6U-06

プログラム実行環境移送のための資源追跡機能のユーザレベルでの実現

大西 史洋† 黒木 勇作† 横山 和俊† 谷口 秀夫‡
高知工科大学情報学群† 岡山大学大学院自然科学研究科‡

1. はじめに

近年の多くのクラウドコンピューティングは広域分散システムによって実現されている．このようなシステム形態において，負荷の分散や計算機のメンテナンスのために，別の計算機へ実行環境を移動させ再配置することが頻繁に行われる．そのため，クラウドコンピューティングでは応用プログラム(以下 AP と呼ぶ)と実行環境を移送する技術が重要となっている．移送するための技術として提案されている手法として，移送する AP と使用している資源および，同じ資源を共有している AP を特定・追跡する技術が提案されている[1]．しかし，この技術ではカーネルの修正が必要であり，OS のアップデートなどへの対応が困難となる．また，追跡対象となる資源が増加すると，追跡時間が増大する問題もある．

本稿では，AP 実行環境の資源追跡機能をユーザレベルで実現する方式を提案する．具体的には，AP が発行するシステムコールをシステムコールライブラリ内部で監視し，取得したデータから，AP 実行環境の資源追跡機能を実現する．

2. 資源追跡機能

2.1. 従来手法

文献[1]の従来手法では，open システムコールが発行されたとき，カーネル内部でシステムコールハンドラが実行されることに着目し，システムコールハンドラ内部で資源追跡機能を実現している．しかし，カーネルを編集して資源追跡機能を実現しているので，カーネルのバージョンアップへの対応が困難になる問題がある．具体的には，カーネルがバージョンアップされるたびにカーネルを編集し再コンパイルする必要があるので，編集とコンパイルで時間がかかる問題がある．

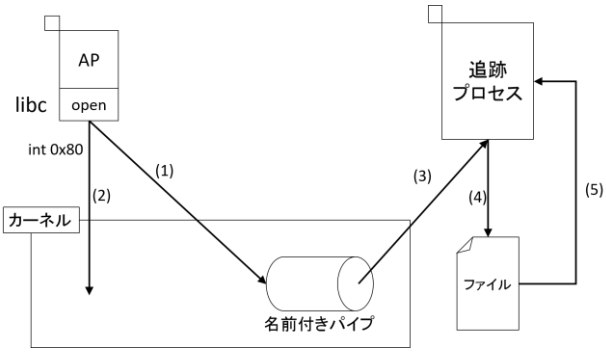

図 1 追跡機能の処理の流れ

2.2. 提案手法

提案手法では，open 関数が実行される時に，システムコールライブラリの open 関数が呼び出されることに着目し，システムコールライブラリ内で資源追跡機能の実現を行う．以降では，FreeBSD 11.0 STABLE での実現を説明する．

2.2.1. アクセス監視

図 1 は資源追跡機能の 1 つの機能であるファイルへのアクセス監視の流れを表している．アクセス監視の処理の流れは以下の通りである．

(1) AP が open システムコールを発行したとき，/usr/src/lib/libc/sys/open.c 内の open 関数が呼び出され，オープンするファイル名やオープンするフラグなどの情報が渡される．open システムコールは以下の通りに実装されている．
int open(const char *path, int flags, …)
open.c 内では， AP を特定するためのプロセス ID，AP がアクセスしているファイルのパス，アクセス時のオープンフラグを取得している．これらのデータ取得後，名前付きパイプへ出力する．

(2) ソフトウェア割り込み命令が実行されカーネル内の open 処理へと移行する．

(3) 資源追跡プロセスは，(1)の名前付きパイプへの書き込みを検知すると，名前付きパイ

User level implementation of resource tracking function for migration program execution environment
†Fumihiro Ohnishi，†Yusaku Kurogi
†Kazutoshi Yokoyama，‡Hideo Taniguchi
†School of Information，Kochi University of Technology
‡Graduate School of Natural Science and Technology，Okayama University

Copyright ©2018 Information Processing Society of Japan.
All Rights Reserved.

プからファイル名とフラグの情報を読み出す.

(4) (3)で蓄積されたデータが一定数に達した場合，蓄積されたデータを別のファイルへ書き出す.

2.2.2. 共有関係に着目した資源特定

文献[1]では，AP のファイル操作を，読み込みのみと書き込みのみの 2 種類に分類し，一括して移送する必要がある AP とファイル群を特定している．例えば，AP1 と AP2 がファイル X を共有している状況で AP1 を移送する場合，移送モデルは以下の 2 つのモデルに分けることができる．

(モデル1) AP1 と AP2 がファイル X に対して読み込みのみの場合，AP1 と AP2 の間には処理の依存関係がないため，AP1 とファイル X のみを移送すればよい．ただし，AP2 の動作のため，ファイル X をコピーしたものを残す必要がある．

(モデル2) AP1 と AP2 のどちらかがファイル X に対して書き込みを行う場合，AP1 と AP2 の処理の間には依存関係がある．よって，AP2 も移送対象とし，AP1 とファイル X と一緒に移送する．

本研究では，文献[1]で提案されている共有関係に着目した追跡アルゴリズムをユーザレベルで実現した．具体的な処理を以下に示す

(A) 資源追跡プロセスに対して，外部から資源特定の開始命令が与えられる．このとき，どの AP を移送するか指定される．

(B) 資源追跡プロセスは名前付きパイプからの読み出しを中断し，これまでに保存したデータをファイルから読み出す.

(C) 資源追跡プロセスは，ファイルの共有関係に着目して，追跡アルゴリズムにより移送対象とする AP とファイル群を特定する.

3. 評価

3.1. 評価内容

資源追跡機能のうち，ファイルのアクセス監視のオーバヘッドを評価した．評価環境を表 1 に示す．計測を行うために，ファイルをオープンする AP と，オープンされるファイル A をそれぞれ 1 つ用意した．AP はファイル A に対して open システムコールを発行する．このとき，ファイル A はキャッシュされていない状態で計測した．ファイルのアクセス監視を行っている open.c と修正を行っていない open.c を用いた場合について，5 回計測し平均を取った．測定には gettimeofday 関数を使用した.

表 1 評価環境

ホスト OS	Windows10 Pro 64bit
CPU	Intel® Core™ i7-4700MQ CPU @ 2.40GHz
ゲスト OS	FreeBSD 11.0 STABLE 32bit
仮想環境	VMware Workstation 12 Player 12.5.8

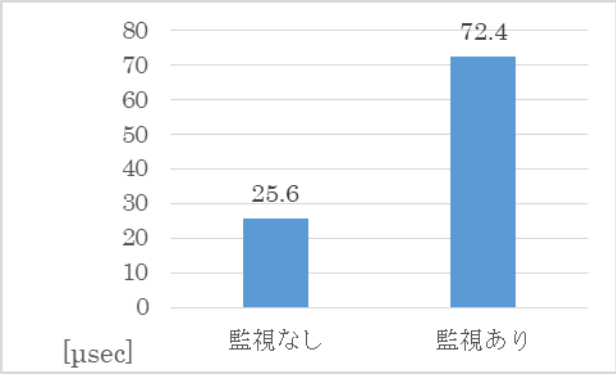

図 2 ファイルアクセス監視のオーバヘッド

3.2. 評価結果

図 2 より，ファイルアクセスを監視している場合のオーバヘッドは 72.4μsec であり，アクセス監視なしのオーバヘッドは 25.6μsec である．つまり，提案手法は監視なしに比べて約 2.8 倍であることが分かる．これは，名前付きパイプへ書き込むたびに，資源追跡プロセスが読み出しのために同期をとっているからである．今後，名前付きパイプへ書き込む度に読み出すのではなく，一括して読み出すことで改善を図る.

4. おわりに

本稿では，資源追跡機能をシステムコールライブラリ内部で実現することにより，ユーザレベルでの資源追跡機能の実現する方法を示した．また，ファイルへのアクセス監視のオーバヘッドを計測して評価を行った．今後は共有関係に着目した資源の特定手法の評価を実施する.

謝辞

本研究の一部は，科研費(17K00107)の支援を受けて実施しています.

参考文献

[1] 畑翔太，谷村直哉，横山和俊，”ファイルの共有関係に着目した移送するプログラムと実行環境の特定方式”，情報処理学会研究報告，Vol.2015-DPS-164 No.11，pp.1-6 (2015).

Copyright ©2018 Information Processing Society of Japan.
All Rights Reserved.