ライブマイグレーションの影響を考慮して機器集約を実現するLinux コンテナ配置手法の提案

Linux Container Placement Method in Consideration of Live Migration Effect for Server Consolidation

沖津健吾 [1]　村山隆彦 [1*]
Kengo Okitsu[1]　Takahiko Murayama[1]

[1] NTT ソフトウェアイノベーションセンタ
[1] NTT Software Innovation Center

Abstract: 近年，Web サービスを提供するクラウド環境として，従来の仮想マシンベースではない，Linux コンテナベースの環境が注目されつつあり，Linux コンテナを顧客に提供するクラウド事業者も増えている．そのようなクラウド事業者にとって，少ない物理サーバ数でより多くの顧客 Linux コンテナを配置すること，すなわち機器集約率の向上は重要な課題の１つである．本論文では，適切な配置変更を行うために把握が必要な，ライブマイグレーションの影響を定量的に予測する手法を提案する．さらに，ライブマイグレーションの影響予測に基いた適切な配置決定に，強化学習を適用した手法を提案する．

1　はじめに

クラウド事業者の提供形態のうち，物理サーバを仮想化したクラウド環境を構築，運用し，仮想化された単位である仮想インスタンスを顧客に提供する形態は IaaS (Infrastructure as a Service) と呼ばれている．IaaS は従来，仮想インスタンスとして仮想マシンを提供することがほとんどであった．しかし近年，Docker[1] をはじめとした Web サービスのリリースプロセスを効率化できる Linux コンテナ管理技術の登場により，Linux コンテナを提供する IaaS が増加している [2, 3].

IaaS の効率運用において，仮想インスタンスをその計算資源要求に応じつつ必要な物理サーバが少なくなるように配置することは重要な課題の 1 つである．本研究では，特定の利用主体が計画的に IaaS を利用する場合を想定し，既に物理サーバに配置されている Linux コンテナを機器集約率が向上するよう配置する再配置問題に取り組む.

再配置問題については，これまで仮想マシンを前提に数多くの研究がなされてきた [4]．しかし，Linux コンテナは仮想マシンとは仮想化の仕組みが異なるため，再配置操作であるライブマイグレーションに係る所要時間と生じる計算資源使用量が小さくなると期待されている．そのため，これらのライブマイグレーション影響の違いを定量的に考慮せず仮想マシンと同様の配置方式を採用すると，期待通りの機器集約率が得られない場合が生じる．例えば再配置の影響を過大に見積もってしまうことで，本来は再配置した方が機器集約率がする場合でも再配置しないと判断してしまう可能性が高い.

本研究は，ライブマイグレーション影響を定量的に考慮した Linux コンテナ再配置方式を提案することを目的とする.

2　課題とアプローチ

2.1　ライブマイグレーション影響の予測

ライブマイグレーションの基本的な処理は仮想インスタンスに関わらず共通している．まず，仮想インスタンスが使用している送信元物理サーバのメモリ内容を，宛先物理サーバに一定期間毎に反復してコピーする．最後に送信元物理サーバ上の仮想インスタンスを停止させ，宛先物理サーバ上で起動する．最後の停止と起動の処理は，停止時間が予め定められた閾値を下回ることが判明した場合，または予め定められたコピー回数を上回った場合に実行される [5].

* 連絡先：日本電信電話株式会社
〒180-8585 東京都武蔵野市緑町 3-9-11
E-mail: murayama.takahiko@lab.ntt.co.jp

処理内容から，本研究で対象とする再配置位置決定に関わるライブマイグレーション影響は次の通り：

- 送信元／宛先物理サーバの計算資源の使用
  - CPU
  - メモリ
  - ネットワーク帯域
- ライブマイグレーション所要時間
- Linux コンテナ停止時間

Linux コンテナ停止時間が含まれているのは，停止している時間だけ Linux コンテナからの計算資源要求に応えられていないと考えられるためである．

Linux コンテナのライブマイグレーション影響について調査した研究は現時点では存在しないため，計測をしてその結果を元にモデル化する必要がある．従ってライブマイグレーション影響の予測における課題は次の通り：

(課題 1) 再配置位置決定に関わるライブマイグレーション影響を予測すること

Linux コンテナに関する研究が存在しない一方で，仮想マシンを対象として影響を予測する研究は取り組まれてきた．Verma らは，ライブマイグレーション対象の仮想マシンの計算資源使用状況などを操作して，ライブマイグレーション所要時間に与える影響を詳細に調べている [6]．その結果，所要時間は仮想マシンのメモリ使用量に大きく依存することが分かっている．ライブマイグレーション処理の共通性から，この依存関係は Linux コンテナの場合にも存在することが予想される．

よって本研究では，Linux コンテナのメモリ使用量を操作し，その結果ライブマイグレーション影響がどのように変化するかを計測してモデル化する．なお，今回はライブマイグレーション専用の大きなネットワーク資源があることを前提として，ネットワーク帯域についてはスコープ外とする．

2.2 再配置パターンの決定

再配置パターンの決定について，仮想マシンを対象として数多くの研究がなされてきた．しかし，ライブマイグレーション影響の予測の場合とは異なり，仮想インスタンス種別を問わず適用可能な手法が提案されているため，仮想マシンの研究から課題を抽出する．

機器集約の観点で適切な再配置パターンの決定を行うためには，ライブマイグレーション影響に加えて，仮想インスタンスの計算資源使用の変動予測が必要であることが知られている [6]．これは，同時に計算資源要求が上昇する仮想インスタンスを同じ物理サーバ上に配置してしまい，要求された計算資源を提供できない過負荷状態が発生するのを避けるためである．

これまで，予測した仮想インスタンスの計算資源要求の変動に応じて再配置する方式 [8]，ライブマイグレーションの所要時間を考慮した配置方式 [7] が提案されてきた．しかし，仮想インスタンスの計算資源要求の変動予測とライブマイグレーション影響の双方を考慮した配置方式は存在しない．そのため，ライブマイグレーション影響と仮想インスタンスの計算資源要求の変動を同時に考慮しなければ機器集約率を向上できない場合には対応できない．従って再配置パターンの決定おける課題は次の通り：

(課題 2) 仮想インスタンスの計算資源要求の変動とライブマイグレーション影響の双方を考慮した再配置方式を確立すること

本研究ではこの (課題 2) に対して、IaaS 環境で一般的に観測される，仮想インスタンスからの計算資源要求の変動幅が大きい状況において長期間に渡って機器集約率を向上できる方式確立を目指す．

再配置問題の定式化には，大きく 2 つのアプローチが考えられる．1 つは，組合せ最適化問題であるビンパッキング問題として定式化するアプローチである．この定式化はある限られた期間における最適配置を決定する問題設定である．そのため，計算資源要求の変動幅が小さい場合は，再配置による効果が持続するため，高い機器集約効果を実現できる．また，高速に解を求めるヒューリスティクス解法がよく研究されており，実装が容易という利点がある．一方で，計算資源要求の変動幅が大きい場合は，配置先を決めた以外の期間は考慮にいれないため，長期的には機器集約率を向上できない．

もう 1 つのアプローチは，長期に渡る目的関数を最大化する制御系列を決定する制御問題として定式化するアプローチである．そのような手法で代表的なものとして，一連の制御の結果将来に渡って得られる報酬の期待値を最大化する枠組みである，強化学習がある．この定式化は目的関数が長期に渡って計算されるものであるため，計算資源要求の変動幅が大きい場合でも，長期的に見て機器集約率が向上する制御則を得ることができる．一方で，すぐにそのような制御則を得られるわけではないため，制御則の学習効率を向上させる工夫が必要となる．

本研究では，想定している状況に適している強化学習を用いて再配置問題を定式化する．

3 提案方式

3.1 ライブマイグレーション影響の予測

プロセスのメモリの使用量を約 30MB から 600MB まで操作し，ライブマイグレーション影響がどのように変化するかを計測し，モデル化を行った．今回ライブマイグレーション対象をプロセスとしたのは，使用したライブマイグレーション OSS p.haul[10] の実装状況に因る．計測対象をプロセスとしても，ライブマイグレーション処理の内容から，メモリ使用量に対するライブマイグレーション影響の変化に関する結果は，Linux コンテナとほぼ同等と考えることができる．しかし，今回の計測結果の定量的な値にはあまり意味がないため，ここでは得られた結果の依存関係のみを定式化する．

計測結果から，ライブマイグレーション所要時間，Linux コンテナ停止時間，及び宛先サーバの CPU ／メモリ使用量の平均値は，プロセスのメモリ使用量に対して線形に増加することが分かった．また，送信元サーバの CPU ／メモリ使用量のの平均値は，プロセスのメモリ使用量に対してほとんど変化しないことが分かった．

従って，時刻 t の i 番目の Linux コンテナのメモリ使用量を $m_i(t)$ と書くと，i 番目の Linux コンテナのライブマイグレーション所要時間 mgt_i，Linux コンテナ停止時間 ft_i，ライブマイグレーション宛先サーバの CPU ／メモリ使用量のの平均値 mgc_i^d, mgm_i^d，ライブマイグレーション送信元サーバの CPU ／メモリ使用量のの平均値 mgc_i^s, mgm_i^m は以下のように予測が可能である．

$$
\begin{aligned}
mgt_i &= \alpha_{mgt} m_i(t) \\
ft_i &= \alpha_{ft} m_i(t) \\
mgc_i^d &= \alpha_{mgc}^d m_i(t) \\
mgm_i^d &= \alpha_{mgm}^d m_i(t) \\
mgc_i^s &= \alpha_{mgc}^s \\
mgm_i^s &= \alpha_{mgm}^s
\end{aligned}
$$

ただし, $\alpha_{mgt}, \alpha_{ft}, \alpha_{mgc}^d, \alpha_{mgm}^d, \alpha_{mgc}^s, \alpha_{mgc}^s, \alpha_{mgm}^s$ は全て正の定数.

3.2 再配置方式

3.2.1 強化学習

強化学習の概要について説明する．強化学習では，学習する主体であるエージェントが，離散的な時刻 t における状態 $s_t = s \in S$ を観測した後，行動則に従って行動 $a_t = a \in A$ を選択する．ここで，S は状態集合，A は行動集合である．この行動則は方策と呼ばれており，方策は確率 $p(a|s)$ の集合 π で表される．そして次の離散時刻 $t+1$ において，エージェントは報酬 r_{t+1} を得た後，状態 $s_{t+1} = s'$ を観測する．この一連の流れの背後には，状態遷移モデル $\mathcal{P}_{ss'}^a$, 期待報酬モデル $\mathcal{R}_{ss'}^a$ という 2 つの環境モデルが存在し，状態遷移と報酬獲得はこれらの環境モデルに従って生じる．

$$
\begin{aligned}
\mathcal{P}_{ss'}^a &= P(s_{t+1} = s' | s_t = s, a_t = a) \\
\mathcal{R}_{ss'}^a &= E[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']
\end{aligned}
$$

この環境モデルの中で，エージェントは収益と呼ばれる以下の値を最大化する最適方策を得ることを目的に学習を行う [9].

$$
R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} (0 \leq \gamma \leq 1)
$$

最適方策を得る学習アルゴリズムは数多く提案されており，環境モデルを陽に必要とするものはモデルベース学習，必要としないものはモデルフリー学習と呼ばれている．両者の学習の間には学習効率と計算時間のトレードオフが存在し，一般にモデルフリー学習の方が最適方策獲得までに多くの経験を要するが，計算時間は小さいと言われている．

提案方式では，既に構築したライブマイグレーション影響の予測モデルを活かして学習効率を向上できる，モデルベース学習を用いる．その中でも比較的計算時間を小さくすることを志向し，一般によく用いられるモデルフリー学習である Q-learning の学習効率を環境モデルの利用により向上したアルゴリズムを，再配置問題に適用することを試みる．

ここで，提案方式のベースとした Q-learning アルゴリズムの概要を説明する．Q-learning とは，次に表される状態 s に対して行動 a を選択する価値，行動価値関数 $Q(s,a)$ の学習を行うアルゴリズムのことである．

$$
Q(s,a) = E[R_t | s_t = s, a_t = a]
$$

次の時刻の状態を観測する度に，つまり標本 $(s_t, a_t, r_{t+1}, s_{t+1})$ を得る度に，$0 \leq \alpha \leq 1$ として次の更新則で $Q(s,a)$ を更新することで，最適な行動価値関数を直接学習できる．

$$
\begin{aligned}
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [& r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \\
& - Q(s_t, a_t)]
\end{aligned}
$$

方策は $Q(s,a)$ を元に, 例えば $a_t = \arg\max_a Q(s_t, a)$ のように設定する．

環境モデルを利用して Q-learning の学習効率を向上するために，提案方式では 1 ステップ先までの予測を

表 1　基本的な変数

変数	説明
n_{co}	Linux コンテナの個数
n_s	物理サーバの台数
$\boldsymbol{MX}_c$	物理サーバの最大 CPU 使用量 $(1 \times n_s)$
$\boldsymbol{MX}_m$	物理サーバの最大メモリ使用量 $(1 \times n_s)$
$\boldsymbol{C}(t)$	時刻 t のコンテナの CPU 使用量 $(1 \times n_{co})$
$\boldsymbol{M}(t)$	時刻 t のコンテナのメモリ使用量 $(1 \times n_{co})$
$\boldsymbol{P}(t)$	時刻 t のコンテナの配置物理サーバ $(1 \times n_{co})$

用いて Q を更新する．つまり，状態 s_t を観測した後，行動 a を選択した場合の次の報酬 $\hat{r}_{t+1}$ と状態 $\hat{s}_{t+1}$ を推定して，推定した値を含む標本 $(s_t, a, \hat{r}_{t+1}, \hat{s}_{t+1})$ を元に $Q(s_t, a)$ を更新するものとする．これにより，通常の Q-learning よりも早く最適な行動価値関数を学習できる．

3.2.2　時間ステップ，状態，行動

提案方式の基本的な変数を表 1 に示す．$\boldsymbol{MX}_c, \boldsymbol{MX}_m$ は，k 番目の物理サーバを S_k とすると，$(1, k)$ 要素を $MX_c(S_k), MX_m(S_k)$ と表す．また，$\boldsymbol{C}(t), \boldsymbol{M}(t), \boldsymbol{P}(t)$ は，i 番目の要素をそれぞれ $c_i(t), m_i(t), p_i(t)$ と表す．ここで，CPU 使用量は今回は使用率と同等と考え，$\boldsymbol{MX}_c$ の要素は全て 100，$c_i(t)$ は 0 から 100 までの整数値を取るものとする．$\boldsymbol{MX}_m$ の要素と $m_i(t)$ の単位はメガバイトとする．$p_i(t)$ は，i 番目の Linux コンテナが k 番目の物理サーバに配置されている場合，値 k を取るものとする．

強化学習では時刻は離散的で，行動により次の時刻の状態遷移が生じる必要がある．よって行動である再配置により状態繊維することを保証するため，時間ステップは 1 ステップ毎のアルゴリズム計算時間に，最大のライブマイグレーション所要時間 $\alpha_{mgt} \max_k MX_m(S_k)$ を足した値 τ と設定する．

時刻 t における状態 s_t は, $t-\tau$ から t までの Linux コンテナ i の CPU ／メモリ使用量の平均値 $c_i^m(t), m_i^m(t)$ を $(1, i)$ 要素に持つ $1 \times n_{co}$ ベクトル $\boldsymbol{C}_m(t), \boldsymbol{M}_m(t)$ と，配置位置 $\boldsymbol{P}(t)$ の組とする．行動は次の時間ステップ $t+\tau$ における配置位置とする．配置位置の集合を $\boldsymbol{P}_a$ とすると，状態と行動は次のように表せる．

$$s_t = (\boldsymbol{C}_m(t), \boldsymbol{M}_m(t), \boldsymbol{P}(t))$$
$$a_t = \boldsymbol{P} \in \boldsymbol{P}_a$$

3.2.3　状態遷移モデルによる状態予測

状態 s_t を観測した後 1 ステップ先の状態を予測するために，状態遷移モデルを定式化する．状態には計算資源使用量と，配置位置が含まれているため，双方について状態遷移モデルを定式化する．

配置位置については，ライブマイグレーションは必ず成功するとして，確率 1 で選択した行動の通りになるものとする．

$$\boldsymbol{P}(t+\tau) = a_t$$

計算資源使用量については，予測方法と，予測の精度を向上する更新方法を定式化する必要がある．予測方法としては，時系列パターン予測の様々な手法が考えられるが，ここでは経済指標予測などで広く用いられている AR(Auto Regressive) モデルを採用する．

まず，AR モデルのパラメータである次数は，適当な自然数 n に対して，$n\tau$ 毎に Yule-Walker 法を用いて推定を行う．パラメータ推定された AR モデルを用いて，時刻 $t-\tau$ から t までの使用量から $t+1$ から $t+\tau$ の使用量を推定する．予測された CPU 使用量を $\hat{\boldsymbol{C}}(t+1), \hat{\boldsymbol{C}}(t+2), \cdots, \hat{\boldsymbol{C}}(t+\tau)$ とすると，これらを要素毎に時間方向に平均したベクトル $\hat{\boldsymbol{C}}_m(t+\tau)$ が，次の時間ステップにおける CPU 使用量の平均値である．メモリについても同様で，予測した使用量の平均値を $\hat{\boldsymbol{M}}_m(t+\tau)$ とする．

従って，状態 s_t に対して行動 $a_t = \boldsymbol{P}$ を選択したと仮定した時，予測される 1 ステップ先の状態は次の通り．

$$\hat{s}_{t+\tau} = (\hat{\boldsymbol{C}}_m(t+\tau), \hat{\boldsymbol{M}}_m(t+\tau), \boldsymbol{P})$$

3.2.4　期待報酬モデル

1 ステップ先の状態を予測した際に報酬量が計算できるよう，期待報酬モデルを定式化する．期待報酬モデルは，報酬量が大きければ大きいほど機器集約率が向上するように設計する必要がある．よって，計算資源要求満足度である，物理サーバ計算資源使用の余剰から Linux コンテナ停止ペナルティを差し引いた値から，さらに物理サーバ数増加によるペナルティを差し引いた値を期待報酬として定式化する．計算資源要求満足度とペナルティをどの程度重視するかを調整できるよう，これらは重み付きで足しあわされるようにする．

まず，物理サーバ計算資源使用の余剰を定式化する．時刻 t から $t+\tau$ の間に，物理サーバ k 上に Linux コンテナ i が配置されている場合は 1，それ以外の場合は 0 を取る変数を ce_{ki} と置く．また，時刻 t から $t+\tau$ の間

に，物理サーバ k が Linux コンテナ i のライブマイグレーション送信元である場合は 1，それ以外の場合は 0 を取る変数を mgs_{ki}，宛先に関する同様の変数を mgd_{ki} と置く．この時，時刻 t から $t+\tau$ における物理サーバ k の CPU 使用量の平均値 $SC_k^m(t+\tau)$ は，ライブマイグレーション影響を合わせると次の通り．

$$SC_m^k(t+\tau) = \sum_{i=1}^{n_{co}} \Big\{ ce_{ki}\ c_i^m(t+\tau) + \frac{\alpha_{mgt}}{\tau} \big(mgs_{ki}\ \alpha_{mgc}^s\ m_i(t) + mgd_{ki}\ \alpha_{mgc}^d\ (m_i(t))^2 \big) \Big\}$$

$\sum$ 内の第一項は，物理サーバ k 上の Linux コンテナの使用量合計，第二項は送信のための使用量，第三項は宛先のための使用量である．

$SC_k^m(t+\tau)$ と同様のメモリに関する値を $SM_k^m(t+\tau)$ とする．この時 $SC_k^m(t+\tau), SM_k^m(t+\tau)$ を $(1,k)$ 要素に持つ $1 \times n_s$ ベクトルを $\boldsymbol{SC}_m(t+\tau), \boldsymbol{SM}_m(t+\tau)$ と置くと，時間ステップ $t+\tau$ における物理サーバの計算資源余剰 $r_s(t+\tau)$ は，次の通り．

$$r_s(t+\tau) = \frac{1}{N_c}(\boldsymbol{MX}_c - \boldsymbol{SC}_m(t+\tau))\boldsymbol{I}_{n_s}^T + \frac{1}{N_m}(\boldsymbol{MX}_m - \boldsymbol{SM}_m(t+\tau))\boldsymbol{I}_{n_s}^T$$

ここで，N_c, N_m は CPU 使用量とメモリ使用量の大きさの影響を，最大値 1 に揃える正規化定数．また，$\boldsymbol{I}_{n_s}^T$ は全ての要素が 1 の $1 \times n_s$ ベクトル．

次に，Linux コンテナ停止ペナルティを定式化する．時刻 t で Linux コンテナ i をライブマイグレーションすることを決定した場合 1，それ以外の場合は 0 を取る変数を $mgf_i(t)$ と置く．この時 $mgf_i(t) = 1$ なる Linux コンテナは，時刻 $t + (\alpha_{mgt} - \alpha_{ft})m_i(t)$ から $\alpha_{ft} m_i(t)$ だけ停止する．この停止の間はそれだけ計算資源要求に応えられていないと見なせるため，停止によるペナルティ $p_s(t+\tau)$ は次の通り．

$$p_s(t+\tau) = \sum_{i=1}^{n_{co}} \frac{mgf_i(t)\ \alpha_{ft}\ m_i(t)}{\tau} \Big\{ \frac{c_i(t + (\alpha_{mgt} - \alpha_{ft})m_i(t))}{N_c} + \frac{m_i(t + (\alpha_{mgt} - \alpha_{ft})m_i(t))}{N_m} \Big\}$$

従って，計算資源要求満足 $R_s(t+\tau)$ は次の通り．

$$R_s(t+\tau) = r_s(t+\tau) - p_s(t+\tau) \tag{1}$$

続いて，物理サーバ数増加によるペナルティを定式化する．時間ステップ $t+\tau$ において，Linux コンテナが配置されている物理サーバ台数を $h(t+\tau)$ とすると，物理サーバ数増加ペナルティは次のように書ける．

$$p_h(t+\tau) = \frac{h(t+\tau)}{n_s} \tag{2}$$

式 (1), (2) を合わせると，状態と行動が決まれば報酬を次のように算出できる．

$$r_{t+\tau} = w_s R_s(t+\tau) - w_h p_h(t+\tau)$$

なお，w_s, w_h は $w_s + w_h = 1$ を満たす正の値であり，計算資源要求満足度と物理サーバ台数減少のどちらを優先するかを調整可能な定数である．

3.2.5 更新則と行動選択

環境モデルを利用して行動価値関数を学習する更新則と，行動価値関数に基づく行動選択を定式化する．状態 s_t を観測した後，行動 $\boldsymbol{P}$ を選択したと仮定して，状態遷移モデルに基いて 1 ステップ先の状態 $\hat{s}_{t+\tau}$ を推定する．そして，期待報酬モデルから推定報酬 $\hat{r}_{t+\tau}$ を算出する．ここまでで得られた標本 $(s_t, a_t, \hat{r}_{t+\tau}, \hat{s}_{t+\tau})$ を用いて，以下の更新則で行動価値関数を更新する．

$$Q(s_t, \boldsymbol{P}) \leftarrow Q(s_t, \boldsymbol{P}) + \alpha[\hat{r}_{t+\tau} + \gamma \max_{\boldsymbol{P'}} Q(\hat{s}_{t+\tau}, \boldsymbol{P'}) - Q(s_t, \boldsymbol{P})]$$

これを $\boldsymbol{P} \in \boldsymbol{P}_a$ について計算し，以下のようにグリーディに行動を選択する．

$$a_t = \arg\max_{\boldsymbol{P}} Q(s_t, \boldsymbol{P})$$

4 まとめと今後の課題

本研究は，ライブマイグレーション影響を定量的に考慮した Linux コンテナ再配置方式確立を目的として行われた．計算資源要求の変動幅が大きい場合でも長期的に見て機器集約率が向上するよう，制御手法の一種である強化学習を適用するアプローチを選択し，方式を提案した．

ライブマイグレーション影響に関する (課題 1) に対しては，計測実験から，Linux コンテナのメモリ使用量から再配置位置決定に関わるライブマイグレーション影響を予測できることが分かった．再配置パターン決定に関する (課題 2) に対しては，従来手法では対応できない，ライブマイグレーション影響，Linux コンテナの計算資源使用の変動の双方を考慮しないと機器集約率を向上できない場合にも適用できることが分かった．

しかしながら，いくつか残された課題も存在する．ライブマイグレーション予測においては，メモリ使用量に

対する依存関係までは把握できたものの，その定量的な値までは正確に予測することができない．さらに，ネットワーク帯域への影響については，依存関係についてもモデル化できていない．今後は，ネットワーク帯域まで含めた計測を行い，定量的な予測モデルを構築する必要がある．

再配置パターン決定においては，IaaS 規模が大きくなると学習速度と計算速度が実用に耐えない可能性がある．IaaS 規模が大きい場合，状態集合がベクトルの組で表されていることから，同じ状態を観測する頻度が低くなり，学習速度が低下する．また，行動集合の大きさが $n_{co}^{n_s}$ と大きいため，時間ステップ毎の計算時間が増大する．今後は，学習速度を向上する汎化や，計算する行動集合の刈り込みによる計算時間の削減などにより改善する必要がある．

今後はこれらの課題を解決した後に，従来方式との比較実験により提案手法の評価を行い，提案手法の有効性を確認したい．

参考文献

[1] Docker, Inc: Docker, https://github.com/docker/docker

[2] Amazon, Inc: Amazon EC2 Container Service, https://aws.amazon.com/jp/ecs/

[3] Google, Inc: Googler Container Engine, https://cloud.google.com/container-engine/

[4] Jennings, B., Stadler, R.: Resource Management in Clouds: Survey and Research Challenges, *Journal of Network and Systems Management*, Vol. 23, No. 3, pp. 567-619 (2015)

[5] Mirkin, A., Kuznetsov, A., Kolyshkin, K.: Containers checkpointing and live migration, *Proceedings of the Linux Symposium* , Ottawa, Canada (2008)

[6] Verma, A., Kumar, G., Koller, R.: The cost of reconfiguration in a cloud, *Proceedings of the 11th International Middleware Conference Industrial track*, pp. 11-16, Bangalore, India (2010)

[7] Sharma, U., Shenoy, P., Sahu, S., Shaikh, A.: A Cost-Aware Elasticity Provisioning System for the Cloud, *31st International Conference on Distributed Computing Systems*, pp. 559-580, Minneapolis, USA (2011)

[8] Zhani, M.F., Zhang, Q., Simon, G., Boutaba, R.: VDC planner—dynamic migrationaware virtual data center embedding for clouds, *Proceedings of 2013 IFIP/IEEE International Symposium on Integrated Network Management*, pp. 18-25, Sophia Antipolis, France (2013)

[9] Sutton, RS., Barto, AG.: Reinforcement Learning, *MIT Press*, Cambridge, MA (1998)

[10] xemul: p.haul, https://github.com/xemul/p.haul