エッジクラウド環境における eBPF を用いたコンテナスケジューラ

2020SE092 高吉柚伎
指導教員：宮澤元

1　はじめに

近年，インターネットを介して動画や画像を共有するアプリケーションの利用が増加したことにより，大量のデータを素早く通信する需要が高まった．その結果，低遅延な通信を実現するエッジクラウドに注目が集まっている．

エッジクラウドとはクラウドコンピューティングの仮想化技術をエッジコンピューティングに適用した技術である．クラウドコンピューティングではデータセンタのような一地点に置かれた複数の計算ノードで処理をしていたのに対し，エッジクラウドではユーザデバイスのより近くに配置された計算ノードを用いることで通信遅延を最小化し，クラウドの処理をエッジに分散させることができる．

エッジクラウド環境を構築する際，クラウドコンピューティングを利用すると，コンテナ配置を決定する際に通信遅延を考慮しないので，ユーザデバイスから近い場所にある計算ノードに意図的にはコンテナを配置をしない．

本研究の目的は，エッジクラウド環境において，通信遅延を最小限に抑えたコンテナスケジューリングシステムを実現することである．ユーザデバイスと計算ノード間の通信遅延を考慮したコンテナスケジューリングに関する先行研究では，リクエスト先ノード情報を取得するために一度特定のコンテナを経由しなければならないので，サービス応答時間が増加してしまう [1].

本稿では，extended Berkeley Packet Filter(eBPF)[3] を用いたコンテナスケジューリングシステムを提案する．eBPF を用いることでリクエスト先ノード情報をカーネル空間で取得して，コンテナ間通信のオーバーヘッドを削減する．

研究課題は，以下の二つである．

- eBPF を用いた提案システムの実装
- 実装したプロトタイプを用いた実験による提案手法の有用性の検証

2　関連研究

本節では，先行研究および eBPF を用いた関連研究を紹介する．

2.1　ユーザデバイスと計算ノード間の通信遅延を考慮したコンテナスケジューリングシステム

Miyazawa は，ユーザと計算ノード間の通信遅延を考慮したコンテナスケジューリングを提案した [1]．この手法の概要を図 1 に示す．コンテナと共に動作するサイドカープロキシがリクエストを中継することによって，サイドカープロキシはどのノードがリクエストを受信しているのかを把握して，スケジューラに再スケジューリング要請を出す．つまり，リクエストはサイドカープロキシを経由する必要があり，サービス応答時間が増加する．

2.2　eBPF を用いた iptables の改善

cilium は，eBPF を用いたコンテナワークロード間のネットワーク接続を提供，監視するオープンソースソフトウェアである [2]．Kubernetes 環境では iptables は Pod の数に比例して計算量が増えてしまう問題があった．cilium では，独自のハッシュテーブルを用いて計算量を減らすことが出来る．

3　提案手法

エッジクラウド環境においてサービス応答時間を削減するために，eBPF を利用したコンテナスケジューリングシステムを提案する．提案手法の概要を図 2 に示す．リクエスト先のノード情報を，TCP パケットが受信された段階で eBPF を用いてカーネル空間で取得する．そうすることで，リクエストはサイドカープロキシを経由することなく，アプリケーションが稼働するコンテナに直接送られる．eBPF プログラムはリクエストを監視し，必要があればサービスの再スケジューリングを要請する．

4　実験

提案手法の有効性を確かめるために，提案システムを実装して実験を行った．

4.1　実験環境

PC4 台をクラウドノード，PC2 台をそれぞれ別のエッジノードとして Kubernetes クラスタを構成する．クラ

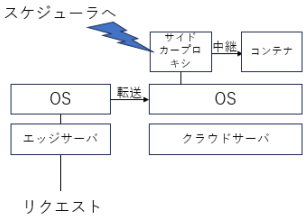

図 1　先行研究でのリクエスト先ノード情報取得

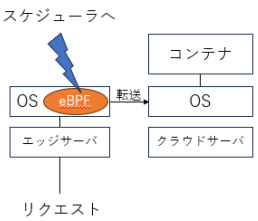

図 2　eBPF を用いたリクエスト先ノード情報取得

スタ上のサービスへのアクセスを行うエンドデバイスには PC1 台を用いる．実験で使う機器のデバイスとソフトウェアの情報を表 1 に示す．エッジクラウド環境をエミュレーションするために，クラウドノードとエッジノード間の通信には Linux の tc コマンドを用いて往復 60msec.±10msec. の通信遅延を設定した．エンドデバイスは，他のノードとの間で同様の通信遅延を設定した状態を初期状態とする (図 3)．

表 1　実験で用いる機器の仕様

	Cloud edge node	End device
CPU	Intel(R)Core(TM) i7-8700K @ 3.70Hz	Intel(R) Core(TM) i7-4770 @ 3.40GHz
Memory	32GiB	16GiB
OS	Ubuntu22.04.2 Kernel 5.15.0	
Kubernetes	v1.26.4	

4.2　実験内容

eBPF を用いて最も近いエッジノードに接続するコンテナスケジューリングが可能であることを確認するため，エンドデバイスからサービスにリクエストを出し続け，サービス応答時間を計測し，提案手法と 2.1 節の Miyazawa の手法の比較を行う．エンドデバイスがネットワーク上を

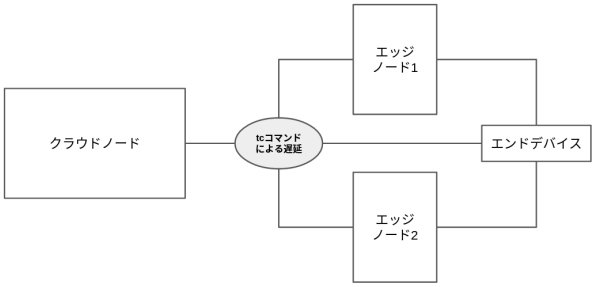

図 3　実験環境の概要

移動していると仮定して，エンドデバイスが計算ノードに HTTP リクエストを 2 秒間隔で 5 分間送信し続ける．エンドデバイスから HTTP リクエストを受信した計算ノードは，1KiB のテキストファイルを送信し返す．またエンドデバイスの通信遅延は，実験開始 60 秒後から 120 秒後までの間はエッジノード 1 との間で設定せず，180 秒後から 240 秒後までの間はエッジノード 2 との間で設定しないようにした．これはエンドデバイスが，遠い場所からエッジノード 1 に接近し，また遠い場所に移動した後，エッジノード 2 に接近して，最後に遠い場所に移動する場合を想定している．

4.3　実験結果

4.2 節の実験をそれぞれ 10 回ずつ行い，平均を取った結果を図 4 に示す．提案手法，先行研究の手法ともに，エンドデバイスが低遅延で通信できる計算ノードを動的に選択して接続していることがわかる．エンドデバイスと全ノード

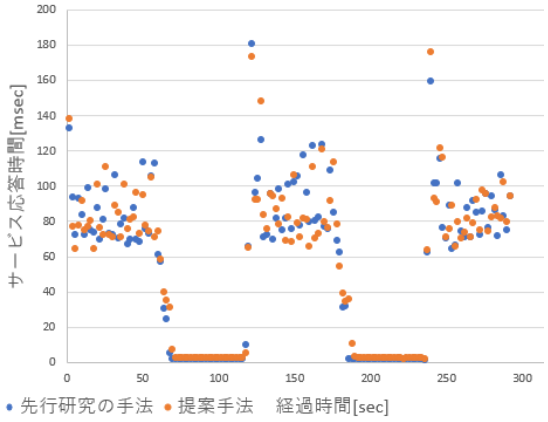

図 4　サービス応答時間 [msec.] の変化

ドとの間で通信遅延が設定されている間，応答時間にばらつきが出たのは，接続先がランダムに切り替わって Pod が再スケジューリングされることがあるからだと考えられる．エンドデバイスとエッジ間の通信遅延設定を解除しても，すぐに応答時間が短くならないのも，Pod の再スケジューリングによるオーバーヘッドが原因である．

表 2　エッジノードの通信応答時間の平均 [us]

接続先ノード	従来手法	提案手法	削減率
エッジノード 1	2466	1980	20%
エッジノード 2	2396	1910	20%

エンドデバイスとエッジノードの間で通信遅延を設定していない間の応答時間の平均を表 2 に示す．実験の結果，提案手法は従来手法と比べて，どちらのエッジノードに接続してもサービス応答時間を約 20% 削減できた．

5　終わりに

本稿ではエッジクラウド環境向けに eBPF を用いたコンテナスケジューリングシステムを提案した．実装した提案システムを用いて実験を行い，エッジノードが利用できる場合には提案手法によってクライアントのサービス応答時間が改善できることを示した．今後の課題としては，エンドデバイス上で通信遅延を測定する際の通信量を最小限に抑える手法の実装および実験，cilium を活用した新たなコンテナスケジューラを提案が挙げられる．

参考文献

[1] H.Miyazawa, “A Latency-aware Container Scheduling in Edge Cloud Computing Environment,” in *Proceedings of the 24th International Conference on Internet Computing and Internet of Things (ICOMP’23)*, 2023.

[2] cilium,`https://cilium.io`,(accessed Sep. 6,2023).

[3] eBPF,`https://https://ebpf.io/`,(accessed Sep. 14,2023).