コンテナの処理特性に応じた割当リソース量を推論するリソース量推論モデルの検討

水野 和彦† 坂田 匡通‡ 石井 陽介‡

†日立製作所研究開発グループ 〒185-8601 東京都国分寺市東恋ヶ窪 1 丁目 280

E-mail: †{kazuhiko.mizuno.pq, masayuki.sakata.nm, yohsuke.ishii.bz@hitachi.com

あらまし ハイブリッドクラウドの普及に伴い, 多拠点に散在する IT リソースの使いこなしによる利用コスト適正化が求められている．利用コスト適正化では，複数の拠点の中からアプリケーションの最適な実行場所を選定するために，アプリケーションが実行される各コンテナで必要なリソース量を推論するモデル作成が課題となる．そこで，各コンテナの稼働情報からコンテナの処理特性に応じてリソース量を推論するリソース量推論モデルを提案する．提案モデルの原理検証結果から，コンテナの処理特性を分類し，コンテナに必要な CPU コア数を推論できた．また，提案モデルを適用することで，IT リソース使いこなしのためのアプリケーションの最適な実行場所を選定できる見込みを得た．

キーワード 回帰モデル，最適配置，コンテナのリソース割当

1. はじめに

近年，ビッグデータならびに分析ソフトウェアを扱う市場は急速な拡大を続けており，2018 年に World Wide で約$60B の市場規模に達し，2019-2023 年の CAGR (Compound Annual Growth Rate) は 12.5%と予測されている[1]．この市場の伸びは，日々の業務で発生する大量のデータの蓄積と，様々な観点での分析結果の活用による新しい付加価値の創出で裏打ちされている．また，近年特に規模が拡大している IoT 市場[2]においても同様のことが言える．

また，パブリッククラウドサービスプロバイダが提供するパブリッククラウドサービスの利用も広がってきている．これに伴い，工場の製造ラインと連携した現場における装置監視ならびにデータ収集システムと，オンプレミス環境で利用していた IT(Information Technology)インフラ設備で稼働していた IT システムと，パブリッククラウドサービスを連携させたハイブリッドクラウド環境によるデータ利活用環境を構築し，利用するケースも増えてきている．パブリッククラウドサービスが提供する柔軟かつ迅速なリソース提供を活用することで，新規の各種サービス向けのシステムを過大な設備投資をすることなくスモールスタートすることも可能になっている．

このようなパブリッククラウドサービスの活用の利用が進む一方で，最近では，データを扱う上で EU の GDPR(General Data Protection Regulation)などの法規制に対応するために，対象データの保管場所や利用可能な範囲を明確にした上で，その範囲内でデータを IT システム内で利用制限する仕組みが求められてくるようになってきている．このために，対象データが保管されているサイトを意識したデータ管理サービスも求められるようになってきている．

さらに, パブリッククラウドサービスは, pay-per-use の課金モデルに基づいて，比較的短期間で多量のリソースを一時的に利用するケースにおいて費用対効果の高い環境である．しかし，中長期にわたって一定量のリソースを継続的に利用するケースにおいては割高になるケースもある．このために，現有の IT インフラのリソースとパブリッククラウドサービスのリソースを効率よく使いこなすことも求められるようになってきている．

著者らは，このような状況下において，ハイブリッドクラウド環境におけるデータストレージリソースのみならずコンピューティングリソースを賢く使いこなすサービスを提供することは，利用者に対して様々な価値を提供することができ有益と考えており，当該サービスをデジタルインフラ面で支えるサービスやソリューション実現のための検討を進めている．本検討では，データストレージ環境ならびにコンピューティング環境のインフラ情報ならびに利用アプリケーション（以下,アプリと呼ぶ)などの稼働構成情報をもとに，アプリやインフラをモデル化し，利用者のポリシーやデータ利用上の法規制などに基づいて，アプリ実行環境となるコンテナやデータの適正な配置場所を立案し，最適配置制御できるようにする．

本研究では，ハイブリッドクラウド環境におけるコンテナ・データ最適配置を実現するために，対象となるアプリやインフラに関する構成情報や稼働情報を取得し，その特性情報を抽出するモデルを検討する．特に，アプリに関するリソース割当量を推論するリソース量推論モデルの検討を行った．本稿では，アプリに関するリソース割当量を推論するリソース量推論モデルについて検討した内容を報告する．

2. リソース量推論モデルの課題

2.1 アプリケーションの最適配置の概要

ハイブリッドクラウド環境では，工場の製造ラインと連携した現場における装置監視ならびにデータ収集システム，オンプレミス環境で利用していた IT インフラ設備で稼働していた IT システム，および，パブリッククラウドサービスを連携させて構築したデータ利活用環境を利用するケースが増えている．

ハイブリッドクラウド環境において，コンテナ実行環境向けのコンピュートリソースならびにデータストア環境におけるデータ配置を適正に制御することが必要不可欠となる．また，サイト間のネットワークリソースの物理制約，および，業務アプリやサービスの特性をふまえて，業務アプリやサービスが利用データに効率よくアクセスできるようにする必要がある．

著者らは，ハイブリッドクラウド環境のデータストレージリソースならびにコンピューティングリソースを賢く使いこなすサービスを提供することが，利用者に対して様々な価値を提供でき有益と考え，当該サービスをデジタルインフラ面で支えるサービスやソリューション実現のための検討を進めている．

本検討では，データストレージ環境ならびにコンピューティング環境のインフラ稼働構成情報ならびに利用アプリなどの稼働構成情報をもとに，アプリやインフラをモデル化し，利用者のポリシーやデータ利用上の法規制などに基づいて，アプリ実行環境となるコンテナやデータの適正な配置場所を立案し，最適配置制御できるようにする．この最適配置制御の構成としては，コントロールプレーンとデータプレーンを利用することを想定する．

コントロールプレーンでは，データストレージ環境ならびにコンピューティング環境のインフラ稼働構成情報ならびに利用アプリなどの稼働構成情報をもとに，アプリやインフラをモデル化し，それらに基づいてハイブリッドクラウド環境におけるコンテナやデータの最適配置先を立案する．この最適配置先プランに基づいて，業務システムやサービスを稼働させるコンテナ実行環境は，適切なサイトにデプロイされる．

データプレーンは，各サイトのデータストアサービスを仮想的に一つに束ね，サイトを透過的なデータストアサービスとして提供し，その中で最適配置先プランなどに基づいて，データの適正な配置制御を実施する．各サイトのコンテナ実行環境では，業務アプリやサービスをコンテナ環境で稼働させ，データプレーンに格納される各データを利用した処理を行う．

コントロールプレーンでは，コンテナ実行環境に割り当てるリソース量を推論するモデルや，各サイトのデータストレージ環境やコンピューティング環境のインフラ稼働構成情報を管理し余剰リソースなどを推論するモデルなどが想定され，それぞれのモデルの推論結果をハイブリッドクラウド環境におけるコンテナやデータの最適配置処理で利用される．

著者らは，主に最適配置先へのデプロイやコンテナ実行環境に割り当てるリソース量を推論するリソース量推論モデルの研究開発を推進している．

2.2 リソース量を推論するための課題

コンテナ実行環境に割り当てるリソース量を推論するリソース量推論モデルでは，モデル対象のアプリを開発環境に準備し，テスト実行を行うことでコンピューティング環境で稼働する対象アプリなどのメトリクス情報を取得し，この取得したメトリクス情報に基づいて対象アプリのリソース利用状況などの気づきや特徴をアプリの特性情報として学習し，最適配置に必要な CPU やメモリのリソース量に関する情報，および，IO やネットワークに関する情報を推論するモデルを作成する．

このモデル作成では，対象アプリを最適配置させるために，それぞれのリソース量を推論できるモデルの立案が必要となる．しかし，対象アプリにより特徴や挙動などが異なるため容易にモデルを立案することができない課題がある．

この課題を解決するために，まずは，特定のアプリに対して 3 つのアプローチでモデル立案を行う．これによりモデル立案のアプローチの妥当性を調査することができる．なお，モデル立案のアプローチの概要を図 1 に示す．

まずは，モデル化対象となるアプリの稼働状況を確認するために，通常業務が行われる本番環境とは別に対象アプリをデプロイした開発環境を準備し，対象アプリに合わせて計画した負荷やテストケースに基づいてテスト実行を行い，対象アプリの稼働状況を表すメトリクス情報を取得する．このテストケースでは，対象アプリの入力情報やインフラのリソース情報がテストパターンとなる．

次に，取得したメトリクス情報に基づいて対象アプリの処理特性を分類する．この対象アプリの処理特性の分類では，取得したメトリクス情報の稼働状況から対象アプリの処理時間に影響を及ぼす要因を特定する．この要因として，対象アプリの単体性能に CPU コア数やメモリ量が関連し，対象アプリのデータストアへのアクセス性能に IO が関連し，対象アプリを構成する複数のコンテナなどの要素間の通信性能にネットワーク帯域が関連することが考えらえる．また，対象アプリの処理特性としては，複数の要因が対象アプリの処理時間に影響を与えることが予想される．

最後に，対象アプリの処理特性に基づき，メトリクス情報から対象アプリに割り当てるリソース量を推論するモデルを立案する．このモデル立案では，対象アプリに割り当てる CPU コア数を対象としたモデル化を推進する．これは，IO やネットワークについては，モデル化対象の範囲がアプリを構成する複数のコンテナの中のコンテナ単体からサイト間をまたぐ複数のコンテナ単位まで幅広く，各サイトのインフラ構成やリソースの利用状況など検討対象が広範囲となるため，まずは小規模な状態でのモデル立案として割り当てる CPU コア数をモデル立案の対象とした．

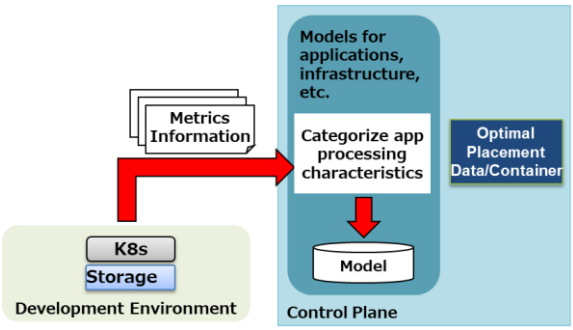

図 1　モデル立案のアプローチ概要

3. リソース量推論モデルの提案

3.1 アプリケーションの処理特性の分類定義

リソース量推論モデルでは，最初のアプローチとして，対象アプリの稼働情報を取得する．この稼働情報の取得に関しては，SoundCloud 社が中心になって開発しているプル型のリソース監視ソフトウェアである Prometheus[3]や Redhat が提供するリアルタイムデータの監視・管理を行う Performance Co-Pilot(PCP)[4]など様々なツールが提供されており，これらツールにより対象アプリのメトリクス情報を取得することができる．

本節では，上述したツールを利用して取得した対象アプリのメトリクス情報に基づいて対象アプリを分類する処理特性について説明する．対象アプリの処理特性として，CPU-intensive，Memory-intensive，IO-intensive，NW(NetWork)-intensive の 4 つを定義し，これらの定義を対象アプリのメトリクス情報から処理特性として分類する．このアプリ特性の分類を表 1 に示す．

CPU-intensive には，CPU 利用量が大きく，CPU の割当待ちとなる CPU スロットル[5]が多発するなどで，対象アプリの処理時間に影響を与える要因が CPU コア数不足となる対象アプリが分類される．また，対象アプリの CPU スロットル回数と CPU 割当回数の比率から閾値判定により分類することも可能と考えている．ここに分類される対象アプリの例として，大量の CPU を利用する機械学習処理や深層学習処理を実行するアプリが考えられる．

Memory-intensive には，メモリ利用量が大きく，メモリのスワップが多発するなどで対象アプリの処理時間に影響を与える要因がメモリ割当量の不足となる対象アプリが分類される．ここに分類される対象アプリの例として，データ取得，前処理，学習といった多段の処理を実行するアプリが考えられる．

IO-intensive には，ファイルシステムや DBMS(Data Base Management System)へのアクセス頻度やデータ量が大きく，アプリとデータの配置や IO 帯域の共有利用などで対象アプリの処理時間に影響を与える要因が IO 帯域の都合により IOPS(Input/Output Per Second)やスループットの不足となる対象アプリが分類される．ここに分類される対象アプリの例として，深層学習用などに大量のデータを取得するアプリが考えらえる．

NW-intensive には，コンテナ間通信の頻度やパケットサイズが大きく，複数コンテナの配置サイトの関係や，相対的に各コンテナの配置が遠隔地となった場合などで対象アプリの処理時間に影響を与える要因がネットワーク帯域の都合によりスループットの不足となる対象アプリが分類される．ここに分類される対象アプリの例として，多数のコンテナ群からなるマイクロサービス構成で，深層学習用などに大量のデータを取り扱うアプリが考えられる．

表 1　アプリの処理特性の分類定義一覧

#	App Characteristics	Description
1	CPU-intensive	Services with high CPU usage. CPU throttle generation affects execution time.
2	Memory-intensive	Services with high Memory usage. Restart due to OutOfMemory. Swap processing affects execution time.
3	IO-intensive	Services with frequent file system and DB access. Container and data store deployment locations affect execution time.
4	NW-intensive	Service with frequent inter-container communication. Container and data store deployment locations affect execution time.

3.2 リソース量推論モデルの立案

割り当てる CPU コア数を推論するリソース量推論モデルでは，対象アプリの CPU スロットル回数と CPU 割当回数の比率を閾値として CPU 割当モデルを 2 つに仕分ける．次に，仕分けた結果に基づいて対象アプリに割り当てる CPU コア数を推論するリソース量推論モデルを立案する．なお，当該モデルの立案までの流れを図 2 に示す．

当該モデルの仕分けについては，対象アプリの CPU 割当回数に対する CPU スロットル回数の比率が高い場合には，CPU コア数の不足により対象アプリの処理時間に影響を与えると判断し，対象アプリに割り当てる CPU コア数を推論するリソース量推論モデルを立

案する．CPU スロットル回数の比率が低い場合には，CPU コア数が十分であり，これ以上 CPU コア数を増やしても対象アプリの処理時間に影響を与えないと判断し，対象アプリに割り当てる CPU コア数に，開発環境の対象アプリに割り当てた CPU コア数の最小値を設定する．但し，CPU スロットル回数の比率を判断する閾値については，開発環境で取得したメトリクス値から指定する．

対象アプリに割り当てる CPU コア数を推論する当該モデルでは，CPU スロットル累計時間，入力負荷(入力データ数)，CPU コア数に着目して立案する．具体的には，開発環境でテスト実行して取得したメトリクス値において，CPU スロットル累計時間と CPU コア数の関係を，入力負荷が最小と最大となる 2 つのパターンでグラフ化する．グラフ化した例を図 3 に示す．

このモデルでは，入力負荷が最小となる CPU スロットル累計時間と，入力負荷が最大となる CPU スロットル累計時間との差分が所定の値以下となる点を算出し，その点に対応する CPU コア数を対象アプリの推論値とする．この CPU スロットル累計時間の差分は，対象アプリに CPU コア数を十分に割り当てられれば，小さくすることが可能であるが，実際には，対象アプリに割当可能な CPU コア数に制限があるため，CPU コア数を推論する当該モデルの対象とする．また，CPU スロットル累計時間の差分を判定する所定の値については，開発環境で取得した対象アプリのメトリクス値の分散などから選定する．

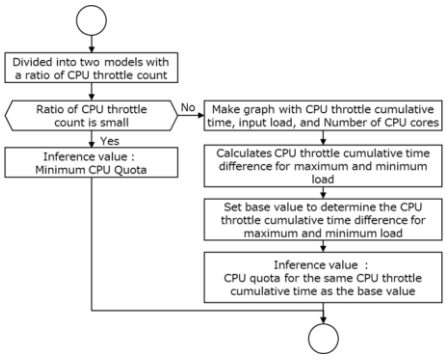

図 2　リソース量推論モデルの立案フローチャート

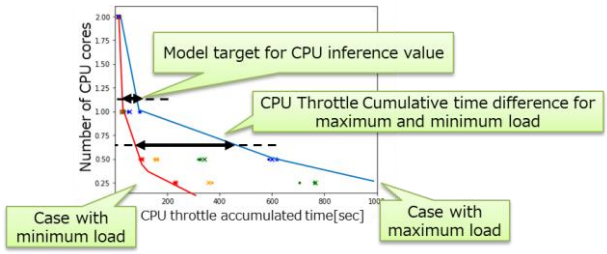

図 3　CPU スロットルのグラフ概要

4. リソース量推論モデルの原理検証

4.1 リソース量推論モデルの検証方法

リソース量推論モデルを実機環境で検証するために，テストパターンを策定し，テスト実行によるメトリクス情報の取得・可視化により検証する．なお，検証に利用した実機環境を図 4 に示す．

実機環境には，VM(Virtual Machine)上に対象アプリとして自前で準備した画像異常検知アプリ，対象アプリを実行するクライアントプログラム，メトリクス情報を取得する Prometheus を構築する．

対象アプリは，画像診断用のモデル学習や画像診断モデルを利用した推論を提供する GUI(Graphical User Interface)と，画像診断用のモデル学習や推論に関する機能とその API(Application Programming Interface)を提供する対象アプリ本体で構成され，をアップロード処理，学習処理，推論処理の 3 つの処理を実行する．

クライアントプログラムでは，学習/推論用データ(画像データ)のアップロード処理を対象アプリに対して実行する．このアップロードされた画像データは，対象アプリのコンテナにローカルファイルとして格納され，学習処理や推論処理に利用される．また，クライアントプログラムでは，対象アプリの各処理を実行し，各処理の実行時間の測定や，メトリクス情報の取得を行う．

リソース量推論モデルの立案可否の検証では，まず，当該実機環境を利用して表 2 に示すように，CPU コア数を 2 パターン，学習用データ数/推論用データ数を 2 パターン，IO 制限を 3 パターンの計 12 パターンをテスト実行し，対象アプリのメトリクス情報を取得する．次に，CPU コア数に対する CPU スロットル累計時間のグラフから割り当てる CPU コア数を推論するリソース量推論モデルを立案する．最後に，推論した CPU コア数を実機で確認し，想定する処理時間を得られるか検証する．

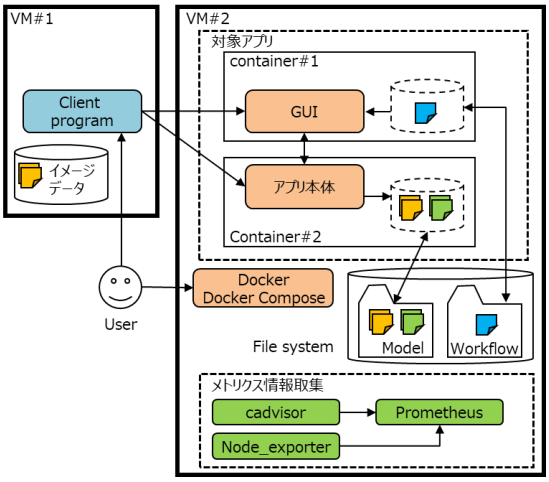

図 4　対象アプリの概要

表 2　テストケース

#	parameter	value
1	CPU core	0.25, 2.0
2	Memory Size[Mbyte]	2000
3	Number of learning data	100, 500
4	Number of inference data	100, 500
5	Limit IO rate[Bytes per second]	Nolimit, 10M, 1M

4.2 リソース量推論モデルの検証結果

実機環境でテスト実行した時の対象アプリのアップロード処理と学習処理の処理時間，および，アップロード処理と学習処理の処理時間の比率を図 5 に示す．

IO 制限がない場合，アップロード処理よりも学習処理の処理時間が支配的となっており，データ数の増減に合わせて学習処理の処理時間が増加する．また，CPU コア数を増やすことで処理時間が短縮されることから IO 制限がない場合には，CPU-Intensive と分類できる．

IO 制限がある場合，IO 制限の増減に合わせてアップロード処理の処理時間が増加しており，学習処理よりもアップロード処理の処理時間が支配的となっている．また，CPU コア数を増やすことで若干処理時間が短縮するが，IO 制限による処理時間の影響が大きいため，IO-intensive と分類できる．

リソース量推論モデルの仕分けとして，対象アプリの CPU 割当回数に対する CPU スロットル回数の比率に閾値を設定し，当該モデルの対象かどうかを判定する．ここでの当該モデルの対象には，閾値を超えるテストケースを選定した．

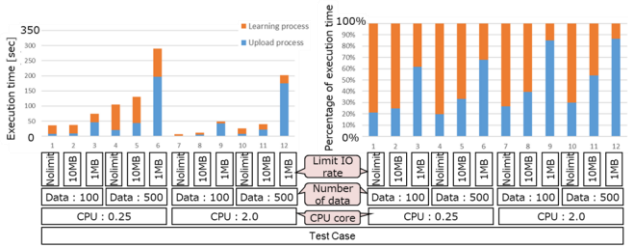

図 5　対象アプリの実行時間内訳

当該モデルの対象について，学習用データ数が最小となる CPU スロットル累計時間と，学習用データ数が最大となる CPU スロットル累計時間との差分を図 6 に示す．また，リソース推論モデルで推論した CPU コア数を実機環境の対象アプリに割り当てて実測した処理時間(レスポンスタイム)を図 7 に示す．

本検証では，CPU コア数を 0.25，0.50，1.0，2.0 に刻み，学習データ数を 100，200，500，1000 と拡充したテスト実行を行っている．学習用データ数が最小となる CPU スロットル累計時間と，学習用データ数が最大となる CPU スロットル累計時間との差分が放物線となっている．ここでは，CPU 割当量を推論するための所定の値を 50 と定義し，CPU コア数を 1.0 と推論している．

また，CPU コア数を 1.0 として実測した際には，CPU 不足による影響を受けずに対象アプリの処理時間を維持することができおり，CPU コア数を 1.0 未満に割当てた際には，対象アプリの処理時間が大きくなることがわかる．

以上のことから，対象アプリの処理特性の分類を行い，取得したメトリクス情報から対象アプリに対するリソース量推論モデルを立案でき，当該モデルで対象アプリに割り当てる CPU コア数を推論できる見通しを得た．また，CPU 割当モデルで推論した CPU コア数を対象アプリに割り当て実機評価することで処理時間を維持できることを確かめられた．

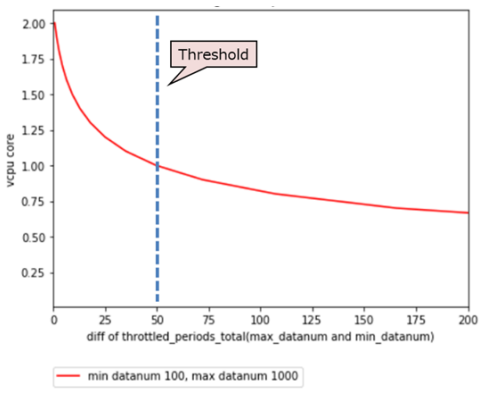

図 6　CPU スロットル累計時間の差分

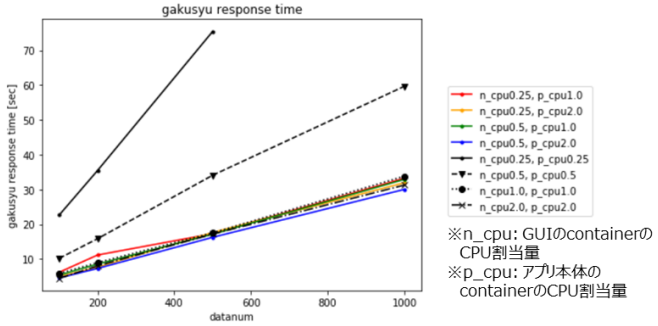

図 7　対象アプリの実行時間検証結果

5. 考察

本研究では，ハイブリッドクラウド環境におけるコンテナ・データ最適配置を実現するために，対象となるアプリやインフラに関する構成情報や稼働情報を取得し，その特性情報を抽出するモデルを検討し，特に，アプリに関するリソース割当量を推論するリソース量推論モデルを提案した．

コンテナで動作するアプリの処理特性をモデル化する取り組みについては，各ベンダで実施されており，コンテナ稼働情報を収集し，必要な CPU コア数/メモリ量を算出し，サイジングやオートスケーリングへの適用例が多い．また，パブクラでの性能/コスト最適化への取り組みが行われているが，最適配置への適用例はまだ少ない．

IBM では，watson クラウドの DLaaS(DataLake as a Service)のメトリクスから DL(Deep Learning)処理コンテナのモデル化を取り組んでいる[6][7]．CPU とメモリのメトリクスのみ利用してモデル化を行っており，IO やネットワークは future work となっている．コンテナの複数拠点に跨ったデプロイや最適配置は未対応となっている．

Google では，自社のコンテナ基盤 borg にて autopilot を導入している[8]．コンテナからメトリクス情報を収集し，Machine Learning でモデル化を行っている．また，サイジングやオートスケールに適用し slack(リソースの割当量と実利用量のギャップ)を低減させている．IO やネットワークの取り組み，および，コンテナの複数拠点に跨ったデプロイや最適配置への対応は確認できていない．

NEC では，VNF(Virtualized Network Function)のリソースサイジングと配置の最適化がおこなわれており[9]，コンテナの複数拠点に跨ったデプロイや最適配置に近い取り組みが行われている．

今後，コンテナ・データ最適配置には，アプリの処理特性に合わせた割当リソース量を推論するリソース量推論モデルと連携して，コスト面，性能面，セキュリティ面などの各ポリシーやモデルに合わせてコンテナとデータの両方を最適配置させる手法の確立が必須になると考えている．

6. まとめ

ハイブリッドクラウド環境におけるコンテナ・データ最適配置を実現するために，アプリに関するリソース割当量を推論するリソース量推論モデルを提案した．

リソース量推論モデルで対象アプリに割り当てる CPU コア数を推論できるか確認するために，実機環境を構築して，検証した．

検証結果からアプリに割り当てる CPU コア数を推論するリソース量推論モデルでの立案を行う一連の流れを実機環境で実際に行えることを確認できた．また，当該モデルで推論した CPU コア数を実際に対象アプリに割り当て，対象アプリの処理時間を実機環境で実測した所，推論した CPU コア数であれば処理時間を維持することが確かめられ，推論した CPU コア数未満を割当て際に処理時間が CPU 不足で大きくなることを確かめられた．

今後は，当該モデルにおいて人手で設定する判断基準を対象アプリのメトリクス情報などから選定できる仕組みの検討，および，メモリ，IO，ネットワークに関しても同様に推論するリソース量推論モデルの研究開発を推進する．

参　考　文　献

[1] IDC, “Worldwide Big Data and Analytics Software Forecast, 2019–2023,” IDC Doc #US44803719, (2019).

[2] IDC,“Worldwide Internet of Things Forecast, 2019-2023,” IDC Doc #US45373120, (2019).

[3] Prometheus,“From metrics to insight Power your metrics and alerting with a leading open-source monitoring solution,” https://prometheus.io/

[4] Redhat, “PCP を使ってパフォーマンスの謎を数分で解く,” https://www.redhat.com/ja/blog/pcp%E3%82%92%E4%BD%BF%E3%81%A3%E3%81%A6%E3%83%91%E3%83%95%E3%82%A9%E3%83%BC%E3%83%9E%E3%83%B3%E3%82%B9%E3%81%AE%E8%AC%8E%E3%82%92%E6%95%B0%E5%88%86%E3%81%A7%E8%A7%A3%E3%81%8F

[5] Indeed 社, “スロットリング解除：クラウドにおける CPU の制限の修正,” https://jp.engineering.indeedblog.com/blog/2019/12/%E3%82%B9%E3%83%AD%E3%83%83%E3%83%88%E3%83%AA%E3%83%B3%E3%82%B0%E8%A7%A3%E9%99%A4-%E3%82%AF%E3%83%A9%E3%82%A6%E3%83%89%E3%81%AB%E3%81%8A%E3%81%91%E3%82%8B-cpu-%E3%81%AE%E5%88%B6%E9%99%90%E3%81%AE/

[6] D. Buchaca, J. L. Berral, C. Wang and A. Youssef, "Proactive Container Auto-scaling for Cloud Native Machine Learning Services," 2020 IEEE 13th International Conference on Cloud Computing

(CLOUD), Beijing, 2020, pp. 475-479, doi: 10.1109/CLOUD49709.2020.00070.

[7] J. L. Berral, C. Wang, and A. Youssef, “AI4DL: Mining behaviors of deep learning workloads for resource management,” in 12th USENIX Workshop HotCloud, 2020.

[8] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, and John Wilkes. 2020. Autopilot: workload autoscaling at Google. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20). Association for Computing Machinery, New York, NY, USA, Article 16, 1?16. DOI:https://doi.org/10.1145/3342195.3387524

[9] M. Nakanoya, Y. Sato and H. Shimonishi, "Environment-Adaptive Sizing and Placement of NFV Service Chains with Accelerated Reinforcement Learning," 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 2019, pp. 36-44.