柔軟で動的な実行環境を提供するための コンテナ型仮想化基盤アーキテクチャの検討

2110525　平地浩一　　矢崎研究室

1　はじめに

研究や開発でアプリケーションの実行環境が必要な際，大人数に環境を提供するために 1 つのコンピュータを共用することが多い．しかし，共用マシンでは管理者によるリソースやアプリケーションの制限がある．ユーザ専用の環境を提供することで，管理者の制約なく自由に環境を利用できるようになる．一般的には物理マシンや仮想マシンを割り当てるが，リソースが有限であり，大人数に割り当て続けるのは現実的ではない．これを解決するため，リアルタイムに最適なリソース配分を行う必要がある．仮想化の粒度に応じた様々な仮想化技術があるが，その中でもコンテナ仮想化技術に注目した．コンテナは OS カーネル上で実行環境を分離し，物理マシンの仮想化よりも細かく制御できる利点がある．

本研究では，利用者が自分自身で様々なコンピューティング環境をオンデマンドに使用できるコンテナ型仮想化基盤を提案・構築する．

2　提案するシステムの構成

本研究で提案するシステムの構成を図 1 に示す．本研究では主に，図 1 中における Controller(コントローラ) に該当する機能を実装する．

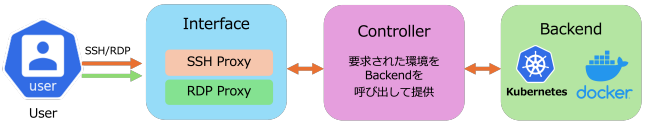

図 1　提案システムの構成

システムでは，ユーザーに対して実行環境毎に異なるエンドポイントが提供されている．ユーザ ssh や RDP といったプロトコルによってシステムの Interface へ接続すると，Controller へユーザが要求する環境が伝達される．Controller では，Kubernetes や docker といったバックエンドを API 経由で呼び出し，要求されたコンピューティング環境を動的に提供する仕組みである．

3　関連研究

3.1　ContainerSSH

ContainerSSH は，ユーザからの SSH 接続に対して Kubernetes や Docker などのバックエンドを介して動的にコンテナを立ち上げる仕組みを提供する OSS である [1]．ユーザは管理者が予めコンテナイメージを選択することはできないという課題がある．

3.2　sshr

sshr は，鶴田氏らによって開発された SSH Proxy サーバである [2]．sshr では，大規模なサーバ群においてユーザが SSH 接続に用いたユーザ ID に応じて，予め紐づけられたサーバへの接続を中継する．ユーザ自身が接続するサーバを選択することはできないという課題が残されている．

4　既存システムのレイテンシ測定実験

実装するにあたり，ContainerSSH の実装を参考とする．提案するシステムに要求される時間的な制約を確認するため，ContainerSSH を用いて複数人が同時にアクセスした際のレイテンシを測定した．

実験では，backend に Docker を用いた．検証環境において ContainerSSH と Docker を実行し， ssh コマンドで同時接続を行なった際の接続時間を計測した．

4.1　クライアント側から SSH 接続時のターンアラウンドタイムの計測

まず初めに複数のユーザが同時に接続した際に，ユーザが SSH 接続をしてからコマンドを実行して終了するまでのターンアラウンドタイムの変化を計測した．スクリプトを用いて同時に接続する数を 1 から 10 づつ 100 まで増やしていき，接続数ごとの処理時間を記録した．結果を図 2 に示す．

グラフの横軸は同時接続数を表している．縦軸は SSH コマンドを実行してから接続が成功し，date コマンドを実行したのち，ssh 接続を切断するまでの時間を，クライアント側から time コマンドで測定した時間を秒で表している．

4.2　サーバ側における各部分処理にかかる時間の測定

前の実験より，ユーザが SSH コマンドを実行してからの応答時間は同時接続数に応じて長くなることがわかった．ここでは，どの処理によって接続時間が増加するのか

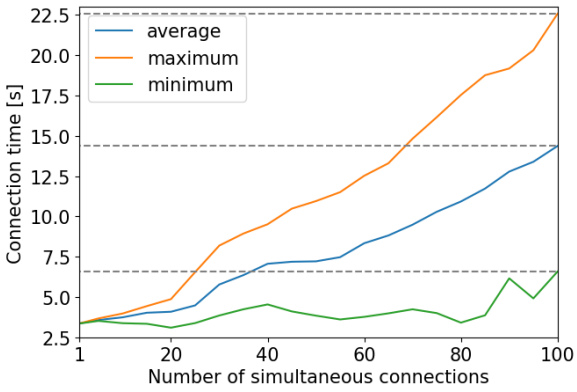

図 2　並列実行数と接続時間の関係

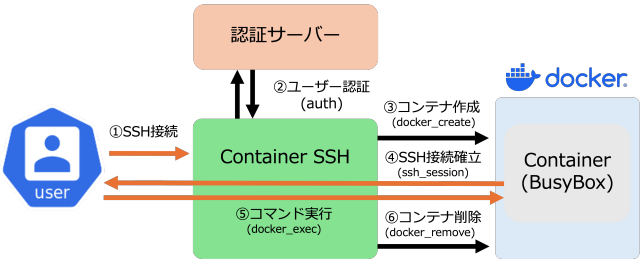

図 3　ContainerSSH における内部処理

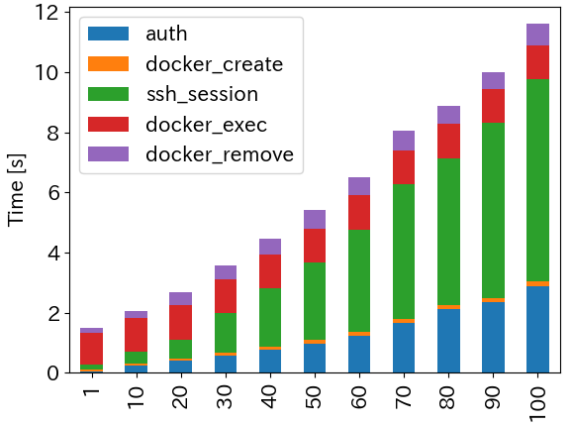

図 4　並列実行数と主な処理時間の関係

を検証した．

実験では，並列に接続するクライアント数を 1 から 100 まで変化させ，それぞれの場合で各処理にかかる平均の時間を算出した．結果を図 4 に示す． グラフの横軸は同時接続数を表している．縦軸は SSH コマンドを実行してから接続が成功し，date コマンドを実行したのち，ssh 接続を切断するまでの時間を，クライアント側から time コマンドで測定した時間を秒で表している．数値は上記のように測定した時間について，全接続のうち ssh の接続にかかった最長の時間を maximum，最短の時間を minimum，接続数での平均を average で示している．

5　考察

図 2 より，ContainerSSH では同時接続数が多くなるほど 1 つの接続にかかる時間が増加していくことが確認できた．同時接続数が 100 程度の場合，最長で 22 秒ほどの時間がかかっており，ユーザがストレスなく現実的に利用できるとは言えない．平均の接続時間に関しても 100 接続であっても約 14 秒程度かかっている．より接続数が増えた場合，これらの時間はさらに増加していくと考える．提案システムが想定している 200〜300 名程度の同時利用を考えると，ユーザ数に応じて顕著に処理時間が増加する部分においては対策が必要である．

図 4 を参照すると，ContainerSSH の内部処理のうち，認証 (auth) と SSH セッション管理 (ssh_session) の各処理は同時接続数に応じて特に増加していくことが分かった．認証と SSH セッション管理に関しては，ユーザの待ち時間への影響が大きいため，提案手法ではこの時間を短縮する実装上の工夫が必要であると考える．コンテナ削除処理も接続数に対して若干増加しているが，コンテナ削除処理はユーザが環境を利用し終わった後に行われるため接続時間に大きな影響はないと考える．

認証部分に関しては，同時接続数が増えるほど認証サーバの負荷が増えて応答が遅くなっていると考える．現在の認証サーバは ContainerSSH が提供する簡易的なものを利用している．今後 OpenID Connect や LDAP 等のより高度な認証機能を追加していくとより大きなボトルネックになると予想する．そのため，1 度認証したクライアントに対して一時的なトークンを払い出し，以降は一定時間認証サーバにアクセスする必要をなくといったような仕組み等を検討している．

SSH セッション管理部分に関しては，接続数ごとに SSH の中継をするプロセスが増えていき負荷になっていると想定する．このため，負荷分散をするための仕組みが必要になると考える．

6　まとめ

本論文では，利用者の様々な要求に対して柔軟なコンピューティング環境を動的に提供するためのコンテナ型仮想化環境を提案し，これを実現するためのアーキテクチャについて，既存実装を踏まえて検討および検証した．

今後，特に時間を要する認証と SSH の Proxy 部分に関して認証サーバへの負荷が少ない認証の仕組み作りや，SSH の Proxy を負荷分散するなどの対策を行なっていく予定である．

参考文献

[1] Cloud Native Foundation. ContainerSSH. https://

containerssh.io/.

[2] Hirofumi Tsuruta and Ryosuke Matsumoto. Sshr: An ssh proxy server responsive to system changes without forcing clients to change. *Proceedings - 2020 IEEE 44th Annual Computers, Software, and Applications Conference, COMPSAC 2020*, pages 1761–1766, 7 2020.