7X-01

複数計算機上に跨るプログラム実行環境の特定手法の提案

黒木 勇作† 西 拓人† 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

広域分散システムでは，応用プログラム(以降 AP と略す)の実行環境を他の計算機に移送させることが発生する．代表的な手法である仮想マシンを移送させる方法は，仮想マシンすべてを移送させるため，移送に時間がかかる問題がある．この問題に対し，移送対象となる最小限の AP 実行環境を特定し移送する手法を提案している[1][2]．提案手法は，移送対象をファイル単位で特定する特定ステップ[1]，およびサービス停止時間を短縮させるような転送を行う転送ステップ[2]からなる．本稿では文献[1]に示す手法を，同ネットワーク上に配置された複数計算機上に拡張し，複数計算機上に跨るような AP 実行環境を特定する手法を提案する．

2. 移送モデル

2.1. AP 実行環境特定の基本的な考え方

ある AP の実行環境を移送すると，同じファイルを参照していた別の AP の実行に影響を与える．また，別の AP と通信をしていた場合，その AP を移送してしまうと通信先の AP の実行に影響を与える．つまり，AP 単独ではなく，ファイルの共有関係や通信関係を考慮し，移送対象を特定する必要がある．

2.2. 共有関係に着目したファイル資源の追跡

ファイルの共有関係は， open システムコールを監視することで追跡することができる．提案手法では，open 時のアクセス種別が read-only か(open ファイルが read-only であるか)，1 回でも write を伴う可能性があるか(open ファイルが read-write であるか)に着目しそれぞれの場合について，移送対象を追跡するアルゴリズムを実現している．

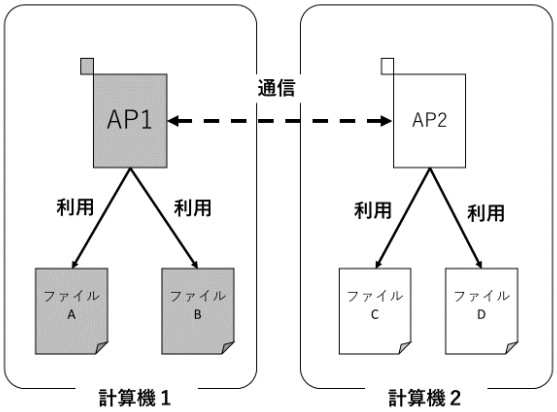

図 1 複数計算機に跨る AP 実行環境

2.3. 計算機との通信の追跡

データセンタ内には通常複数の計算機があり，複数の AP と通信を行いながら処理を行う．この時，AP やファイルが複数計算機に跨って存在すため，複数計算機にわたって追跡を行う必要がある．具体例を図 1 に示す． AP1 実行環境(図中では計算機 1 内)を移送する場合，AP1 と AP1 が利用しているファイル A・ファイル B を転送する必要がある．このとき AP1 が，異なる計算機(図中では計算機 2)に存在している AP2 と通信をしている場合，AP2 の有無によって，AP1 の動作に差異が発生する場合がある．そのため，AP2 も AP1 の実行に必要であると考え，AP2 と AP2 が利用しているファイル C とファイル D も転送する必要がある．提案手法では，AP が発行する socket 通信を監視することで，通信を行っている AP の組み合わせの特定を実現する．

3. 実現方法

提案手法では，open/accept/connect の各システムコールを監視して，どの AP がどのファイルを利用しているのかを全て特定し，実行環境を追跡する．open は，文献[1]の手法を用いて監視する．ここではネットワークを跨る追跡を行うための accept/connect の監視方法を説明する．

例を図 2 に示す．socket 通信は socket を始めとする，いくつかのシステムコールの組み合わせにより行われる．この時，送信側では connect，受信側では accept を利用する．accept/connect

---

Resource tracking method of program execution environment spanning multiple computers
Yusaku Kurogi†, Takuto nishi†
Kazutoshi Yokoyama†, Hideo Taniguchi‡
†School of Information, Kochi University of Technology
‡Graduate School of Natural Science and Technology, Okayama University

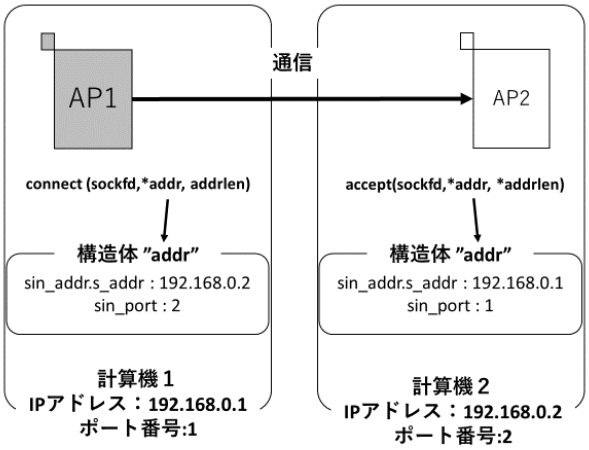

図 2　socket 通信の例

のシステムコールの定義を以下に示す.

```
int accept(int sockfd, struct sockaddr
    *addr, socklen_t *addrlen);
int connect(int sockfd, const struct
    sockaddr *addr, socklen_t addrlen);
```

これらのシステムコールを監視し，どの AP 間で通信が発生しているかを追跡する．この追跡は，以下の 2 段階で行われる.

(1) 通信を行っているプロセスの特定

上記の 2 つのシステムコールには，sockaddr 構造体(図中では addr)を持ち，その中にはそれぞれ接続先の IP アドレスとポート番号が格納されている．システムコールライブラリ内で sockaddr 構造体から取得した情報を，特定の場所に保存しておく．具体的には，クライアント側は自分の PID と IP アドレス，サーバの IP アドレスとポート番号を記録する．サーバ側は，自分の PID と IP アドレス，接続してきたクライアントの IP アドレスとポート番号を記録する．この IP アドレスを突き合わせることで，通信を行っているプロセスを特定する.

(2) プロセスが実行している AP の特定

アクセスを行った AP を特定するために、プロセス生成時(execve システムコール発行時)に，PID と実行される AP のパスの組み合わせを特定の場所に記録しておく．その記録と PID を対応付け，AP のパスを取得する.

4. 評価

4.1 評価内容

socket 通信を監視する際のオーバヘッドを評価した． acceptを一度発行するプログラムと，connect を一度発行するプログラムを 1 つずつ用意し，それぞれ異なる計算機に配置して相互に socket 通信させる．監視をしない場合と，connect の監視を行う場合のプログラム実行時間を，gettimeofday 関数を用いてマイクロ秒まで計測し，それぞれの項目で 5 回の平均時間をとったものを比較した．なお，accept は OS による接続待ちが発生し，正確なオーバヘッドが測定できないため，accept.c 内における追記部分の処理時間を，gettimeofday 関数を用いてマイクロ秒まで計測し，5 回の平均時間をとったものを比較した.

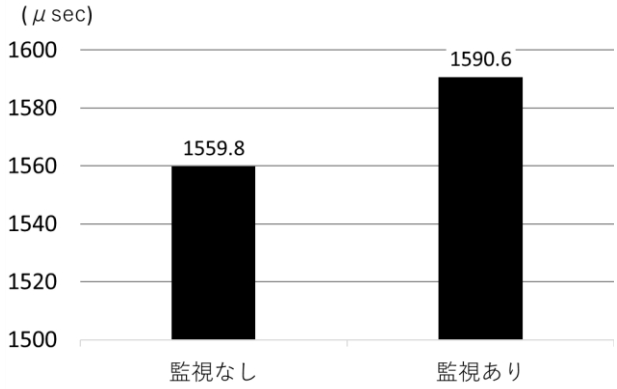

図 3　connect 監視オーバヘッド

4.2 評価結果

connect 監視のオーバヘッドを図 3 に示す．監視なしの場合，プログラム実行時間は約 1560 マイクロ秒となった．監視時は，約 1591 マイクロ秒となり，監視なしの場合と比較して約 2%オーバヘッドがあった．また，accept システムコールでの監視オーバヘッドの増加は約 35 マイクロ秒であった.

5. おわりに

本稿では，複数計算機上に跨るような AP 実行環境の特定手法を提案した．残された課題として，実行環境の追跡時間を評価することがある.

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] 大西史洋 他，“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表現”，情報処理学会第 80 回全国大会，第 3 分冊，pp.319-320 (2018).

[2] 黒木勇作 他，“サービスの停止時間を短縮するプログラム実行環境のプリコピー移送手法”，情報処理学会研究報告，vol.2018-DPS-175，No.16，pp1-6 (2018).