エッジコンピューティングにおける I/O バウンドサービスへのアクセス時間の比較

2016SE052 松林佑斗　2017SE045 松井洸貴
指導教員：宮澤元

1　はじめに

身の回りのあらゆるものをインターネットに接続する Internet of Things(IoT) が普及し，各種のセンサを備えた家電製品などが IoT デバイスとしてネットワークに接続されるようになっている．IoT デバイスの処理能力は一般に高くないので，センサで取り込んだデータをクラウドコンピューティング (クラウド) などの外部の計算リソースを用いて処理する必要がある．しかし，IoT デバイスからのデータ処理を全てクラウドで行うには問題がある．まず，IoT デバイスからクラウドへのデータ処理要求が集中してしまう．次に，クラウドが IoT デバイスからネットワーク的に遠距離にあり，アクセスレイテンシが大きくなってしまうことが挙げられる．

こういったクラウドの問題点を解決するために利用者からネットワーク的に近い距離にある計算リソースを用いて処理を行う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を用いることで，クラウドへのデータ処理やネットワーク負荷の集中を軽減することができる．一方で，エッジコンピューティングを用いて IoT デバイスのデータ処理を行う場合，多様な計算ノードが利用されることを考慮する必要がある．

本研究の目的は，エッジコンピューティングにおいて利用される計算ノードによって I/O バウンドサービスの処理性能が異なることを示すことである．このような違いを理解することはエッジコンピューティングにおいて利用する計算ノードを適切に選択できる基盤ソフトウェアを開発する上で重要である．

研究課題は以下の 2 点である．

- サービスインスタンスが動作する計算ノードによる I/O バウンドサービスへのアクセス時間の違いを確認する
- エッジコンピューティングにおけるサービスインスタンスの選択手法について考察する

具体的には，異なる種類の計算ノードを利用するコンテナオーケストレータを用いて，ファイル転送サービスにアクセスする実験を行い，処理時間を計測する．サービスインスタンスが動作する計算ノードの違いによる処理時間の違いを調べる．実験結果に基づき，エッジコンピューティングにおけるサービスインスタンスの選択手法について考察する．

2　研究の背景

本節では，研究の背景として Kubernetes とエッジコンピューティングについて述べる．

2.1　Kubernetes

Kubernetes はクラウドコンピューティングにおけるアプリケーションコンテナの運用自動化のために設計されたオープンソースのプラットフォームである [1]．Kubernetes の登場により，複数の Docker などのコンテナの管理や自動化が進んだことによって，この仕組みは「コンテナオーケストレーション」と呼ばれるようになった．

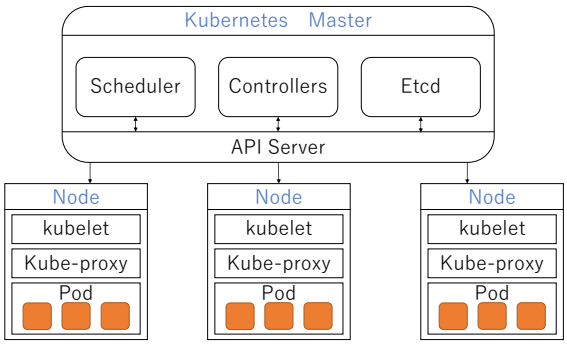

図 1　Kubernetes アーキテクチャ

図 1 に Kubernetes の基本的なアーキテクチャを示す．全体の制御を行う Kubernetes Master にコンテナのスケジューラやコントローラ，クラスタ全体の情報を管理する etcd が存在している．一方，アプリケーションコンテナは Node で動作し，Pod という単位で管理される．Pod とは一つ以上のコンテナをまとめたものを指す．Pod 上のアプリケーションを公開するには Service という抽象的な方法を用いる．各 Node にはその Node を管理するための Kubelet や Node ネットワークを扱う Kube-proxy が動作している．Kube-proxy は Kubernetes Master を監視し，各 Service でその Service に対するトラフィックの行き先を対応する Pod がある場所に変更する．このとき，標準では Kube-proxy は Pod をランダムに選択する．

2.2　エッジコンピューティング

エッジコンピューティングとは，利用者に近いエリアのネットワークに分散配置されたエッジデバイスを用いて処理を行うコンピューティングモデルである．特に多数の IoT デバイスからのデータを処理する方法として注目され

ている [2]．IoT デバイスの処理性能はそれほど高くないので，IoT デバイスからのデータはクラウドで処理されていた．しかし，多数の IoT デバイスからのデータ処理要求が集中するとクラウドの処理負荷やネットワークトラフィックが増大してしまう問題点がある．また，IoT デバイスとクラウドのデータセンタ間の通信レイテンシは一般に大きく，リアルタイム性が必要な IoT アプリケーションのデータ処理をクラウドで行うのは難しい．エッジコンピューティングを用いることにより，このようなクラウドの問題点を軽減できる．

エッジコンピューティングにおいてアプリケーションを実行するために，クラウドコンピューティングと同様のコンテナ実行環境を提供するエッジコンピューティング基盤が必要となる．クラウドにエッジコンピューティングを組み合わせることでクラウドに足りない要素を補うことが期待されており，アプリケーションを実行するコンテナを最適に配置することが重要となる．

既存のエッジコンピューティング基盤として，Kubernetes を拡張した様々なシステムが提案されている．Cloud4IoT[2] はコンテナ化された IoT サービスをノード間でマイグレートし，コンテナを最適な配置で実行できる．KubeEdge[3] はエッジノードとマスター間のプロトコルを拡張し，エッジノードをオフラインモードで動作させたり，MQTT を用いて接続するデバイスとの連携を行うことができる．

3 エッジコンピューティング基盤における計算ノードの I/O 性能

エッジノードにはさまざまな種類の計算ノードが利用されるが，そこで用いられる入出力装置もさまざまである．PC の場合，入出力装置には HDD(ハードディスクドライブ) や SSD(ソリッドステートドライブ) など，大容量かつ比較的データ転送が高速な装置が使われることが多い．一方，小型の組込みコンピュータの入出力装置には入出力性能があまり高くない SD カードなどが使われる場合がある．

HDD や SSD の転送速度は数百 MB/秒程度以上であり，その入出力性能や容量を活かして高画質動画などのサイズが大きいデータの保存や処理に用いられる．これに対し，SD カードはデジタルカメラや音楽プレーヤなどの機器の記録メディアとして作られており，主な用途は写真や音楽などの比較的サイズが小さいデータの保存や持ち運びである．そのため SD カードの記憶容量は比較的小さく，データ転送も数十 MB/秒程度と低速である．しかし，低消費電力で動作するという利点があり，入出力速度がさほど要求されない組込みコンピュータには向いていると言える．

このように，エッジコンピューティング基盤で利用するエッジノードの持つ入出力装置はノードごとに異なる．それぞれの装置の入出力性能も異なるので，エッジコンピューティング基盤においてサービスを実行するノードの選択がサービスの処理性能に影響を与える可能性がある．

4 実験

計算ノードの違いにより入出力性能も異なることに着目し，異なる種類の計算ノードを利用するコンテナオーケストレータを用いて利用者ノードからサービスにアクセスを行い，アクセス時間を計測する実験を行う．

4.1 実験の目的

実験の目的は計算ノードの違いによるサービスアクセス時間の違いを確認することである．クラウドで利用されるような高性能なノード (クラウドノード) とエッジで利用されるようなノード (エッジノード) では入出力性能に差があるので，エッジコンピューティング基盤においてコンテナオーケストレータの計算ノードとして利用した時，クラウドノードとエッジノードでどのような差がでるかについて具体的な数値を確認する．

4.2 実験内容

クラウドノード 2 台とエッジノード 2 台を計算ノードとして持つ Kubernetes クラスタを作成し，IoT デバイスに見立てた利用者ノードからサービスにアクセスする実験を行った．サービスとして Web サーバを利用し，それぞれのクラウドノードとエッジノードで動作する Web サーバに利用者ノードからアクセスしてファイルを転送する時間を計測した．

実験を行った具体的なシステム構成を**図 2** に示す．利用者ノードがクラスタ外部から各ノードに設定した NodePort に対してアクセス要求を行うと，いずれかのノード上の Web サーバが動作しているポッドが選択されアクセスされる．通常，利用者がアクセスする NodePort が存在するノードと，実際にアクセスされるポッドが存在するノードは必ずしも同じではないが，デプロイメントに含まれるポッドを 1 つだけとして，あらかじめ指定したラベルがつけられたノードのみにポッドを配置することで実際にアクセスするノードを指定している．

4.2.1 ファイルの転送時間の測定

各ノードに作成したサイズの異なったファイルを用い，以下の実験を行う．

- クラウドノードから利用者ノードに対してのファイルの転送
- エッジノードから利用者ノードに対してのファイルの転送

アクセスするファイルのサイズによるアクセス時間の違いを確認するために，いくつかの異なるサイズのファイルを用意した．データ量の少ない場合として静止画，データ量の多い場合として高解像度の動画像を想定し，またデータ量とファイル転送時間の関係を調べるためにそれらの中間のデータ量のファイルを用意した。具体的なデータサイズは 1MB，512MB，1GB，2GB，3GB，4GB，5GB の 7 通

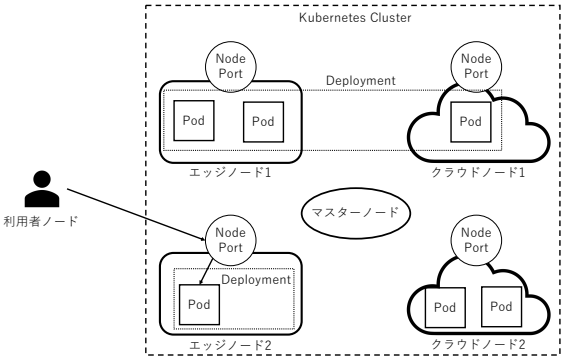

図 2 システム構成図

りである．

4.3 実験環境

実験にはコンテナオーケストレータとして Kubernetes を用いた．Kubernetes マスターノードとクラウドノード，利用者ノードには PC，エッジノードには NVIDIA Jetson Nano を用いた．利用した計算機の詳細を**表 1** に示す．各ノードは 1000Base-T Ethernet で接続した．サービスとして起動する Web サーバには nginx 1.19.6 を利用した．Web サービスへのアクセスは GNU wget 1.20.3 を用い，利用者ノードでのファイル保存のオーバヘッドが影響しないように転送したファイルの出力先を/dev/null とした．Kubernetes のバージョンは v1.19.2 である．

表 1 実験で利用する計算機の仕様

	Jetson Nano	PC
CPU	NVIDIA Tegra X1 @ 1.48GHz	Intel(R) Core(TM)i7-7700K CPI @ 4.20GHz
OS	Ubuntu 18.04.5	Ubuntu 20.04.2
カーネル	Linux 4.9.140	Linux 5.4.0-65-generic
メモリ	4GB	32GB
ストレージ	SanDisk SDSQXA1-128G-GN6MA	PLEXTOR PX-256M8SeGN
NIC	Realtek RTL8111 (64bit, 33MHz)	Intel I219-V (32bit, 33MHz)

4.4 実験結果

利用者ノードからクラウドノードとエッジノード上のポッドで動作する Web サーバにアクセスしファイル転送にかかる時間を計測した．転送するファイルのサイズは 4.2.1 節で示した 7 通りである．同じ設定の実験を 10 回ずつ行った．Web サーバがクラウドノード 1 で動作しているときの結果を**表 2**，クラウドノード 2 で動作しているときの結果を**表 3**，エッジノード 1 で動作しているときの結果を**表 4**，エッジノード 2 で動作しているときの結果を**表 5** にそれぞれ示す．また，実験における転送時間の平均をグラフにまとめたものを**図 3** に示す．表の括弧内の数値は読出し停止に伴うリトライ回数を示す．

表 2，**表 3** に示すクラウドノードからのファイル転送時間からは，クラウドノード同士の間でファイル転送時間に大きな差がないことがわかる．**図 3** からもファイル転送時間はファイルサイズにおおむね比例している．またファイルのサイズに関わらず，実験の回数ごとにファイル転送時間に大きなばらつきはない．ファイル転送においてリトライが発生していないことから，安定したファイル転送が行われていることがわかる．

表 2 クラウドノード 1 から利用者ノードに対しての各サイズのファイルの転送時間 (秒)

回数	1MB	512MB	1GB	2GB	3GB	4GB	5GB
1	0.03 (0)	5.53 (0)	11.04 (0)	22.07 (0)	33.20 (0)	44.25 (0)	55.19 (0)
2	0.02 (0)	5.52 (0)	11.02 (0)	22.14 (0)	33.14 (0)	44.25 (0)	55.26 (0)
3	0.02 (0)	5.51 (0)	11.00 (0)	22.17 (0)	33.05 (0)	44.19 (0)	55.18 (0)
4	0.02 (0)	5.52 (0)	11.10 (0)	22.19 (0)	33.13 (0)	44.11 (0)	55.12 (0)
5	0.02 (0)	5.52 (0)	11.13 (0)	22.09 (0)	33.12 (0)	44.25 (0)	55.07 (0)
6	0.02 (0)	5.53 (0)	11.06 (0)	22.06 (0)	33.13 (0)	44.13 (0)	55.20 (0)
7	0.02 (0)	5.56 (0)	11.04 (0)	22.09 (0)	33.13 (0)	44.13 (0)	55.34 (0)
8	0.02 (0)	5.59 (0)	11.05 (0)	22.04 (0)	33.15 (0)	44.13 (0)	55.17 (0)
9	0.03 (0)	5.53 (0)	11.04 (0)	22.09 (0)	33.21 (0)	44.09 (0)	55.33 (0)
10	0.03 (0)	5.52 (0)	11.09 (0)	22.12 (0)	33.08 (0)	44.25 (0)	55.12 (0)
平均	0.02	5.53	11.06	22.11	33.13	44.18	55.20
標準偏差	0.005	0.024	0.039	0.048	0.048	0.067	0.089

※ 括弧内はリトライ回数.

表 3 クラウドノード 2 から利用者ノードに対しての各サイズのファイルの転送時間 (秒)

回数	1MB	512MB	1GB	2GB	3GB	4GB	5GB
1	0.03 (0)	5.50 (0)	11.02 (0)	22.13 (0)	33.13 (0)	44.25 (0)	55.10 (0)
2	0.02 (0)	5.49 (0)	10.97 (0)	22.10 (0)	33.07 (0)	44.03 (0)	55.22 (0)
3	0.02 (0)	5.52 (0)	11.05 (0)	21.97 (0)	33.09 (0)	43.98 (0)	55.01 (0)
4	0.02 (0)	5.50 (0)	11.17 (0)	21.95 (0)	33.08 (0)	44.05 (0)	54.91 (0)
5	0.02 (0)	5.50 (0)	11.05 (0)	22.09 (0)	33.02 (0)	44.09 (0)	55.14 (0)
6	0.02 (0)	5.51 (0)	10.99 (0)	22.16 (0)	32.94 (0)	43.92 (0)	55.14 (0)
7	0.02 (0)	5.56 (0)	11.04 (0)	21.97 (0)	33.01 (0)	44.06 (0)	55.00 (0)
8	0.02 (0)	5.55 (0)	11.03 (0)	21.96 (0)	33.07 (0)	43.93 (0)	55.21 (0)
9	0.02 (0)	5.50 (0)	10.99 (0)	21.99 (0)	33.09 (0)	43.94 (0)	55.05 (0)
10	0.02 (0)	5.53 (0)	11.03 (0)	22.12 (0)	32.93 (0)	43.92 (0)	55.08 (0)
平均	0.02	5.52	11.03	22.04	33.04	44.02	55.09
標準偏差	0.003	0.024	0.055	0.083	0.067	0.103	0.097

※ 括弧内はリトライ回数

表 4，**表 5** に示すエッジノードからのファイル転送時間からは，エッジノード同士の間では，ファイルサイズが 5GB の場合の平均転送時間の差がやや大きい他は，ファイル転送時間に大きな差がないことがわかる．

表 4 エッジノード 1 から利用者ノードに対しての各サイズのファイルの転送時間 (秒)

回数	1MB	512MB	1GB	2GB	3GB	4GB	5GB
1	0.08 (0)	5.87 (0)	11.63 (0)	23.13 (0)	38.15 (1)	51.43 (1)	65.96 (2)
2	0.05 (0)	4.61 (0)	9.25 (0)	24.72 (0)	39.46 (1)	52.11 (1)	67.21 (2)
3	0.02 (0)	4.60 (0)	9.16 (0)	18.33 (0)	39.53 (1)	51.32 (1)	69.16 (2)
4	0.03 (0)	4.65 (0)	9.38 (0)	18.45 (0)	38.41 (1)	51.53 (1)	65.26 (1)
5	0.03 (0)	4.63 (0)	9.18 (0)	18.33 (0)	37.52 (0)	53.00 (2)	63.20 (1)
6	0.03 (0)	4.69 (0)	9.23 (0)	18.36 (0)	39.36 (1)	51.35 (1)	66.36 (2)
7	0.04 (0)	4.61 (0)	9.17 (0)	18.39 (0)	40.50 (2)	50.88 (1)	65.33 (2)
8	0.06 (0)	4.59 (0)	9.21 (0)	18.32 (0)	37.79 (0)	53.37 (2)	63.46 (1)
9	0.04 (0)	4.61 (0)	9.20 (0)	18.32 (0)	38.20 (1)	51.84 (1)	66.20 (2)
10	0.05 (0)	4.60 (0)	9.17 (0)	18.35 (0)	36.95 (0)	50.78 (1)	68.62 (3)
平均	0.04	4.75	9.46	19.47	38.59	51.76	66.08
標準偏差	0.018	0.396	0.766	2.378	1.092	0.851	1.936

※ 括弧内はリトライ回数

クラウドノードとエッジノードからのファイル転送時間をそれぞれ比較すると，ファイルサイズが 1MB の時はク

表 5　エッジノード 2 から利用者ノードに対しての各サイズのファイルの転送時間 (秒)

回数	1MB	512MB	1GB	2GB	3GB	4GB	5GB
1	0.07 (0)	6.18 (0)	11.76 (0)	23.54 (0)	37.75 (1)	53.94 (2)	63.05 (1)
2	0.05 (0)	4.64 (0)	11.93 (0)	25.16 (0)	39.61 (1)	51.19 (1)	64.89 (2)
3	0.04 (0)	4.61 (0)	9.19 (0)	20.04 (0)	39.40 (1)	50.95 (1)	63.95 (1)
4	0.02 (0)	4.60 (0)	9.24 (0)	18.38 (0)	37.81 (0)	51.03 (1)	63.17 (1)
5	0.04 (0)	4.63 (0)	9.34 (0)	18.39 (0)	39.14 (1)	51.10 (1)	62.98 (1)
6	0.04 (0)	4.64 (0)	9.18 (0)	18.35 (0)	38.61 (0)	49.54 (0)	62.91 (1)
7	0.04 (0)	4.59 (0)	9.18 (0)	18.39 (0)	38.12 (0)	50.77 (1)	64.89 (2)
8	0.04 (0)	4.60 (0)	9.18 (0)	18.45 (0)	38.05 (0)	50.88 (1)	63.37 (1)
9	0.04 (0)	4.60 (0)	9.17 (0)	18.30 (0)	38.43 (0)	52.09 (2)	64.61 (2)
10	0.04 (0)	4.67 (0)	9.22 (0)	18.32 (0)	38.05 (0)	49.47 (0)	62.94 (1)
平均	0.04	4.78	9.74	19.73	38.50	51.10	63.68
標準偏差	0.012	0.494	1.112	2.519	0.671	1.262	0.833

※ 括弧内はリトライ回数

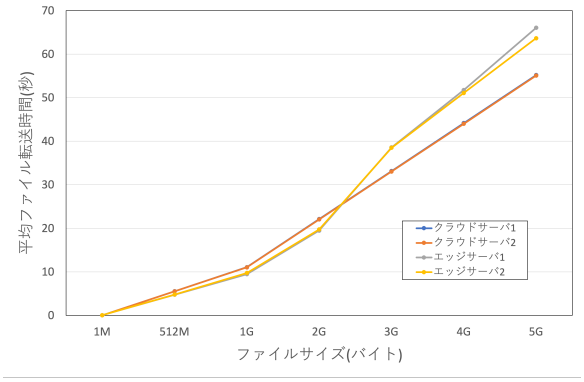

図 3　エッジノードとクラウドノードでの転送時間の比較

ラウドノードからの転送時間がやや短いが，ファイルサイズが 512MB，1GB，2GB の場合はエッジノードからの転送時間の方が短い．しかし，ファイルサイズが 3GB 以上になるとクラウドノードからの方が高速にファイルが転送できていることがわかる．

5　考察

4.4 節に示した実験結果でファイルサイズが 3GB 以上の場合にクラウドノードからのファイル転送時間が短いのは，クラウドノードのメモリ容量やストレージ性能によるものだと考える．クラウドノードでは転送するファイルをストレージから高速に読み出せる上に，一度読み出したファイル内容をメモリにキャッシュすることができる．一方，エッジノードのメモリ量は少ないので，サイズが大きいファイルをキャッシュすることができない．ファイルサイズが 512MB から 2GB の場合にエッジノードの方がファイル転送を短時間で行えるのは，今回利用したエッジノードはクラウドノードよりネットワーク性能の点でやや有利であったからだと考える．

実験結果が示すように，クラウドコンピューティングやエッジコンピューティングにおいて，サービスの処理時間はサービスが動作する計算ノードの処理性能の違いによって異なる．従って，計算ノードの処理性能がそれぞれ異なるような環境で動作するコンテナオーケストレータでは，サービス実行に利用される計算ノードの処理性能を考慮してサービスを選択する必要がある．また，今回の実験ではクラウドノードとエッジノードへの利用者ノードからの通信レイテンシはほぼ同じであったが，実際のエッジコンピューティング環境では利用する計算ノードによって通信レイテンシも異なる．従って，エッジコンピューティング環境で効率的に動作するエッジコンピューティング基盤を開発するためには，計算ノードの処理性能や利用者から計算ノードへの通信レイテンシを考慮してサービスインスタンスを選択する必要がある．

6　おわりに

本研究では，エッジコンピューティングにおける I/O バウンドサービスへのアクセス時間の比較を行った．クラウドノードとエッジノードとでは入出力性能に違いがあるので，サービスが動作する計算ノードが異なる場合のサービスアクセス時間の違いを確認するために，利用者ノードからサービスにアクセスする時間を計測する実験を行った．サービスとして Web サーバを利用して、計算ノードから利用者ノードにファイルを転送する時間を計測したところ，転送するファイルサイズによってクラウドノードとエッジノードのどちらが転送時間が短いかが異なる結果となった．

また実験結果から，エッジコンピューティングにおけるサービスインスタンスの選択手法について考察した．エッジコンピューティングでは，サービスが動作する計算ノードの性能に違いがあるのが普通であり，これを考慮して適切なサービスインスタンスを選択するような仕組みが必要である．実際のエッジコンピューティング環境では利用者ノードからの通信レイテンシが計算ノードによって異なるので，これも考慮に入れてサービス選択を行う必要がある．

今後は，計算ノード間の差異を通信レイテンシも含めて考慮して，利用者ノードによって適した計算ノード上のサービスを選択するようなエッジコンピューティング基盤の実現方法を検討する．エッジコンピューティングで計算ノードとして様々な機器が計算ノードとして使用されているが，実際にどれほど処理性能に違いがあるのか，どの程度の処理なら十分な性能なのかを知ることによって，適切に計算ノードを活用することができると考える．

参考文献

[1] Kubernetes の概要 : `https://kubernetes.io/ja/docs/concepts/overview/`,(2021,2/11)

[2] C. Dupont, et al., “Extend Cloud to Edge with KubeEdge,” in *Proceeding of 2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp.373–377, 2018.

[3] Y. Xiong, et al., “Extend Cloud to Edge with KubeEdge,” in *Proceedings of the Third ACM/IEEE Symposium on Edge Computing*, pp.373–377, 2018.