異種 OS コンテナ・マイグレーション実現に向けた Linux-FreeBSD 間プロセス・マイグレーション実現方式の検討

高 川 雄 平 † 松 原 克 弥 †

1. はじめに

クラウドコンピューティング分野において，コンテナ型仮想化が注目されている[1)]．コンテナ型仮想化は，OS が提供する資源を分離・制限し，単一動作する OS 上に複数の独立した実行環境を構築できる軽量な仮想化を実現する．また，クラウドコンピューティング実働環境では，負荷分散や可用性の実現を目的としてライブマイグレーションが活用されている．ライブマイグレーションとは，サービスを実行している仮想マシンを動的に別のマシンに移動させる技術である．コンテナ型仮想化のライブマイグレーション (以下，コンテナ・マイグレーション) は，CRIU(Checkpoint/Restore in Userspace)[2)] や FreeBSD VPS(Virtual Private System)[3)] により実現されている．しかし，コンテナ型仮想化の実装が OS に依存するため，異種 OS 間におけるコンテナ・マイグレーションは実現されていない．

コンテナ・マイグレーションを行うには，プロセス実行状態のマイグレーション (以下，プロセス・マイグレーション) とプロセス隔離状態のマイグレーションが必要になる．本稿では，異種 OS コンテナ・マイグレーション実現に向けた Linux と FreeBSD 間のプロセス・マイグレーションの実現方式を検討する．

2. 各 OS における ABI の差異

一般的に，OS が異なると，準拠する ABI(Application Binary Interface) が異なる．ABI とは，アプリケーションと OS 間のインタフェース仕様のことで，システムコールやデータ型，データ配置などを指す．実際，Linux と FreeBSD では，システムコールとメモリレイアウトに差異がある．

2.1 システムコール

UNIX 系 OS である Linux と FreeBSD では，多くのシステムコールで提供する機能が類似している．しかし，同じ機能のシステムコールにおいても，システムコール番号や引数パラメータ，引数の渡し方が異なる (表 1)．そのため，スタックの状態やレジスタの状態が異なり，結果として，Linux バイナリと FreeBSD バイナリはそれぞれの OS 上でしか動作しない．

表 1 open システムコールにおける OS の差異

	Linux	FreeBSD
open() のシステムコール番号	2	5
open() の引数オプション O_CREAT	0x0200	0x0040
引数の渡し方	レジスタ経由	スタック経由

2.2 メモリレイアウト

メモリレイアウトは，仮想メモリ空間におけるデータ領域やスタック領域などの配置である．FreeBSD 以外の多くの OS では，ASLR(Address Space Layout Randomization) によって，テキスト，データ，ヒープ，スタックの各領域がランダムに配置される．また，ASLR を無効化して領域の配置を固定した場合でも，Linux と FreeBSD では，ヒープ領域，スタック領域，および，共有ライブラリの配置に差異がある．この差異を補正するためには，スタック内にあるベースアドレス，リターンアドレス，共有ライブラリ内の命令を示すアドレス，ヒープ領域の変数を参照するアドレスを全て特定して値を書き換える必要が生じる．

3. プロセス・マイグレーションの実装

プロセス・マイグレーションは，対象プロセスの実行状態を示すレジスタやメモリの状態を保存 (スナップショット) し，別の OS 環境で生成したプロセスに状態を復元 (レストア) することにより実現できる．

3.1 スナップショット

対象プロセスに関するレジスタとメモリの情報の取得方法について述べる．

レジスタ情報は，ptrace システムコールによるプロセスへのアタッチ時に，GETREGS コマンドを用いて，レジスタ情報を格納する構造体 (以下，レジスタ構造体) から取得する．レジスタ構造体には，汎用レ

† 公立はこだて未来大学

ジスタ，インデックスレジスタ，セグメントレジスタ，フラグレジスタの各値が格納されている．

メモリ情報に関しては，ptrace システムコールのアタッチ時に，procfs の mem ファイルを読み込むことで取得できる．ただし，3.2 節で述べる Linux バイナリ互換機能を用いることで，Linux と FreeBSD で同じ Linux バイナリを実行できるため，テキスト領域と共有ライブラリは保存する必要がない．

3.2 ABI 変換

2 章で述べたように，OS 毎に ABI が異なる．本実装では，ABI の変換により各 OS のプロセス実行状態の表現差異を吸収する．

システムコールの差異は，FreeBSD カーネルの Linux バイナリ互換機能[4] を用いて，システムコール番号，引数値の変換，引数の渡し方の相互互換性を実現する．Linux バイナリ互換機能によって，Linux と FreeBSD の対象プロセス実行時におけるスタックとレジスタの状態を同じ表現にすることができる．

メモリレイアウトの差異に関して，Linux では，prctl システムコールのメモリレイアウト変更機能を用いる．メモリレイアウト変更機能は，プログラムやライブラリのロード時に決まるメモリレイアウトを，プロセス実行中に変更できる．スナップショット時とレストア時のメモリレイアウトを同じレイアウトにすることで，メモリ内部を変更することなく，メモリレイアウトの差異に対応させる．メモリレイアウト変更機能を用いることで，ASLR によるプロセス生成毎のランダムなメモリレイアウトにも対応できる．しかし，FreeBSD は，Linux のメモリレイアウト変更機能に相当する機能がないため，新たに実装する必要がある．

3.3 レストア

レストアは，スナップショットで作成したファイルを用いて，転送先で生成したプロセスに対して，レジスタとメモリの復元を行う．プロセスの実行状態を復元するためには，C プログラムにおける main 関数が実行されるまでにレジスタとメモリの状態を復元する必要がある．ELF ヘッダで取得できるエントリーポイントの呼び出しの直前に，レジスタやメモリの状態を復元することを検討したが，本手法で用いる Linux バイナリ互換機能が main 関数実行前の初期化処理内で有効化されるため，エントリーポイント呼び出し時の実行状態復元は適切でない．本手法では，main 関数の最初の命令を int3 命令 ☆ に置き換えることで，実行状態復元の機会を再現する．

レジスタの復元は，int3 命令による停止時に，ptrace システムコールの SETREGS コマンドにより行う．Intel x86 アーキテクチャでは，レジスタ構造体の情報を復元する際に，構造体の差異とセグメントレジスタの扱いの差異を考慮しなければならない．Linux では，システムコール番号をメンバ変数 orig_rax に格納し，システムコールの返り値をメンバ変数 rax に格納する．FreeBSD では，どちらもメンバ変数 rax に格納する．rip メンバ変数が指す命令からシステムコール実行の前後を判断して，FreeBSD の rax に格納する値を決める必要がある．また，メンバ変数の順番も異なるため，各 OS が持つ構造体のメンバ変数に合わせて格納しなければならない．セグメントレジスタは，OS がプロセス生成時に割り当てた値を使用する必要がある．本手法では，スナップショット時に取得したセグメントレジスタの値を使用せず，復元先の OS が割り当てた値をそのまま利用する．

メモリの復元は，int3 命令による停止時に，procfs の mem ファイルを介して，データ，ヒープ，スタックの各領域のメモリ状態を書き込む．この際，3.2 節で述べたメモリレイアウト変更機能を用いてスナップショット時のメモリ配置を再現することで，メモリ情報の内容を変更せずにプロセスを復元できる．

4. おわりに

本稿では，異種 OS 間コンテナ・マイグレーション実現に向けた，Linux-FreeBSD 間プロセス・マイグレーション実現方式の検討を行った．システムコールに関する差異は Linux バイナリ互換機能を用い，メモリレイアウトに関する差異はメモリレイアウト変更機能を用いて吸収する．

今後の課題として，実運用に向けてネットワークソケットやファイルポインタなどのカーネル等プロセス外で管理されているプロセス関連情報を対象プロセスのマイグレーションに含める技術の実現がある．

参考文献

1) 451 Research, ”451 Research: Application containers will be a $2.7bn market by 2020”, 2017.
2) CRIU Project: CRIU Main page, `https://criu.org/`, 2010 (accessed November 1 2017).
3) Klaus P. Ohrhallinger: Virtual Private System for FreeBSD, `http://www.7he.at/freebsd/vps/`, 2010 (accessed November 6, 2017). EuroBSDCon 2010.
4) Jim Mock: Linux Binary Compatibility, https://www.freebsd.org/doc/handbook/linuxemu.html, 1999 (accessed November 2, 2017).

☆ デバッガへのトラップ用割り込みを発生させるアセンブラ命令