B-020

CRIU を利用したコンテナ分割手法
A Method for Isolation of Specific Processes from A Container Using CRIU

高家 雄太郎 † 乃村 能成 † 谷口 秀夫 †
Yutaro Takaie Yoshinari Nomura Hideo Taniguchi

1. はじめに

コンテナ内にある複数のプロセスから特定のプロセスを別のコンテナとして分割したい．これを実現するためには，特定のプロセスのチェックポイントを作成し，別コンテナに復元することで，実現可能であると言える．そこで，Docker[1] と CRIU[2] を用いて，分割が実現可能か否か調査し，分割手法の案を述べる．

2. コンテナの分割

2.1 コンテナの分割とは

コンテナの分割とは，1 つのコンテナを複数のコンテナに分けることである．**図 1** を用いて説明する．図 1 は 1 つのコンテナを 2 つに分割する様子を表す．プロセス B を新しく作成したコンテナ 2 に移行することでコンテナを 2 つに分割できる．

2.2 実現方針

コンテナの分割の手順について，**図 2** に示し以下に説明する．なお，1 つのコンテナを 2 つに分割する場合を例としてあげる．

(1) 新規コンテナを作成

(2) コンテナ内の 1 つのプロセスのみについてチェックポイントを作成

(3) 作成したチェックポイントを (1) で作成したコンテナに復元

なお，コンテナ内にプロセスが複数ある場合，上記手順 (2)，(3) を繰り返す．

Docker では，コンテナのチェックポイントの作成を CRIU により実現している．そこで，CRIU を用いて，上記手法が実現可能か否か調査する．

3. CRIU

3.1 CRIU とは

CRIU とは，Linux 上で走行するプロセスについて，ユーザ空間からチェックポイントの作成やプロセスの復元を行うツールである．チェックポイントの作成とは，走行中のアプリケーションのメモリの状態やプロセスツリーの情報等をアプリケーションの状態として取得することである．復元とは，取得した状態からプロセスを再開させることである．

3.2 Docker と CRIU の関係

Docker は CRIU を利用し，コンテナを対象としたチェックポイントの作成とチェックポイントからのコンテナを復元を実現している．チェックポイントの作成とコンテナの復元を行う様子を図 2 を用いて，以下で説明する．Docker は，RPC を用いて，CRIU を利用する．CRIU は RPC リクエストを受けとり，コンテナ内のプロセスのチェックポイントの作成や復元を行う．コンテナを対象としたチェックポイント作成や復元は，コンテナ内で動作する全てのプロセスを対象に CRIU でチェックポイントを作成し，新規のコンテナ内へ復元することで実現される．

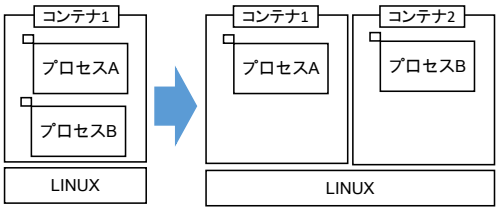

図 1 コンテナの分割

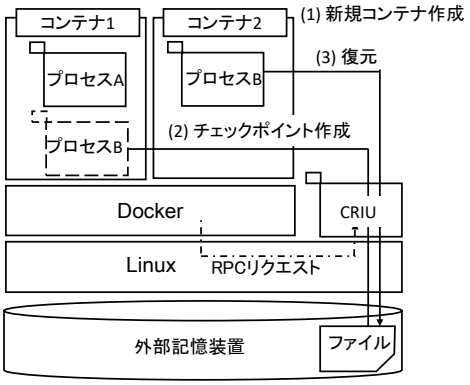

図 2 Docker による CRIU を利用したチェックポイントの作成とプロセスの復元

3.3 CRIU の動作

3.3.1 解析方針

CRIU はドキュメントの整備が十分でない．このため利用方法や動作の詳細は，CRIU と CRIU を利用するアプリケーション間の情報のやり取りや，CRIU のソースコードから判断する必要がある．Docker と CRIU 間でやりとりする RPC のパラメータには，コンテナ内のプロセスについてチェックポイントの作成や復元に必要な情報が含まれている．そこで，これを解析することで，Docker がコンテナ内のプロセスについてチェックポイントの作成や復元に必要な情報を含み，かつ 1 つのプロセスのみを対象としてチャックポイントを作成できる RPC のパラメータを導出できる．

†岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University

3.3.2 動作内容

(1) Docker が RPC で CRIU に送信するパラメータ
Docker が RPC で CRIU に送信するパラメータを解析した．この結果，パラメータは，チェックポイント作成の対象とするプロセスツリーのルートプロセスの PID，ルートファイルシステムのパス，およびコンテナ外部にマウントされているファイルシステムのパスなどの情報を含むことが分かった．このパラメータを用いてコマンドラインから CRIU を実行することで，コンテナのチェックポイントを作成するために Docker に呼び出された CRIU の処理と同様の処理を得られる．そこで，コンテナ内の一部プロセスについて，チェックポイントの作成を行うために，指定する PID のみ変更を行う．これを実行した結果，出力されたログから，PID 名前空間を書き出す際に，問題が発生することが分かった．

(2) PID 名前空間情報に関する処理流れ
PID 名前空間情報について該当部分をソースコードから解析した．PID 名前空間の情報を書き出す際の処理流れを**図 3** に示し，以下に説明する．

(A) CRIU 自身の名前空間の情報を取得する．

(B) パラメータとして与えた PID をもつプロセスの名前空間の情報を取得する．

(C) (A) と (B) で取得した PID 名前空間が一致するか否かを比較する．

(D) CRIU の PID 名前空間と対象プロセスの PID 名前空間が一致しない場合，対象プロセスの所属する PID 名前空間からみた PID が 1 であるか否かの確認を行う．

(E) 対象プロセスのコンテナの PID 名前空間からみた PID が 1 である場合，PID 名前空間情報の書き出しを行う．

このように，CRIU の所属する PID 名前空間と異なる PID 名前空間に所属するプロセスを対象にチェックポイントを作成する場合，指定するプロセスは対象プロセスの PID 名前空間からみた PID は 1(ルートプロセス) でなければならない．このため，現在の CRIU は，ホスト環境からコンテナ内の一部プロセスを対象としたチェックポイントの作成はできないことが分かる．

4. CRIU を利用したコンテナ分割の手法

3.3.2 項の結果から．以下の 2 つのことが分かる．

(1) 対象プロセスはルートプロセスでなければならない．

(2) Docker がコンテナのチェックポイントを作成する場合，対象プロセスつまりコンテナは CRIU と異なる名前空間に属する．

これらをふまえて，2 つの案とその課題を示す．

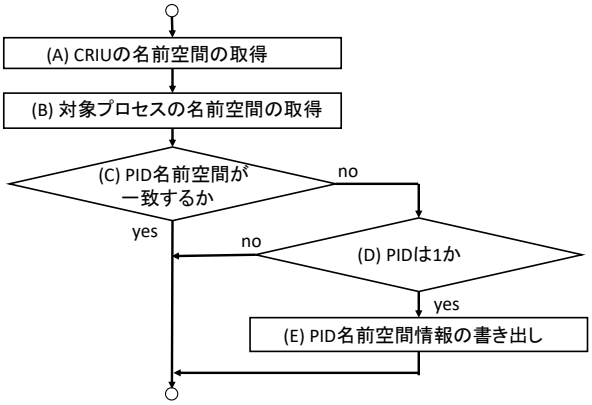

図 3 PID 名前空間情報を書き出すか否かの処理流れ

(案 1) ルートプロセス以外のプロセスもチェックポイント作成の対象とできる実装に CRIU を変更
CRIU は，異なる PID 名前空間に属するルートプロセス以外のプロセスを対象としたのチェックポイントからプロセスを復元する手段を備えていない．例えば，ある名前空間に属するルートプロセスでないものを対象に作成したチェックポイントを復元しようと思うと，名前空間のルートプロセスつまり init プロセスが存在しないため，これを何にするかという問題が生じる．

(案 2) 同一コンテナ内から CRIU を実行
これは，CRIU 自体を最初から対象コンテナ内の 1 プロセスとして実行する方法である．しかし，この手法では，CRIU を意識したコンテナを作成しなければならないという問題がある．また，CRIU のバイナリがコンテナごとに必要となる．

上記実現案により，より問題が詳細化された．これらの詳細化された問題について更に検討を進めることで，実現が可能であるといえる．

5. おわりに

本稿では，コンテナの分割について，その実現手法と実現する上での課題を示した．また，これらをふまえ，2 つの改善案を提案した．1 つは，ルートプロセス以外のプロセスも対象とできる CRIU の改変案，もう 1 つは，コンテナの構成を見直し，目的のコンテナ (PID 名前空間) 内に CRIU 自体を配置して操作を実行する案である．残された課題として，両提案の比較および実装がある．

参考文献

[1] Docker, Inc: Enterprise Container Platform — Docker, Docker, Inc. (online), available from ⟨https://www.docker.com/⟩ (accessed 2019-6-21).

[2] checkpoint-restore: CRIU, checkpoint-restore (online), available from ⟨https://github.com/checkpoint-restore/criu⟩ (accessed 2019-6-21).