5V-07

複数計算機上に跨るプログラム実行環境の特定手法の評価

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

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，異なるデータセンタの計算機に移送させることが発生する．この移送に対し我々は，必要最小限の資源を特定し移送する手法を提案している[1][2]．提案手法は，実行環境を構成する資源をファイル単位で特定する特定ステップ[1]と，移送時の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．また，ネットワークを介して通信を行う AP を対象として文献[1]の手法を拡張し，複数計算機に跨る AP 実行環境の特定手法を提案した[3]．本稿では，文献[3]で提案した特定手法の評価について報告する．

2. 移送モデル

2.1. AP 実行環境特定の考え方

AP 実行環境は，通常，複数の AP がファイルを共有したり，互いに通信を行う．そのため，ある AP を移送すると，その AP とファイルを共有している AP や通信を行っている AP に影響を与える場合がある．提案手法では，そのような AP 同士は互いに依存関係を追跡し，移送対象の AP と依存関係にある AP は全て特定する．

2.2. 計算機内での資源の追跡

ファイルの共有関係は，AP がファイルを利用する際の open システムコールを監視して追跡する．提案手法では open 時のアクセス種別(read-only，read-write)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．また，同一計算機内における AP の親子関係を，fork システムコールを監視して追跡する．

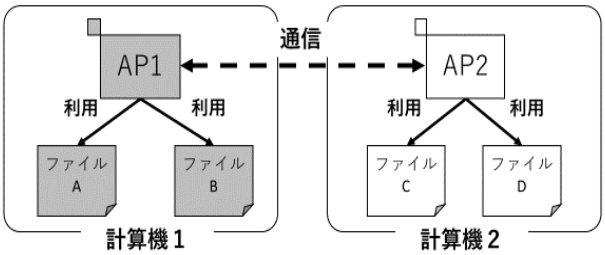

図 1 複数計算機上に跨る AP 実行環境

2.3. 複数計算機上に跨る資源の追跡

サービス処理を実現するために複数の計算機上で複数の AP が動作し，それらが互いに通信を行う．そのため，AP の実行環境は，複数の計算機に跨って追跡する必要がある．具体例を図 1 に示す．図中の計算機 1 内の AP1 はファイル A とファイル B を利用しているが，別計算機である計算機 2 の AP2 と互いに通信を行っている．そのため，AP2 が存在しないと AP1 の実行に何らかの影響が発生する．従って AP1 実行環境を移送する場合，AP2 と AP2 が利用しているファイル C とファイル D も移送対象とする必要がある．提案手法では AP のソケット通信を監視して追跡し，通信を行っている AP の組み合わせの特定を実現する．

3. 実装と評価

3.1. 実装

提案手法では，open/fork/accept/connect 各システムコールの C 言語におけるラッパー関数内に監視機構を実装した．各システムコールを発行する際の引数や返り値から，利用している資源を特定する．同一計算機内の追跡を文献[1]の手法，ネットワークに跨る通信の追跡を文献[3]の手法を用いて監視する．

3.2. システムコールのオーバヘッド

(1) 評価内容

各システムコール監視のオーバヘッドを評価した．open，close を合わせて 1 セット行った際の実行時間，fork，execve，connect をそれぞれ単独で 1 回ずつ行った際の実行時間を測定した．

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

(2) 評価結果

評価結果を図 2 に示す．グラフ上部の数値はシステムコールの実行時間である．各システムコールのオーバヘッドは，open+close は 46μ秒，fork は 64μ秒，execve は 43μ秒，connect は 56μ秒となった．オーバヘッドの要因として，共通した処理である，ユーザレベルで情報をリスト出力する際の処理が考えられる．このことから，システムコールを複数回行うことを想定した場合，元々処理時間の短い open や fork はオーバヘッドの影響が大きく，元々処理時間の長い execve や connect はオーバヘッドの影響が小さい．

3.3. 追跡時間

(1) 評価内容

AP 実行環境の依存関係を全て追跡する際の時間を評価した．実際の AP 実行環境として，銀行のオンラインシステムにおけるバッチ処理を想定し，それに近い処理を行うプログラムの資源利用情報[4]に対して追跡処理を行った際の追跡時間を測定した．用いるパラメータを表 1 に示す．ここで共有率は，AP に open されるファイルのうち，2 以上の AP から open されるファイル(以降共有ファイルと呼ぶ)の割合である．図 3 に共有ファイルへのアクセスを示す．図 3 では，10 ファイルに対し共有ファイル(図中の色付きファイル)は 4 となるため，共有率は 40%となる．

(2) 評価結果

ファイルの共有率別の追跡時間評価結果を図 4 に示す．グラフ横の数値は AP 数が 3000 の場合の追跡時間である．AP 数が増加すると比例して追跡時間も増加する．AP 数が 3000 の場合では，共有率 20%の場合の追跡時間は 32.4 秒，100%の場合の追跡時間は 53.3 秒となり，約 21 秒の差が発生した．この結果から，追跡するファイル数が増加すると追跡時間も増加するということがわかる．文献[4]での，実際の AP バッチ処理では，最大でも同時実行される AP 数は 800 程度である．本手法では AP 数が 800 の場合，追跡時間は 10 秒程度で追跡できている．

4. おわりに

表 1　追跡対象 AP パラメータ

AP 数	10, 20, 50, 100, 200, 500 1000, 2000, 3000
ファイル数	AP 数の 100 倍
共有率(%)	20, 40, 60, 80, 100
共有ファイルへのアクセス回数	3000

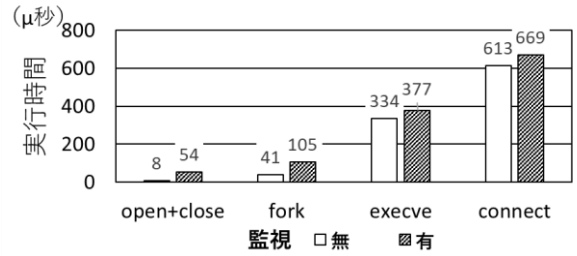

図 2　システムコールのオーバヘッド

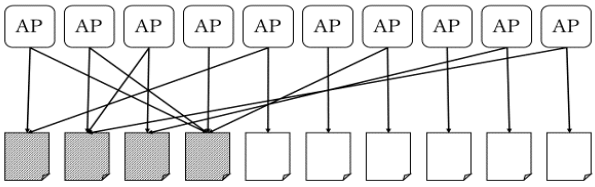

図 3　共有ファイルへのアクセス

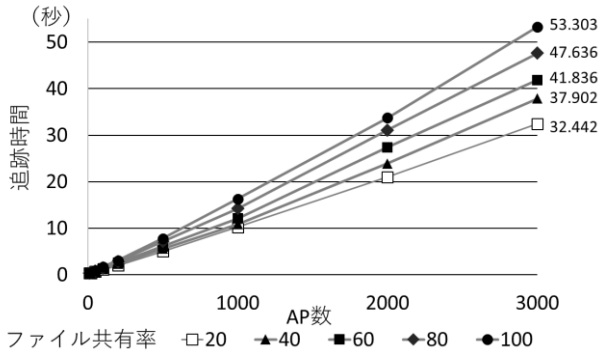

図 4　実行環境の追跡時間

本稿では，複数の計算機に跨るプログラム実行環境を想定し，提案手法に対して追跡時間の評価を行った．提案手法を用いることで，実際の AP 実行環境の AP 数の場合，10 秒程度で資源の追跡が行えることを明らかにした．

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，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表現”，情処第 80 回全大，第 3 分冊，pp.319-320 (2018).

[2] 黒木勇作，大西史洋，横山和俊，谷口秀夫，“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”，情処研報，vol.2018-DPS-175, No.16, pp1-6 (2018).

[3] 黒木勇作，西拓人，横山和俊，谷口秀夫，“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”，情処第 81 回全大，第 3 分冊，pp265-266 (2019).

[4] 田辺雅則，横山和俊，長尾尚，谷口秀夫，“オンライン処理とバッチ処理の処理負荷を分散制御する入出力制御方式の実装と評価”，情処論文誌，vol.61, No.2, (2020).