ページキャッシュのプリフェッチによるコンテナ移送時の性能低下軽減

上司陽平† 青田直大†
古藤明音† 河野健二†

1. はじめに

近年 docker, や Linux Containers(LXC) などのコンテナ仮想化技術が普及してきている．それに伴い，従来ハードウェア仮想化による仮想マシン (VM) をクラウドで提供してきたベンダもコンテナをクラウドで提供するサービスを始めている．ベンダはクラウドサービスを運用する上で，これらの仮想化技術に高い管理性を求める．そのためハードウェア仮想化技術では VM が起動している状態をスナップショットとして保存したり，VM を再起動することなく物理マシン間を移送する VM 移送技術がある．コンテナ仮想化でもハードウェア仮想化のような管理性を実現するために，コンテナを再起動することなく移送する技術などが開発されている．[1)] しかし既存の手法では，コンテナを動かすための必要最低限の情報しか移送しない．そのため移送元のページキャッシュは移送されず，移送先でコンテナの性能低下につながる可能性がある．そこで本研究では，コンテナ移送時にページキャッシュを移送先でプリフェッチすることで，性能低下を軽減する手法を提案する．ファイルを読み続けるプロセスの移送では，本提案により既存手法の 2.2 倍の量のデータを読み込むことができた．

2. コンテナ移送技術

コンテナ移送技術を用いてベンダは物理マシンで稼働しているサービスを再起動することなく他の物理マシンに移送し，物理マシンのメンテナンスを行うことができる．他にも移送技術によりコンテナを必要最低限の物理マシン上に集約させて，他のマシンをスリープにさせることで省電力化も可能となる．

コンテナ移送はプロセスのチェックポイントとリストア (C/R) によって実現されている．C/R とはプロセスのスナップショットを作り（チェックポイント），後にそのスナップショットから作成時の状態のプロセスを復元（リストア）する技術である．チェックポイント時にはプロセスの使用するユーザメモリとカーネルがプロセスを動かすために必要な全ての情報をスナップショットとして保存する．

コンテナ移送は C/R を用いて，まず移送元でコンテナをチェックポイントし，スナップショットを移送先に送信する．そして移送先でスナップショットからコンテナを復元することで実現される．この時コンテナのファイルシステムはネットワークストレージに保存することで，各ホスト間で共有されており，移送する必要はない．

† 慶應義塾大学
Keio University

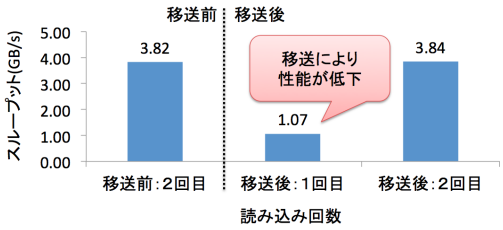

図 1 1 GB ファイルの読み込みスループット

3. 問題点

既存の C/R では OS 上のページキャッシュはスナップショットとして保存されないため，コンテナは移送先でリストア後にページキャッシュを使用することができない．そのため，移送先でページキャッシュがないことによるコンテナの性能低下が起こる可能性がある．実際に性能低下が起こる場合を示した実験結果を図 1 に示す．実験では移送前後で 1GB のファイルをシーケンシャルリードするスループットを測定した．移送後の 1 回目のリードでは，ページキャッシュがないことによりスループットが低下している．移送後 2 回目

ではページキャッシュが作られ，移送前２回目と同等のスループットがだせている．この事前実験では移送先にページキャッシュが送られないことにより，ディスク読み込みが頻繁に起こり性能が低下がする場合を示した．ページキャッシュを送らないことによる性能低下は VM 移送技術でページキャッシュを送らない提案をした研究でも提起されている．[2)]

4. 提　　案

本研究ではコンテナ移送後のページキャッシュがないことによる性能低下を軽減することを目的とする．そこで移送元で使われていたページキャッシュをホスト間で共有しているネットワークストレージから移送先でプリフェッチすることを提案する．これにより，移送後のコンテナのキャッシュミスを削減する．この時あまり使われていないページキャッシュもプリフェッチするようにすると，移送先マシンの負荷が大きくなってしまうので，よく使われているページキャッシュ（*hotcache*）のみを移送先でプリフェッチする．

プリフェッチは移送元でコンテナの使用しているファイルパスと，実際に *hotcache* としてのっているページのファイル内での位置（*page index*）を利用して行う．ファイルパスと *page index* を移送先に送信し，ネットワークストレージからページをプリフェッチする．

5. 実　　装

実装には kernel に Linux 4.4，C/R に CRIU[1)] を用いた．*hotcache* の *page index* を読み込むためにプロセスがオープンしているファイル毎にそのファイルの *hotcache* を取得できるモジュールを実装した．*hotcache* かどうかの判別には Linux が *hotcache* を管理するためにページ毎に持つ active と inactive という状態を用いた．プリフェッチ処理は移送時に CRIU がプロセスを止めた時点から並行して行う．移送元ではプロセス停止後，プロセスのオープンしているファイルを fsync() により同期する．そしてファイルパスと *hotcache* の *page index* を読み込み，移送先に送信する．移送先ではそれらの情報を用いて pread() でページをプリフェッチする．

プロセス移送は python により実装した．

6. 実　　験

移送先，移送元ホストには Intel Xeon CPU X5650, 16GB Memory , Gigbit Ethernet を用いた．それぞれのマシンには移送の専用線として物理的に直接ケーブルを繋げてあり，移送通信のみそのケーブルを使用して通信を行う．

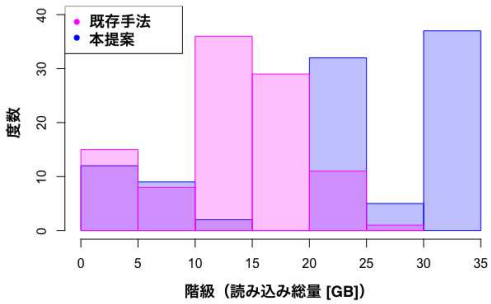

図 2　1 GB ファイルを繰り返し読み込み続けた時の読み込み総量ヒストグラム

実験では NFS 上の 1GB ファイルを読み続けるプロセスを移送し，読み込むことができたデータ総量を測定した．プロセスは開始から一定の時間動いた後に停止する．実験結果を図 2 に示す．X 軸は読み込んだ量 [GB] の階級，Y 軸に度数を示す．赤色が既存手法，青色が本提案のヒストグラムで，重ねて示している．各平均値，中央値は既存手法が 13.3GB, 11.0GB で本提案手法が 23.7GB, 25.0GB となった．本提案は中央値で 2.2 倍多くのデータを読み込むことができた．

7. おわりに

近年コンテナ仮想化技術が普及してきている．それに伴いベンダの管理性を向上させる技術が開発されている．そのひとつであるコンテナ移送では，ページキャッシュを移送しないことによる移送後のコンテナの性能低下が起こる可能性がある．そこで本研究では，コンテナ移送時に移送元で使っていた *hotcache* を移送先でプリフェッチすることで性能低下を軽減する手法を提案した．ファイルを読み続けるプロセスの移送では，本提案により既存手法の 2.2 倍の量のデータを読み込むことができた．

参　考　文　献

1) CRIU: Checkpoint/Restore In Userspace. `https://criu.org/Main_Page`.

2) Koto, A., Kono, K. and Yamada, H.: A Guideline for Selecting Live Migration Policies and Implementations in Clouds, *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pp. 226–233 (2014).