エッジコンピューティングにおけるコンテナ配置の最適化

15se006 深見　宗世 15se012 早川 剛生
指導教員：宮澤 元

1　はじめに

近年 Internet of Things(IoT) が普及している．IoT とはモノとモノのインターネットであり，接続されるモノを IoT デバイスと呼ぶ．多くの IoT デバイスはセンサやアクチュエータを備え，インターネット経由で情報を得たり，操作したりすることができる．この IoT デバイスを複数組み合わせ，IoT デバイスをインターネット上のソフトウェアで制御することにより IoT サービスが提供される． IoT サービスにはクラウドコンピューティングが用いられ，クラウドサーバに IoT デバイスを制御するソフトウェアを配置する．クラウドコンピューティングとは，インターネット上からハードウェアやソフトウェアなどのコンピュータ資源を必要な時に必要なだけ利用できるサービスであり，クラウドと省略されて呼ばれることが多い．また，クラウドサーバとはクラウドに存在する物理サーバのことである．IoT デバイスでデータを収集し，そのデータをクラウドサーバへ送り，そのデータを基にどのように IoT デバイスを制御すれば良いか分析する．そしてその結果を基に IoT デバイスへ指示を出す．このデータ収集/分析/指示によって IoT サービスは成り立っている．

しかし， IoT サービスを実現するにあたり，全ての IoT デバイスで生成したデータを全てクラウドサーバに転送した場合，クラウドサーバに送られるデータの量が膨大になると考えられる．これによりクラウドサーバでの処理が追いつかなくなるだけでなく，クラウドサーバへのネットワークトラフィックが増加してしまう恐れがある．

IoT デバイスからクラウドサーバへ送信するデータ量を低減する技術としてエッジコンピューティングが提案されている．これは， IoT デバイスの近くに設置されたエッジサーバを利用してクラウドサーバでの処理の一部を実行するものである．エッジコンピューティングにおける IoT サービスは Docker [2] などのコンテナ型仮想化を用いて実現されることが多い．コンテナ型仮想化では，ソフトウェアの実行環境をコンテナにまとめて実行する．コンテナは OS のカーネルが同じであればどのコンピュータでもプログラムの依存などを気にすることなく実行できるので，必要に応じて適切な処理ノードに配置できる．しかし，エッジコンピューティングにおいて IoT サービスを構成する処理を実行する適切な処理ノードは，処理の具体的な内容や処理ノードの負荷などに応じて異なるので，コンテナの最適な配置を判断するのは簡単ではない．

本研究では，エッジコンピューティングにおいて IoT サービスを構成するコンテナを最適なサーバで実行するための条件について検討を行う．具体的には，擬似的な IoT サービスを複数のコンピュータからなる実験環境で動作させる性能測定実験を行い，実験結果から最適なコンテナ配置を考察する． IoT サービスとしては我々が実装した簡単なデータ転送プログラムを用い，実験環境は Kubernetes [1] を利用して用意した．

2　研究の背景

2.1　IoT とエッジコンピューティング

IoT は幅広い分野で利用されるようになってきている．IoT とはモノとモノのインターネットであり，このモノは IoT デバイスと言われる．センサやアクチュエータが搭載されている IoT デバイスがデータを生成し，そのデータをクラウドサーバに送信する．クラウドサーバ上のソフトウェアはこのデータの分析を行い，分析結果に基づいて IoT デバイスへ指示を出す．このデータ生成/分析/実行によって IoT サービスは構成されている．IoT サービスの利点として，インターネット上から IoT デバイスによる監視，操作が可能になることが挙げられる．つまり， IoT サービスを利用することにより直接その場に行かなくても対象物を監視，操作することができる．

しかし，このクラウドサーバ/ IoT デバイスの 2 層で構成される IoT サービスには以下の問題が発生する．

1. 複数の IoT デバイスで生成したデータ全てをクラウドサーバへ送信することによるクラウドサーバへのネットワークトラフィックの増加
2. 複数の IoT デバイスで生成したデータ全てをクラウドサーバで処理することによるクラウドサーバでの膨大な負荷
3. IoT デバイス/クラウドサーバ間での大きな通信遅延の発生． 多くのクラウドサーバは海外に設置されており，例えば日米間で 130 ms から 250 ms もの RTT (往復時間) がかかる [4].

これらの問題の解決策として，エッジコンピューティングが提案されている．エッジコンピューティングとは，IoT デバイス付近にエッジサーバを置き，クラウドサーバで行っていた処理の一部をエッジサーバで行う技術である．エッジコンピューティングを利用することにより，上記の問題は以下のように解決される．

1. クラウドサーバへ送信されるデータ量が減少することによるクラウドサーバへのネットワークトラフィックの削減
2. クラウドサーバで行っていた処理の一部をエッジサーバで実行することによるクラウドサーバでの処理負担の軽減

3. クラウドサーバ・ IoT デバイス間にエッジサーバをはさみ，通信の応答時間が早まることによるリアルタイム性の向上

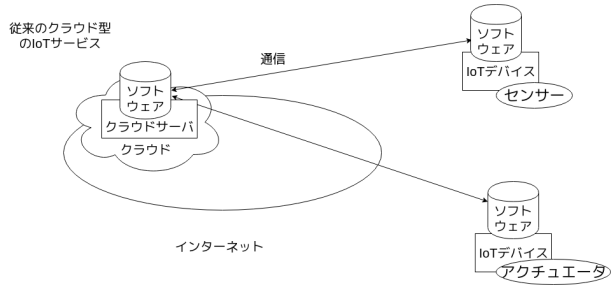

図 1 2 層の IoT サービス

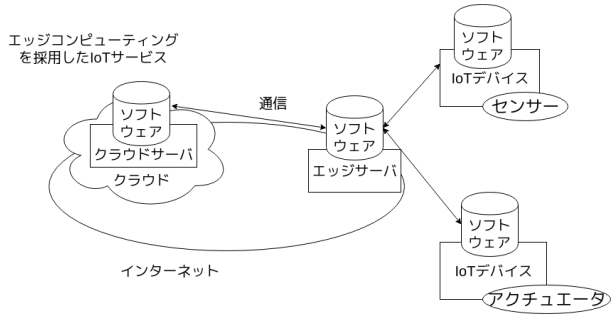

図 2 3 層の IoT サービス

図 1 はエッジコンピューティングを採用していない 2 層で構成された IoT サービスの図で，図 2 はエッジコンピューティングを採用した 3 層で構成された IoT サービスの図である．図 1 の場合， IoT デバイスの制御を全てクラウドサーバで行っているので負荷がクラウドサーバへ集中し，また，クラウドサーバと IoT デバイスの距離がとても長いので通信遅延の影響が大きくでてしまう．図 2 のように IoT デバイスの付近にエッジサーバを置き，クラウドサーバで行っている IoT デバイスの制御の一部をエッジサーバで行うことにより，クラウドサーバでの負荷が減少し，かつエッジサーバ/ IoT デバイス間の距離がクラウドサーバ/ IoT デバイス間と比べ近いので通信遅延の影響を抑え，リアルタイム性を向上させることができる．

2.2 コンテナ仮想化

IoT サービスの実現を支えている技術にコンテナ仮想化がある．コンテナ仮想化とは，プロセスを他のプロセスから隔離されたコンテナと呼ばれる実行環境の中で動作させる技術である．名前空間とリソース管理をコンテナごとに分離することで，互いの影響を気にすることなく，コンテナを実行させることができる．

コンテナが IoT に利用される理由として，コンテナ化したプログラムはコンテナを実行できる環境ではどのコンピュータでも実行できることが挙げられる．この特徴を利用して，コンテナをクラウドサーバからエッジサーバ，エッジサーバから IoT デバイス，また IoT デバイスから別の IoT デバイスとノード間でマイグレーションさせることができる．

コンテナマイグレーションを利用して，クラウドサーバ/ エッジサーバ/ IoT ゲートウェイの三層構造の IoT サービスを実現する先行研究がある [3]．この研究では， IoT デバイスの移動に対応したり， IoT ゲートウェイの過負荷を解消するためにコンテナマイグレーションを利用している．

しかし，コンテナを自動配置させる際，どのコンテナをどこに配置させるかを自動的に様々なケースで決めさせるのは難しい．もし，処理能力が大きいクラウドサーバで処理するべき処理をエッジサーバで実行してしまうと，エッジサーバはクラウドサーバと比べて処理能力が低いので，逆に処理時間が遅くなってしまう．例え，エッジサーバに処理を代行させてクラウドサーバでの負荷を減らすことができても， IoT サービスの応答時間が遅くなってしまってはエッジコンピューティングを採用する意義がなくなってしまう．

3 エッジコンピューティングにおけるコンテナ配置の最適化に向けた検討

本研究では，クラウドサーバ，エッジサーバ， IoT デバイスのそれぞれの処理能力や通信速度，遅延に着目し，その値の変化によってどのように処理時間が変化するかを調べる．今回はクラウドサーバ/エッジサーバ/ IoT デバイスの三層構造の IoT サービスを考え，このサービスに必要な処理をコンテナ化し各ノードに配置させる．どのコンテナをどのノードで実行すればよいか，実験を行った上で考察する．

3.1 想定する IoT サービス

ここでは， IoT デバイスからサーバに何らかのデータを送信する IoT サービスを想定する．データには，サーバまたは IoT デバイスで何らかの処理を加える．この IoT サービスについて，以下の条件がどのように影響するかを実験で調べる．

データ処理のパターン IoT デバイスでデータ処理を行ってからデータ送信を行うパターンと，データ送信を行ってからサーバでデータ処理を行うパターン

サーバの場所 クラウドサーバを用いる場合と，エッジサーバを用いる場合

データ処理の負荷 処理負荷が大きいものと小さいもの

4 実験

通信環境や処理の負荷の違いなどの条件の違いによる最適なコンテナ配置の変化を調べるために実験を行なった．今回の実験では，エッジコンピューティングを導入した様々な IoT サービスにおいて，どのようなコンテナをエッジサーバで実行すればよいか検討するため，クラウドサー

表 1 実験に用いたコンピュータの仕様

ノード	マスターノード	クラウドノード	エッジノード	IoT デバイスノード
種別	ノート PC	デスクトップ PC	Raspberry Pi 3	
CPU	Intel® Core™i5-4210M@2.60GHz	Intel® Core™i7-7700K@4.20GHz	ARM Cortex-A53 4 コア 1.2GHz	
RAM	4GB	32GB	1GB	
OS	CentOS Linux release 7.5.1804 (Core)	Ubuntu 16.04.5 LTS	Raspbian Stretch Lite: debian_9.6	
通信速度	1Gbps	1Gbps	100Mbps	

バ，エッジサーバ，IoT デバイスのそれぞれの処理能力や通信速度，遅延に着目し，その値の変化によってどのように処理時間が変化するかを測定する．

IoT デバイスが使用できる帯域幅の違いと IoT デバイスとエッジサーバ，クラウドサーバの物理的距離を再現するため，通信レイテンシや通信帯域幅を変化させた．

4.1 実験環境

表 1 に示す 4 台のコンピュータで Kubernetes クラスタを構成し，実験に使用した．ノート型 PC をマスターノードとし，他のノードを操作する為に使用する．ほかの 3 台のコンピュータを，それぞれクラウドノード，エッジノード，IoT デバイスノードとし，Kubernetes で制御する．Kubernetes のバージョンは全て v1.12.3 であり，使用した Kubernetes のネットワークは flannel:v0.10.0 である．

4.2 実験内容

通信環境や処理の負荷の違いなどの条件の違いによる最適な IoT サービスのコンテナ配置の変化を調べるため，負荷がそれぞれ異なるデータ処理を各ノードで実行する実験と，データ処理によって生成されるデータを IoT デバイスからクラウドサーバまたはエッジサーバへ転送する実験を行う．

4.2.1 データ処理時間の測定実験

用意した 2 つのファイルを用いたデータ圧縮処理とデータ抽出処理をクラウドノードと IoT デバイスノードのそれぞれで実行し，データ処理にかかる時間を測定する．IoT デバイスノードとエッジノードには同じ仕様のコンピュータを利用しているので，エッジノードでの処理時間は IoT デバイスノードと同じだと考え，今回の実験では計測しない．データ処理の具体的な内容を以下に示す．

データ圧縮処理 ファイルサイズ 550.4MB の動画ファイルを圧縮する．圧縮後の動画ファイルサイズは 37.8 MB

データ抽出処理 ファイルサイズ 48.4MB の動画ファイルから 10 秒ごとの静止画を 7 枚の jpg 画像として取り出す．抽出後のファイルサイズ合計は 1.73 MB

4.2.2 データの転送時間の測定実験

IoT デバイスからクラウドサーバまたはエッジサーバへデータを転送する時間を測定する実験を行う．転送するデータは以下の 5 通りとする．

圧縮前動画 前節のデータ圧縮処理に用いたのと同じファイルサイズ 550.4MB の動画ファイル

圧縮後動画 前節のデータ圧縮処理で圧縮されたファイルサイズ 37.8MB の動画ファイル

抽出前動画 前節のデータ抽出処理に用いたのと同じファイルサイズ 48.4MB の動画ファイル

抽出後写真 前節のデータ抽出処理で抽出された合計ファイルサイズ 1.73MB の jpg ファイル 7 枚

センサ値 センサ値とみなして rand 関数を用いて生成した 0〜99 の乱数 10000 個

転送先のサーバには表 1 のクラウドノードとエッジノードを用いる．実際の通信環境を擬似的に再現するために，IoT デバイスノードとサーバとの通信レイテンシと帯域幅を tc コマンドを用いて変化させた．クラウドノードでは，通信レイテンシを変化させない場合と 50 ms，100 ms，200 ms を加えた場合の 4 通り，帯域幅は制限なし (100 Mbps) と 50 Mbps と 10 Mbps の 3 通りを設定した．エッジノードでは，通信レイテンシを変化させない場合と 5 ms を加えた場合の 2 通り，帯域幅は制限なし (100 Mbps) と 50 Mbps と 10 Mbps の 3 通りを設定した．なお，帯域幅は IoT デバイスノードの出力と入力の両方で制限した．通信レイテンシの制限は往復時間を想定しているので，IoT デバイスノードとサーバノードの出力側でそれぞれ設定値の半分ずつを加えた．

4.3 実験結果

4.3.1 データ処理時間の違い

データ圧縮処理とデータ抽出処理にかかった時間を表 2 に示す．それぞれ 10 回測定し，その平均値を結果とした．

クラウドノードが IoT デバイスノードよりもおよそ 30 倍から 40 倍ほど高速で処理ができることがわかった．単

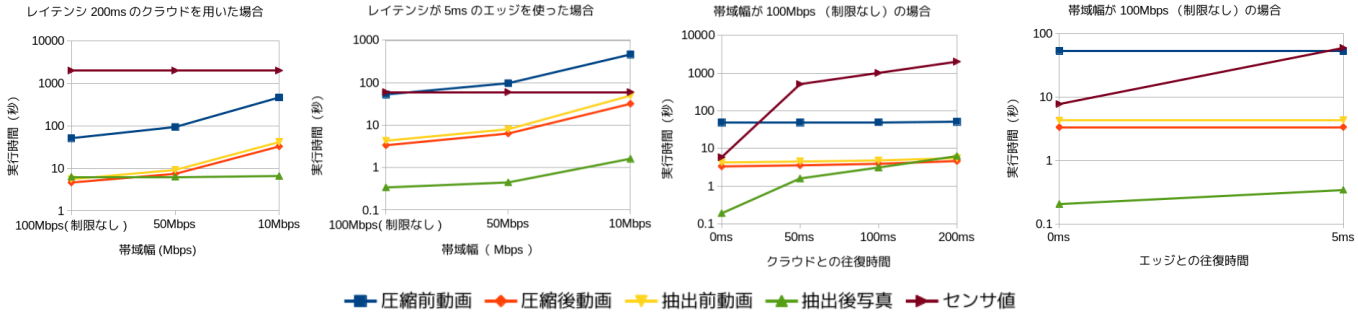

図 3　データ転送時間

表 2　データ処理時間

	IoT デバイスノード	クラウドノード
データ圧縮処理	1006.6 秒	27.8 秒
データ抽出処理	60.7 秒	2.33 秒

純に CPU の処理速度の違いだけではなく，メモリ容量などの他の部分の性能差や冷却性能の違いも結果に現れていると考えられる．

4.3.2　データ転送時間の違い

データ転送実験の結果を図 3 に示す．それぞれ 10 回測定し，その平均値を結果とした．

動画の転送では帯域幅の違いが転送時間に大きく関わり，レイテンシの違いによる変化はほぼなかった．そして動画のファイルサイズが大きいほど帯域幅による違いが大きくなった．写真の転送では，レイテンシによる転送速度の違いが帯域幅による違いよりもやや大きく，レイテンシが小さいほど帯域幅による転送速度の違いが大きくなった．センサ値の転送では，レイテンシの違いによって転送時間が大きく異なり，帯域幅による違いがほとんどなかった．また，クラウドノードへ転送する場合，帯域幅によってはデータ抽出後よりデータ抽出前の方が転送時間が短い場合がある．これはデータ抽出をすることによりデータ転送の往復回数が増加したためだと考えられる

4.4　考察

2 桁の整数のようなごく小さいデータを多数送信する場合には，レイテンシが小さいエッジサーバに送信して処理をした方が効率がよく，動画のようなファイルサイズが大きく処理に時間のかかるものの場合には，処理能力が高いクラウドサーバで処理をしたほうが時間はかからなかった．しかし，エッジサーバに処理能力が高いデバイスを採用すれば，エッジサーバで処理をしたほうが転送するファイルサイズを小さくでき，かつクラウドサーバへのネットワークトラフィックを減少できる．また，データ抽出を行う際には，データ抽出による転送回数の増加も考慮に入れなければならない．

5　おわりに

本研究では，エッジコンピューティングを利用した IoT サービスにおいて IoT サービスを構成するコンテナの内，どのようなコンテナをエッジサーバで実行させるべきか検討を行った．エッジサーバで実行させるとクラウドサーバで実行させるより処理時間が短くなるコンテナを明らかにするため，クラウドサーバ，エッジサーバ， IoT デバイスのそれぞれの処理能力や通信速度，遅延の変化による処理時間の変化を測定した．実験の結果，データ送信回数やファイルサイズに応じてコンテナを動作させるサーバを変更することにより，処理時間を短縮できることが分かった．

今回の実験では，処理に用いるデバイスを 2 種類しか用意できず，処理能力の違いによる処理時間の変化が単純な結果しか得られなかった．よって，今後は多くの種類のデバイスを用意して処理や転送にかかる時間を測定することで，クラウドサーバ，エッジサーバ， IoT デバイスの処理能力の違いによって処理時間がどのように変化するのかを明らかにする必要がある．

参考文献

[1] Google.Inc: “Kubernetes” :https://kubernetes.io/ (2019/1/3) .

[2] Docker.Inc: “Docker” :https://www.docker.com/ (2019/1/3) .

[3] Corentin Dupont, Raffaele Giaffreda, Luca Capra: “Edge computing in IoT context: horizontal and vertical Linux container migration”, in Proceedings of 2017 Global Internet of Things Summit (GIoTS), pp.1–4, 2017.

[4] 建部 修見，中村 昌弘，神林 亮，平賀 弘平，鈴木 克典，木村 浩希，小林 賢司，三上 俊輔 : スケーラブルな広域ファイルシステムの研究 https://tsukuba.repo.nii.ac.jp/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=19225&item_no=1&page_id=13&block_id=83 .