5U-02

スマートバイクのための Cloud Native architecture の検討

瀧崎 尚[1]　寺西 裕一[2,1]　義久 智樹[1]　川上 朋也[3]　松本 智[1]　下條 真司[1]
大阪大学[1]　情報通信機構[2]　福井大学[3]

1. はじめに

近年，様々な IoT デバイスの開発が盛んである．その一つに，センサやエッジデバイスを備えたネットワーク接続可能な自転車（スマートバイク）がある．スマートバイクにおいて，自転車に取り付けられたセンサから集められたデータを元に，走行経路を推薦したり，運転者に注意を促したりといったアプリケーションが考えられる．これらのアプリケーションは携帯端末や車載端末などエッジ環境と機械学習や推薦といったクラウド環境で行われる処理が連携して行われる．しかしながら，自転車に取り付けられたエッジデバイスのリソースには，メモリ・電源・通信環境などに制約があり，必ずしも安定した処理が行われるとは限らない．

アプリケーションを継続的に開発するアプローチの一つとして，アプリケーションをいくつかのコンテナで表現されるマイクロサービスの連携として実現していく，Cloud Native なアプローチが考えられ，CNCF(Cloud Native Computing Foundation)[1]を中心に様々なツールの開発が進んでいる．

本研究では，この Cloud Native なアプローチを上記のスマートバイクのアプリケーション構築に適用することで IoT 環境に適応できるアジャイルなアプリケーション開発を目指す．ここでは，そのプロトタイプ実装を行い，エッジ・クラウド連携を行う機構の提案を行い，評価を行う．

2. 前提

本研究では，図 1 のように，自転車にはエッジ端末として Raspberry Pi，カメラセンサ，モバイルバッテリー，Google Pixel を具備させ，モバイルバッテリーから電力供給を，Google Pixel のテザリングにより通信を行う．
カメラからのセンシングデータを処理するアプリケーションは，Kubernetes で管理されるコンテナで実現されるマイクロサービスの連携で構成される．また，エッジにおけるコンテナ環境を実現するために，Kubeedge を利用する．Kubeedge は，エッジ・クラウドの通信が切れ再接続されたときに，デプロイメントのメタデータを自動的に同期させる機能を持つ．

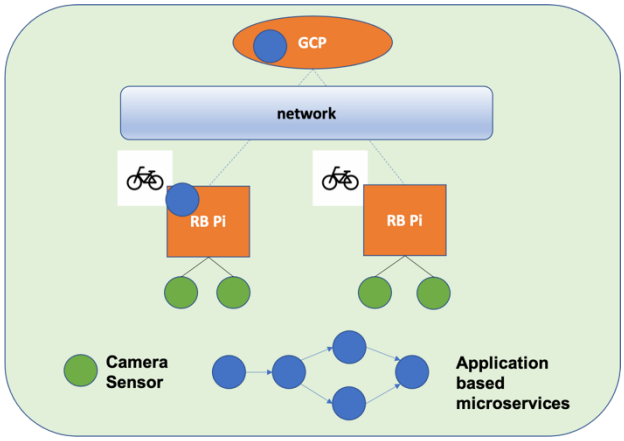

図 1　想定環境

3. 提案

本研究では，図 2 のように自転車に備えられた Raspberry Pi をエッジ，GCP（Google Cloud Platform）をクラウドとして，Kubernetes と KubeEdge[2]を用いたエッジ・クラウドのクラスタを構成し，通信環境や場所，バッテリーの残量に応じて pod の構成を変更する機構を導入する．そのために、Monitor module，Control module を配置する．またアプリケーションのマイクロサービスは Kubernetes 上で Pod として表現され，各ノードに配置して処理される．マイクロサービスの処理データは各ノードで蓄積することを前提とする．エッジリソースに制約があるスマートバイク環境において，様々な環境条件の変化をモニターモジュールが観測し，コントロールモジュールにより，pod の構成，すなわち，Deployment を変更する．エッジデバイスのモニタリングとして，各ノードに NodeExporter を設置し，Kubernetes のクラスタ上に Prometheus をデプロイすることで，それらのメトリクスを収集する．Prometheus のローカルストレージはディスクやノード障害に対して耐性がないため，一時的なデータとして扱う．また，クラウド上では

Examination of Cloud Native architecture for Smart bike
1 Osaka University , Japan
2 NICT , Japan
3 Fukui University , Japan

Prometheus の永続的なリモートストレージとして，時系列データベースである InfluxDB を採用する．

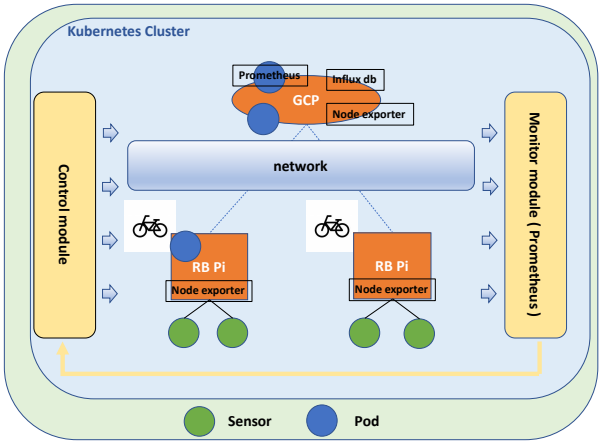

図 2　提案環境

コントロールモジュールでは，アプリケーションの Pod をあらかじめエッジ・クラウド両方に配置しておき，ノードの負荷状況・性能などのモニター結果に従って，使用する Pod を選択する．Pod にはこの負荷状況によって変化するラベルが付与されており，ラベルが一致した Service と紐付けされる．図 3 のように Pod に対して Service を構成し，その Service のラベルを変更することでどちらの Pod を使用するかをコントロールする．Kubernetes のクラスタ上に上記のようなコントロールのための Pod をデプロイする．

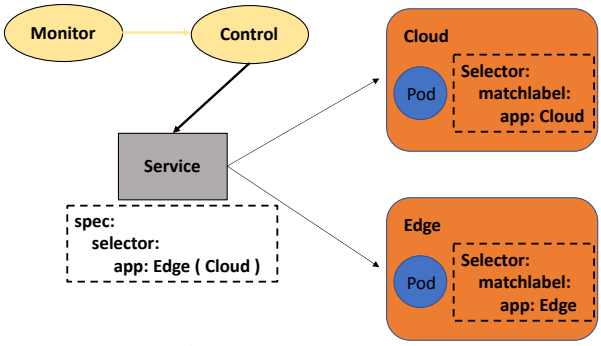

図 3　コントロール

4.　評価

図 3 の手法で Pod の配置をコントロールできているかの確認を行う．評価のために，エッジに具備されたカメラで写真を撮り，エッジまたはクラウドに送り処理し，その結果をエッジに返す，というアプリケーションを用意する．本評価では，アプリケーション Pod をあらかじめエッジまたはクラウドに配置しておき，2 時間おきに使用する Pod を切り替えた場合の，エッジのリソースを測定し，性能を評価した．図 2 の環境を想定し，評価環境としてクラウドを GCP，エッジを LAN 内にある Raspberry Pi で構築した．

図 4 は，エッジの CPU 使用率を測定したグラフを示す．2 時間おきに周期的なグラフの変化の様子が確認できる．

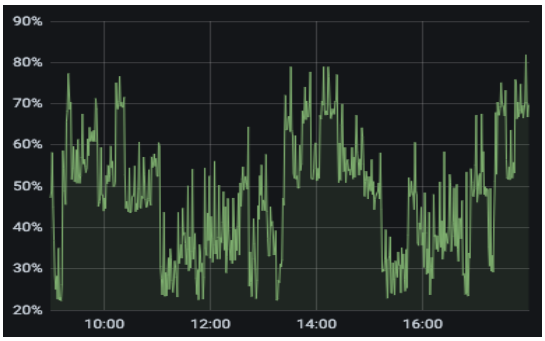

図 4 CPU 使用率

図 5 に，エッジ・クラウドのネットワークトラフィックを測定したグラフを示す．処理 Pod をクラウドに配置した場合，エッジからクラウドに画像を送信するため，ネットワークトラフィックは大きくなる．図 5 からもその様子が確認できる．

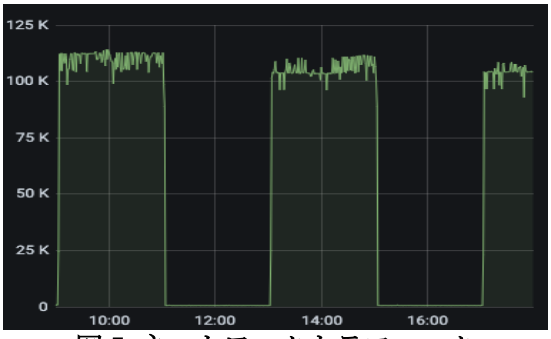

図 5 ネットワークトラフィック

5.　終わりに

本研究では，スマートバイクにおけるエッジデバイスの状態に対応できるアーキテクチャを提案した．本論文の評価を通じて，エッジデバイスのリソースをモニタリングし，アプリケーションの配置をコントロールすることができることを確認した．

謝辞

本研究の一部は科学研究費補助金 JP20H00584 の助成による成果である．

ここに記して謝意を表す．

参考文献

[1] https://www.cncf.io/
[2] https://kubeedge.io/en/