4K-05

マイクロサービスモニタリングのための構成情報解析ツールの設計と実装

天野 治憲† 廣津 登志夫‡
法政大学情報科学部†

1 序論

近年、サービスの大規模化と需要の変化に柔軟かつ迅速に対応するために、マイクロサービスアーキテクチャと呼ばれるサービスの各機能を独立したコンポーネントに分割する手法が利用されている。これにより、各機能を異なる開発基盤で実装したり、短期間でサービスの一部分に変更を加えることができる。このようなサービスにはコンテナと呼ばれる仮想化技術が利用されることが多い。コンテナはリソース効率と可搬性の高さから、分散されたサーバーを用いた環境下でも、マイクロサービスを効率的に運用することができる。

マイクロサービスアーキテクチャを採用するサービスでは、各マイクロサービスとそれらを繋ぐサービス基盤が複合的にサービス全体の品質に影響を与える。しかし、コンテナはオーケストレーションツールの自己修復機能によってクラスタ内を頻繁に移動する。それに伴って、サービス基盤のメトリクス収集に必要なコンテナが接続する仮想ネットワークインターフェイス（vNIC）の情報も変化する。さらに、コンテナは独立した名前空間を利用しているため、コンテナが利用する vNIC を特定するためには、コンテナの名前空間とホストの名前空間の双方から集めた情報を照合する必要があり、サービス管理者がこれらの処理を行うには手間が大きい。

そこで、本研究では eBPF によるメトリクス収集を想定し、コンテナのプロセス ID から自動でコンテナ内の情報とホストの情報を収集し、コンテナが接続する vNIC の名前を特定する機能を実現する。

2 関連研究

先行研究[1]では Kubernetes で運用されるコンテナを利用するマイクロサービスを想定した、リアルタイムモニタリングシステムを提案している。当研究では、モニタリングに必要な情報をコンテナの移動に合わせて自動で収集し、その情報を基に eBPF プログラムを自動生成・アタッチして継続的なメトリクスの収集を可能にしている。コンテナが接続する vNIC の特定には、まず Docker から対象となるコンテナのプロセス ID を取得し、それを使ってコンテナの名前空間に接続して情報を収集する。次に、ホストマシンのコマンドラインインターフェイス(CLI)を利用してホスト OS の名前空間で情報を収集し、コンテナの名前空間で収集した情報と照合して vNIC のインターフェイス名を特定している。

既存の Linux の CLI でこれらの処理を行うためには、nsenter コマンドと ip コマンドの利用が考えられる。しかし、これらのコマンドから得られる情報にはモニタリングに必要のない情報も多く含まれており、不要なオーバーヘッドが発生してしまう。マイクロサービスアーキテクチャを利用したサービスを運用するサーバーでは膨大な数のコンテナが展開されるため、このようなオーバーヘッドはさらに大きくなり、継続的なモニタリングに影響を及ぼすと考えられる。

そこで、本研究では vNIC のインターフェイス名の収集に限定したコマンドを設計・実装し、オーバーヘッドの削減を実現する。

3 設計

コンテナ内部から見えるネットワークインタフェースである vNIC(例えば図 1 3:eth0@if4) には、それと対になるホスト OS レベルの vNIC(4:veth12abc) があり、これはホスト OS の名前空間にあるためにコンテナからは知ることができない。しかし、eBPF によるインフラストラクチャのモニタリングの際には、このホスト OS の名前空間にある vNIC のインターフェイス名を指定する必要がある。

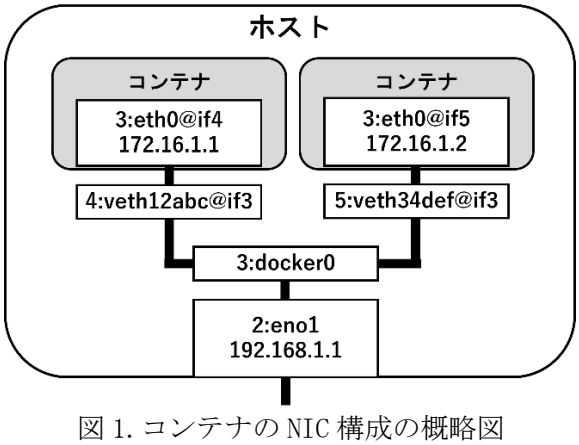

図 1. コンテナの NIC 構成の概略図

Design and Implementation of Configuration Information Analysis Tool for Monitoring Microservices
†Harunori Amano, Hosei University
‡Toshio Hirotsu, Hosei University

その一方で、IP アドレス等の情報が割り当てられるのは、コンテナの名前空間にある vNIC(eth0)のみであり、ホスト OS の名前空間から取得できる情報のみで任意の vNIC がどのコンテナに接続しているかを特定することはできない。そこで、コンテナの名前空間にある vNIC に付与されたペアとなるホスト OS の vNIC のインターフェイス番号(@if)を利用してコンテナに接続している vNIC を特定する。

まず、ホスト OS の名前空間で NIC のインターフェイス名およびインターフェイス番号の一覧を取得する。次に、引数として与えられたプロセス ID を使ってコンテナの名前空間を特定し、接続する。さらに、コンテナの名前空間でコンテナの持つ vNIC の一覧と、対応する vNIC のインターフェイス番号を取得する。最後に、ホスト OS の名前空間から取得した一覧情報からインターフェイス番号が一致する vNIC を特定し、そのインターフェイス名を標準出力に出力する。

4 実装

前節に述べた機能を実現するコマンドラインツール「psnic」を、C 言語を用いて実装した。psnic では、引数に与えられたプロセス ID を使って/proc/”プロセス ID”/ns/net をマウントして名前空間を切り替えた後に、execvp 関数を使って再び psnic コマンドを実行する。execvp 関数で psnic コマンドを呼び出す際には、引数に特定のキー値を与えることで、呼び出し先の psnic コマンドが、どちらの名前空間で呼び出されたものなのかを判定し、処理を実行する。execvp 関数は、引数に与えられたプログラムを別プロセスとして実行する。execvp 関数を使用すると、その段階で元のプロセスの実行を終了するため、新しいプロセスの実行結果を、元のプロセスで利用することができない。そのため、初めにホスト OS の名前空間にある NIC のインターフェイス番号とインターフェイス名の収集を行い、その結果をコンテナの名前空間で実行される新しいプロセスに引数として渡す。

NIC の情報収集は、カーネルにあるルーティングテーブルをダンプして得た情報から、必要となるものだけを抽出して行う。新しいプロセス上ではペアとなるホスト OS 上の vNIC のインターフェイス番号の情報を収集し、引数として渡されたホスト OS で収集した情報との照合処理を行い、一致した vNIC のインターフェイス名を標準出力に出力する。

5 評価

同一コンテナのプロセス ID を使って psnic コマンドを利用した際の実行時間と、nsenter コマンドと ip コマンドを利用した際の合計の実行時間の比較を行った。100 回測定したそれぞれの実行時間の平均値を以下のグラフに示す。nsenter コマンドと ip コマンドの合計実行時間の平均値は 1.728 ミリ秒であり、その実行結果からさらにインターフェイス名の特定処理を行うことを考慮する と、オーバーヘッドはさらに大きくなると考え られる。これに対して、psnic コマンドの実行時間の平均値は約半分の 0.979 ミリ秒であった。この結果から、既存のコマンドを利用した場合に比べてオーバーヘッドを増加させることなく、インターフェイス名の特定処理の手間が削減できることがわかる。

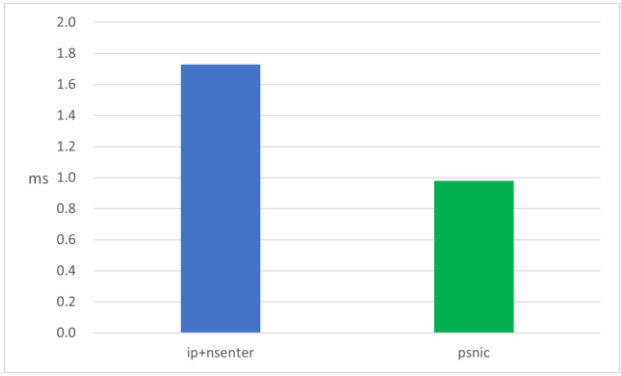

図 2. 既存のコマンドと提案手法の平均実行時間

6 結論

マイクロサービスのモニタリングのためのインターフェイス名特定ツール「psnic」を開発した。実行時間の評価実験では、既存のツールを使用した場合と比較して、オーバーヘッドが約半分になることを確認した。これにより、オーバーヘッドを増加させることなくコンテナの vNIC 名の特定に必要な作業を簡略化し、マイクロサービスのモニタリングを容易にすることができる。

謝辞

本研究は JST 科研費 JP20K11754 の助成をうけたものである。

参考文献

[1] T. Shiraishi, M. Noro, R. Kondo, Y. Takano and N. Oguchi, "Real-time Monitoring System for Container Networks in the Era of Microservices," 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), 2020, pp. 161-166