エッジコンピューティングにおける通信遅延を考慮したコンテナスケジューラ

2018SE079　七野宏紀
指導教員：宮澤元

1　はじめに

IoT の普及に伴いエッジコンピューティングへの注目が高まっている．エッジコンピューティングとはユーザからネットワーク的に近い計算ノードでサービスの処理を行うコンピュータの利用形態である．ユーザからサービスまでのネットワーク距離が近いので，サービスを利用する際の通信遅延を削減できる．また，クラウドにおけるサービスの処理負荷やネットワークトラフィックを低減できる．

エッジコンピューティングを活用してコンテナサービスを行うために，クラウドとエッジ間で双方向通信を行えるように拡張したコンテナオーケストレータが提案されている [1]．しかし，ユーザから見るとクラウドとエッジはリソース量やネットワーク距離の点で非対称であり，ユーザへ効率よくサービスを提供するためにはエッジとユーザの関係性を考慮してスケジューリングを行う必要がある．

本研究の目的はエッジコンピューティングに適したスケジューリングを行うコンテナスケジューラの実現である．ユーザと計算ノード間の通信レイテンシに着目し，従来のコンテナスケジューラに計算ノードとユーザ間の通信時間からコンテナ配置を行う仕組みを追加する．

研究課題は，通信レイテンシを考慮したコンテナ配置を行うコンテナスケジューラの実装と評価である．

2　研究の背景

従来のコンテナオーケストレータではユーザとサーバ間の通信レイテンシを考慮してコンテナ配置を行うことはできない．Ying らが提案したクラウドとエッジ間の双方向通信を可能とする KubeEdge[1] や，中西健登らが提案したネットワーク状況を考慮してコンテナ配置を行うコンテナオーケストレータ [2] のいずれも既に配置済みのコンテナへアクセスを行うユーザとの関係性は考慮されていない．従って, 目的のサービスを異なるエッジサーバへ配置しなおすことにより，ユーザとサーバ間の通信レイテンシを低く抑えることができる可能性がある．

2.1　KubeEdge

KubeEdge とは,Kubernetes[3] 上に構築できるコンテナオーケストレーションをエッジに向けて拡張を行うことのできるオープンソースのソフトウェアである [1]．クラウド側で動作するコンポーネント群とエッジ側で動作するコンポーネント群があり, これらは疎結合の関係である．従ってクラウドとエッジ間のネットワークが切断されても，エッジコンポーネントがコンテナを自律的に管理し再接続された場合にクラウドと同期を行うことができる．KubeEdge はアプリケーションがクラウドとエッジを区別せず動作できる通信基盤を提供することを目的としており，アプリケーションが動作する計算ノードとユーザ間の通信レイテンシを考慮したスケジューリングは行わない．

3　エッジコンピューティングに適したコンテナスケジューラの提案

本研究ではノードから削除されたポッドをユーザとノード間の通信遅延を基にノードへ再配置を行うスケジューラを提案する．提案するコンテナスケジューラの概要図を**図 1** に示す．

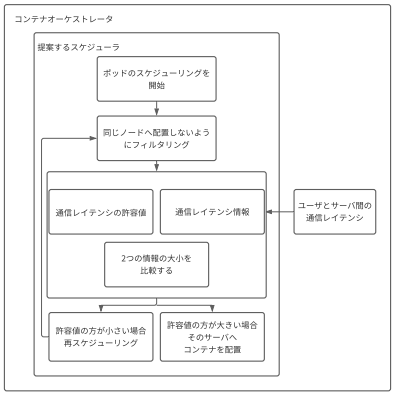

図 1　提案するスケジューラの構成図

スケジューラでは，ポッドの再配置が行われる際に通信レイテンシの値を確認し，設定値を超えた場合に同じノードへ配置を行わないように再スケジューリングを行う．このように，配置時に通信レイテンシを確認し設定値以上であれば再スケジューリングを行うことを繰り返し，全てのエッジとユーザ間の通信レイテンシを測定することなく，目的のノードへとポッドの再配置を行う．

4　コンテナスケジューラの実装

提案したコンテナスケジューラを Kubernetes の Scheduling Framework[4] を用いた Pod スケジューラとして Go 言語で実装した．Scheduling Framework により，Kubernetes 標準の kube-scheduler の変更したい部分だけをプラグインとして用意して，独自のスケジューラを比較的簡単に実装できる．

今回の実装では, 配置先ノードが決定していない Pod に対して, まず Kubernetes の通常のスケジューリング手順に従って配置先の候補ノードを決定する．ここで Scheduling

Framework の PreBind プラグインで，レイテンシマップに登録された候補ノードの通信レイテンシ値を確認する．通信レイテンシ値があらかじめ設定した maxLatency を超えている場合，候補ノードは配置先として不適とみなして配置失敗とし，再度スケジューリングをやり直す．この時，Pod を配置できなかった候補ノードは除外ノードマップに登録する．次回以降のスケジューリングでは，Scheduling Framework の Filter プラグインで除外ノードマップを確認することにより，配置先として不適切なノードを候補ノードとしないようにする．

5　実装したスケジューラの評価実験

1000Base-T ネットワークで接続された master, worker1〜worker8 の計算機 9 台を用いて Kubernetes 環境を構築して実験を行った，各計算機の仕様を**表 1** に示す．

表 1　各計算機の仕様

計算機	master, worker1〜4	worker5〜6	worker7〜8
CPU	Intel Core i7-7700K	Intel Core i7-8700K	NVIDIA Jetson Nano
メモリ量	32GB	32GB	4GB
OS	Ubuntu 20.04.3 LTS	Ubuntu 20.04.3 LTS	Ubuntu 18.04.5 LTS

実装したスケジューラを用いてスケジューリングするよう指定して Docker コンテナを Pod として起動し．Pod が起動時にどのワーカノードに配置されるかを確認する実験 2 種類をそれぞれ 5 回ずつ行った．実験 1 では，Pod 配置時に通信レイテンシの値が大きいノードが除外されているかを確認するために，ワーカノードの通信レイテンシの値を変動させ Pod の配置先ノードを確認する．実験 2 では，Pod 配置時に既存のスケジューラのフィルタリングが正常に動作しているかを確認するために，通信レイテンシの値は実験 1 のままとし，ワーカノードにメモリ負荷を 100MiB 与える Pod を起動させて Pod の配置先ノードを確認する．

実験 1 の結果を**表 2** に示す．5 回の実験のいずれにおいても，想定通りにレイテンシが設定値よりも小さいノードへ Pod が配置されることが確認できる．通信レイテンシを基に再スケジューリングを行う回数が増えるほど再配置にかかる時間が大きくなることが確認できた．

表 2　通信レイテンシの値を変動させた場合の実験結果

	計算ノードのレイテンシ (ms)								配置ノード
	worker1	worker2	worker3	worker4	worker5	worker6	worker7	worker8	
1 回目	963	605	718	838	131	468	414	770	worker6
2 回目	788	108	406	784	389	530	780	143	worker5
3 回目	361	703	146	886	668	209	495	705	worker6
4 回目	89	517	442	927	601	615	423	623	worker1
5 回目	862	186	954	619	977	900	739	228	worker2

実験 2 の結果を**表 3** に示す．1，3 回目では元々配置されたノード以外の通信レイテンシが小さいノードへ配置されることが確認できる．4，5 回目の実験では配置先ノードは変化しなかった．5 回目で配置されるノードの候補は worker2，8 であるが，worker8 はリソース量が worker2 と比べて少ないので，worker2 に配置されたと考えられる．4 回目の実験では，候補ノードは worker1，3，7 であるが，worker7 には 5 回目と同様の理由で配置されないと考える．worker1，3 については，Pod を起動した状態で kubectl describe を使って CPU とメモリの使用率を確認したが，使用率に大きな違いは無かった．もう少し大きな負荷がなければ Pod の配置先は変更されないと考える．

表 3　ノードのリソース量を変動させた場合の実験結果

	リソース量変化前の配置ノード	変化後の配置ノード
1 回目	worker6	worker5
2 回目	worker5	worker2
3 回目	worker6	worker1
4 回目	worker1	worker1
5 回目	worker2	worker2

以上 2 つの実験から実装したコンテナスケジューラが通信レイテンシを考慮しつつ，ノード内のリソース量も考慮してコンテナ配置を行うことを確認できた．

6　おわりに

本研究では，エッジコンピューティングに適したコンテナオーケストレータを提案した．ユーザとエッジサーバ間の通信レイテンシに着目し，Kubernetes を拡張しあらかじめ設定した通信レイテンシの値よりも低い通信レイテンシ値を持つノードへ Pod を配置するスケジューラを実装した．実験結果から通信レイテンシの低くかつリソース量も充分であるノードへと Pod が配置されることを確認できた．しかし，Pod を配置するのに時間が大きくかかってしまうケースが確認された．

今後の課題として今回確認できたコンテナ再配置時にかかる実行時間について，スケジューラアルゴリズムの改善が必要である．また，ユーザとエッジサーバ間の通信レイテンシを測定することができるコンテナオーケストレータの実装と，既に配置済みのポッドを測定したレイテンシに基づいてノードから削除する新たなデスケジューラパラメータの実装も行う必要がある．

参考文献

[1] Ying Xiong, et al. , “Extend Cloud to Edge with KubeEdge” in Proceedings of the Third ACM/IEEE Symposium on Edge Computing, pp.373377，2018.

[2] 中西 健登 他: “ネットワーク状況を考慮したコンテナ型コンテンツ配信基盤の検討”，情報処理学会研究報告，Vol.2019-IOT-44 No.23，2019.3.7

[3] Kubernetes, `https://kubernetes.io/ja/docs/home/`.

[4] Kubernetes:Scheduling Framework, `https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/`.