仮想化技術による世界 OS の実装の提案

石田 航[†] 新城 靖[†]
佐藤 聡[†] 中井 央[†]

1. はじめに

誤った操作や他者からの攻撃によるファイル破壊はコンピュータを操作していく上で起こりうる問題である。それらの破壊からファイルの保護を行いたいという要求がある。さらに、計算機環境を複製することで、効率的に回帰テストを行いたいという要求も存在する。これらの要求に応えるため、本研究室では世界 OS[1)2)] という OS を開発している。

世界 OS とはプロセスの実行環境を世界としてとらえた OS のことである。本稿における世界は、プロセスとファイルの入れ物のことをさす。世界の操作には世界の**生成**、**削除**、**融合**、**内容一覧**の 4 種類が存在する。これらの操作を応用することにより、ソフトウェアのテストやファイル保護を行うことが可能となる。

過去にも世界 OS の実装[1)2)] が行われたことがあったが、それらの実装の難易度は高いものであった。例えば、世界の操作や性質を実装するために複雑なデータ構造を定義する必要がある。さらに、融合の操作におけるプロセスの移動が実現されていない実装も存在する。

これらの問題を踏まえ、本研究では世界 OS を仮想化技術と複数ソフトウェアを組み合わせて実現し、実装が困難だった機能を実装することを目標とする。本稿では仮想化技術の 1 つであるコンテナを用いる。

2. 世界の基本操作

この章では、具体的な例を交えながら世界の基本操作のはたらきについて述べる。例として、メールサーバを動作させながら設定変更やパッチの適用を行うことを考える。ここで、誤った設定やパッチの問題により動作不良が起こるとする。世界 OS では世界の操作によって計算機環境の複製と動作のテストが行える。

まず、Parent という世界で変更が行われていないメールサーバを動作させる。次に世界の**生成**によって

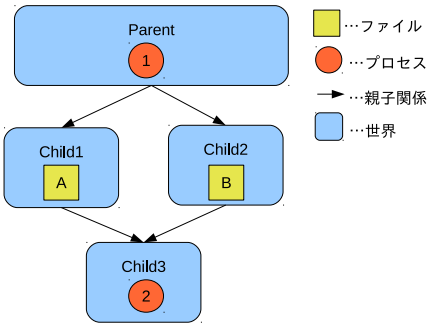

図 1 世界のイメージ

計算機環境の複製とファイルの継承を行う。ここでは Child1, および Child2 という世界を作成する。次に Parent から継承したファイルに変更を加える。Child1 では設定ファイルの変更を行い、Child2 でパッチを適用させる。Parent から変更したファイルの一覧は、世界の**内容一覧**で見ることができる。次に世界の**生成**で世界の多重継承を行う。ここでは Child1, および Child2 から Child3 という世界を生成する。最後に Child3 で新たなメールサーバを動作させ、テストを実施する。この一連の動作で作られる世界の関連を図にすると図 1 のようになる。ここで、テストに失敗すれば世界の**削除**を行って Child3 を消去し、Parent で動作しているメールサーバを動作させたままにする。テストに成功すれば世界の**融合**を行って、Child1, Child2, および Child3 のプロセスとファイルを Parent に移動させる。次に、元のメールサーバを停止させる。このことにより設定変更とパッチ適用後のメールサーバが Parent で動作するようになる。

3. コンテナを用いた世界 OS の設計

本研究では、Linux 上で世界 1 つをコンテナで実装する。コンテナを採用する理由としては、以降で述べるコピーオンライトを行うファイルシステムと組み合わせることで環境の高速な複製が可能となるためである。

世界の基本操作を行うと、図 1 のような世界間の

† 筑波大学

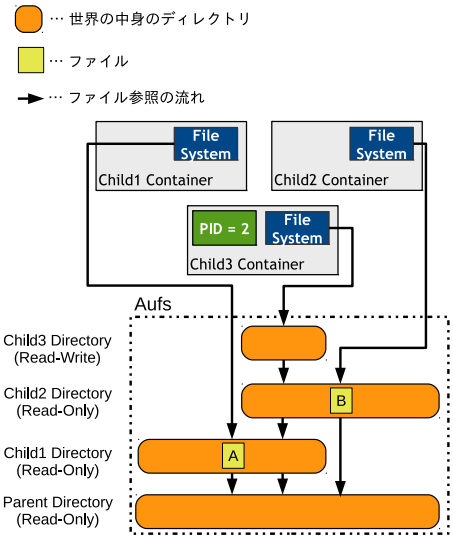

図 2 実装における世界の生成のイメージ

つながりが生成される。本研究では、このつながりをディスクに保存する場合、XML 形式で保存する。理由としては XML は階層構造を記述しやすく、記述したデータ構造がディスクに保存できるためである。

世界を操作するユーザインタフェースは端末上のコマンドとして実装される。実装言語はスクリプト言語の 1 種である Python を用いる。他にもコピーオンライトを実現するファイルシステムの Aufs を用いる。

生成は次のようにして実現を行う。はじめに、ホスト上で世界の中身を保存する、**世界の中身のディレクトリ**を作成する。次に、PID とファイルシステムをネームスペースによって隔離させ、コンテナを生成する。最後に、生成した子世界および親世界の**世界の中身のディレクトリ**を Aufs によって重ねあわせて、コンテナのルートファイルシステムとする。この時、新しく生成した世界のディレクトリを読み書き可、その親のものを読み込み専用にすることで、ファイルのコピーオンライトを行えるようにする。図 1 における Child1, および Child2 から Child3 を生成したとすると、そのイメージは図 2 のようになる。

削除は次のようにして実現を行う。はじめに、削除する世界のコンテナ中の Aufs ファイルシステムをアンマウントする。最後に削除する世界の**世界の中身のディレクトリ**およびコンテナのルートディレクトリを削除する。この時、コンテナ中のプロセスもすべて終了させる。

融合は、融合する世界からされる世界へファイルとプロセスの移動を行うことで実現する。さらに融合する世界のコンテナは削除される。ファイルの移動は、**世界の中身のディレクトリ**に含まれるファイルの移動によって実現する。プロセスの移動はプロセス情報の書き換えにより、プロセスが所属するコンテナを変更することで実現する。

内容一覧は、コンテナに含まれるプロセスやファイルをそれぞれ表示することで実現する。表示されるファイルは**世界の中身のディレクトリ**中のファイルである。また、表示されるプロセスはコンテナ中で動作するプロセスである。

4. 関 連 研 究

関連研究として、Prism[3] と呼ばれるファイルシステムが挙げられる。Prism とはゲストである仮想計算機のファイルシステムをホスト側で一括管理するファイルシステムである。Prism にはゲストのファイルシステムのコピーオンライト機能が備わっており、効率的にファイルシステムの複製を行うことができる。

Prism と本研究が異なる点は、融合の操作を考慮しているかしていないかの点である。世界 1 つを仮想計算機とみなした場合、Prism では融合の操作が行えない。一方、本研究では融合の操作を行えるようにする。

5. まとめと今後の課題

本研究では仮想化技術と既存の技術によって世界 OS の実装を容易にし、実現が困難な箇所を実装することを目標とする。具体的な手法としてはコンテナと呼ばれる仮想化技術や他のソフトウェアを組み合わせ、世界とその操作を実装する。

現在までに既存研究の調査とコンテナによる世界 OS の設計を行った。今後の課題としてはコンテナ間でのプロセスの移動を行うこと、およびユーザインタフェースを完成させることが挙げられる。

参 考 文 献

1) Jun Sun, Yasushi Shinjo, Kozo Itano. ”The Implementation of A Distributed File System Supporting The Parallel World Model”, The Third International Workshop on Advanced Parallel Processing Technologies, pp.43-47, 1999.
2) 石井 孝衛, 新城 靖, 板野 肯三. ”プロセストレース機能を用いた世界 OS の実現”, 情報処理学会論文誌 Vol.43, No.6, pp.1702-1724, 2002.
3) Xin Zhao, Kevin Borders, Atul Parakash. ”Prism: providing flexible and fast filesystem cloning service for virtual servers” Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp.388-407, 2008.