複数計算機上に跨るソフトウェア実行環境の移送手法

1225117　黒木 勇作　【 分散処理 OS 研究室 】

Migration method of software environment spanning multiple computers

1225117　Yusaku Kurogi　【 Distributed System and Operating System Lab. 】

1　はじめに

広域分散システムの効率的な利用のため，アプリケーションプログラム (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][3]．本研究では提案手法である，複数計算機上に跨る AP 実行環境の追跡機構の実装と評価を行い，その結果を示す．

2　移送モデル

2.1　実行環境特定の考え方

通常，AP 実行環境は，複数の AP がファイルを共有したり，互いに通信を行ったりしている．そのため，AP を移送すると，その AP とファイルを共有したり，通信を行ったりしている 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 同士の依存関係を追跡し，移送対象の AP と依存関係にある AP も全て同様に移送対象とする．

2.2　計算機内での資源の追跡

AP は，同一計算機内のファイルの情報をファイルアクセスによって利用する．提案手法では，計算機内のファイルの共有関係を，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．このとき，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，移送対象の追跡アルゴリズムを実現している．また，親子関係にある AP プロセスを，fork システムコールを監視して追跡する．

2.3　複数計算機上に跨る資源の追跡

AP 実行環境は通常，ネットワークの複数計算機上で，複数の AP が互いに通信しあって動作している．そのため複数計算機に跨る AP 実行環境の追跡が必要となる．この具体例を図 1 に示す．図中の計算機 1 では AP1 がファイル A とファイル B を利用している．AP1 を移送する場合ファイル A とファイル B も移送する必要があるが，AP1 は計算機 2 上にある AP2 と互いに通信している．そのため，AP2 が存在しないと AP1 の実行に影響が発生する．従って，AP1 実行環境を移送する場合，ファイル A とファイル B だけでなく，AP2 と AP2 が利用しているファイル C とファイル D も移送する必要がある．提案手法では，AP のソケット通信を監視して追跡を行い，通信を行っている AP の組み合わせを特定する．

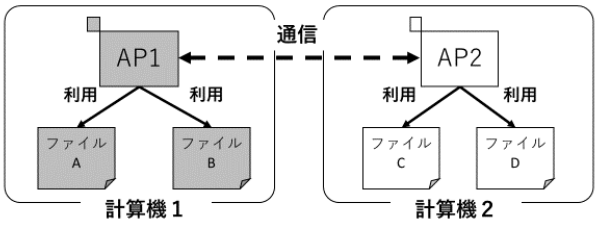

図 1　複数計算機上に跨る AP 実行環境

3　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する．これらのシステムコールの監視のうち，ネットワーク上で通信を行っている AP を特定するための，accept/connect システムコール監視の具体例を図 2 に示す．socket 通信では，受信側 AP は accept，送信側 AP は connect を利用する．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特定する．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は，sockaddr 構造体 (図中の 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，リストとして保存する．具体的には，送信側は自分の PID と IP アドレス，送信先の IP アドレスとポート番号を記録する．受信側は，自分の PID と IP アドレス，受信した通信先の IP アドレスとポート番号を記録する．得られた情報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，プロセス生成が生成される execve システムコールの発行時に，PID と実行される AP のパスの組み合わせをリスト形

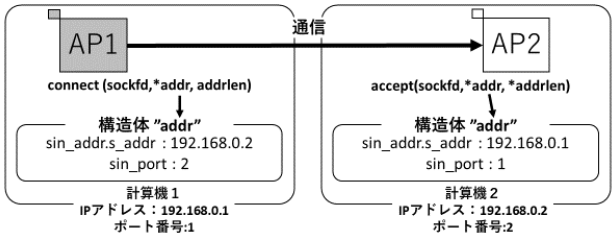

図 2　socket 通信の例

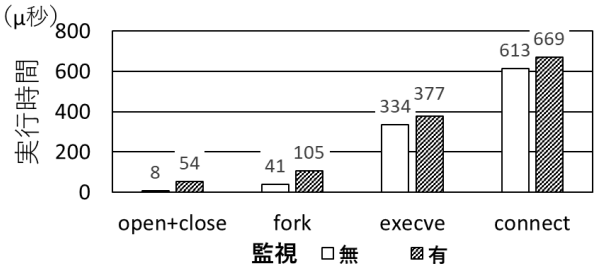

図 3　システムコールのオーバヘッド

式で記録しておく．その記録の PID と，(1) で得られたプロセスの PID を対応付け，AP のパスを取得する．

4　評価

4.1　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．

(2) 評価結果

評価結果を図 3 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．オーバヘッドの要因として，ユーザレベルでリストを出力する際の処理が考えられる．よって，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．

4.2　追跡時間

(1) 評価内容

AP 実行環境の依存関係を追跡する際の時間を評価した．評価に用いる AP 実行環境は，銀行のオンラインシステムで実行されるバッチ処理を想定した．文献 [4] で用いられているパラメータを参考に，実際の AP 実行環境に近い処理を行うプログラムの資源利用情報に対して，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．表中の共有率とは，全ファイルに対して，2 つ以上の 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．例えば，ファイル 100 個に対し共有ファイルが 40 個存在している場合，共有率は 40%となる．

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

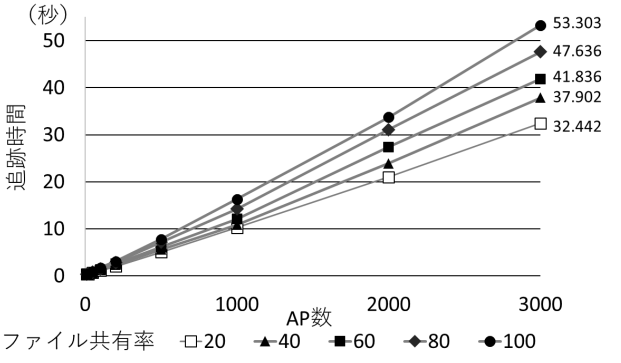

図 4　実行環境の追跡時間

(2) 評価結果

ファイルの共有率別の追跡時間を図 4 に示す．グラフ右側の数字は AP 数 3000 の場合の追跡時間を表している．AP 数 3000 の場合，追跡時間は，共有率が 20%であれば 32.4 秒，共有率が 100%であれば 53.3 秒となった．文献 [4] では，実際の AP バッチ処理の場合，最大でも同時に実行される AP 数は 800 程度となっている．提案手法では AP 数が 800 の場合，10 秒程度で追跡することができた．

5　おわりに

本研究では，複数計算機上に跨るソフトウェア実行環境を想定し，提案手法の実装と，提案手法を利用した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．

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).