前提としたサーバ適用や実環境の要求処理量，遅れ時間での評価を進めていく．また，提案手法では入力の遅れに伴い単位時間内に待ちリクエストが必ず発生するため変化点検出 [4] を組み合わせて遅れ時間自体を短縮する方式も検討したい．

参考文献

[1] Dominique Bellenger, Jens Bertram, Andy Budina, Arne Koschel, Benjamin Pfänder, Carsten Serowy, Irina Astrova, Stella Gatziu Grivas, and Marc Schaaf. Scaling in cloud environments. *Recent Researches in Computer Science*, Vol. 33, pp. 145–150, 2011.

[2] 三宅悠介. Ebira: アクセス負荷に応じて継続的にスケーリング基準を最適化する汎用オートスケーリング機構. 入手先〈`https://blog.monochromegane.com/blog/2019/04/14/wsa_4_ebira/`〉(参照 2019-10-17).

[3] 三宅悠介, 松本亮介, 力武健次, 栗林健太郎. 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. 情報科学技術フォーラム講演論文集, Vol. 17, No. 4, pp. 7–12, Sep 2018.

[4] Jun-ichi Takeuchi and Kenji Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*, Vol. 18, No. 4, pp. 482–492, 2006.

[5] 松本亮介, 近藤宇智朗. Criu を利用した http リクエスト単位でコンテナを再配置できる低コストで高速なスケジューリング手法. 研究報告インターネットと運用技術 (IOT), Vol. Vol.2019-IOT-44, pp. 1–8, Feb 2019.