ソケット通信に着目した計算機にまたがるソフトウェア実行環境の特定方式

1190352　西 拓人　【 分散処理 OS 研究室 】

1　はじめに

広域分散システムでは，効率的な運用のために応用プログラム (以降 AP と呼ぶ) の実行環境を別の計算機へ移送することがある．実行環境を必要最小限のみ移送する手法として，先行研究 [1] で示されているものがある．しかし，この手法では，同計算機内のファイル資源へのアクセスを監視するに留まり，計算機をまたいだ資源の依存関係を特定できない問題がある．そこで本研究では，計算機にまたがるソフトウェア実行環境を特定する方法を提案する．

2　提案方式

2.1　提案方式の概要

AP を移送するとき，それ単独ではなく，利用する資源も一緒に移送する必要がある．さらに，他の AP が同じ資源を利用している場合，その AP も移送対象となる．つまり，AP と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．

AP が利用する資源は，ファイル，プロセス間通信，ネットワーク通信の 3 つに分類できる．提案方式では，監視フェーズでこれらの資源利用情報をフックし，その情報を基に，追跡フェーズで移送対象となる AP やファイルを特定する．

2.2　監視フェーズ

監視フェーズでは，システムコールと C ライブラリ関数に追記をして改造し，資源利用情報をフックする．表 1 に，フック機構を追記した関数とそのフック情報を示す．AP やファイルを一意に識別するためには，ファイルパスと IP アドレスが必要だが，open 関数等からは AP の識別情報を PID でしか取得できない．そのため，exec システムコールを監視して，PID とファイルパスの対応を取得する．

2.3　追跡フェーズ

監視フェーズで得られたフック情報を基に，AP の依存関係を特定する．最終的には，AP の資源利用状況を網羅した依存関係リストを作成し，このリストを参照することで，ある AP を移送したいときに一緒に移送する必要がある資源を全て特定できるようにする．

はじめに，フック情報の PID をパスに置き換える．次に，各マシンで得られたフック情報を LAN 内のいずれか 1 台のマシンに集め，依存関係リストを作成する．このとき，open と fork のフック情報は、そのままではどのマシンの資源であるかが分からなくなるため，出身マシンの IP アドレスを付加してから集める．最後に，集められたフック情報をマージして依存関係リストを作成する．

表 1　監視に利用した関数

用途	関数名
ファイルアクセス	open
プロセス間通信	fork
ネットワーク通信	accept connect
PID とパスの解決	exec

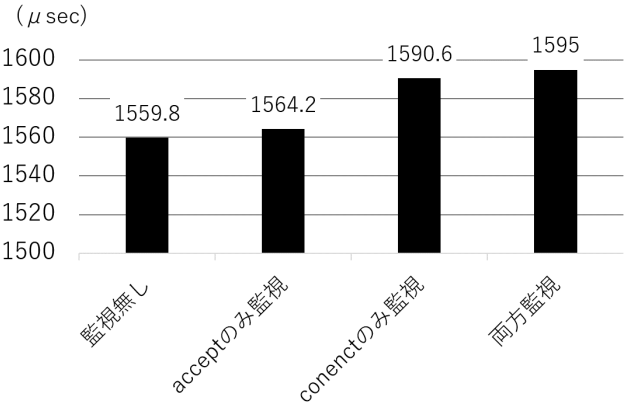

図 1　connect 監視機構のオーバーヘッド

3　評価

監視を実行していない状態としている状態の accept 関数及び connect 関数のオーバーヘッドを計測した．

accept 関数は，accept.c の追記部分の先頭と末尾でタイムスタンプを取り，その差分をオーバーヘッドとした．結果は，5 回計測した平均が 34.4 μ秒であった．

connect 関数は，単純なソケット通信のクライアント側プログラムの実行時間を監視有無それぞれで計測し，その差分をオーバーヘッドとした．グラフを図 1 に示す．単純なソケット通信のプログラムでも増加分は約 2%であり，十分無視できる数値であった．なお，図 1 で accept 監視のオーバーヘッドが無い様に見えるのは，サーバがクライアントに処理を返してから accept の監視機構が実行されるためである．

4　おわりに

本研究では，計算機にまたがるソフトウェア実行環境の特定が，十分実用に耐えうる実行時間で行えることを示した．

参考文献

[1] 大西 史洋，黒木勇作，横山 和俊，谷口秀夫: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，第 80 回全国大会講演論文集，第 3 分冊，pp.319-320(2018).