Vol.2015-DPS-164 No.11
Vol.2015-EIP-69 No.11
2015/9/11

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法

畑 翔太[1]　谷村 直哉[1]　横山 和俊[1]

概要：広域分散システム上で，応用プログラムとその実行環境を移送する代表的な手法として仮想マシンを用いる手法がある．しかし，この方法ではオペーティングシステム (以降:OS と略す) とその OS 上で動作する応用プログラムをまとめて移送する必要がある．このことから，広域分散環境下において，容量の大きいファイルを移送することになる．よって，移送時間が増大するといった問題が発生する．そこで，本研究では，応用プログラムが利用する資源を特定することで，その応用プログラムと利用する資源のみを移送する方法の提案を行う．提案手法は，応用プログラムが利用する資源が他の応用プログラムからも利用されている可能性を考慮し，資源の共有関係を追跡し，一緒に移送しなければならない資源を特定する機能を有している．提案手法を用いて，apache を監視，追跡時間の測定を行った結果，十分実用的な時間で特定することが可能であった．

1. はじめに

近年，広域分散システムの技術発展により，クラウドコンピューティングが広く普及している．これらのシステム形態では，応用プログラム (以降，AP と呼ぶ) の実行する計算機の負荷に応じて動的に増減させることや，AP が実行する計算機を変更し再配置することが頻繁に行われる．

この再配置に関して，効率良く仮想マシンの再配置を行うかについての研究が盛んに行われている．例えば，サービスを別のサーバに移送する研究として，Haikun らは，実行時にメモリに書き込まれることに着目し，そのメモリを移送することで，サービスを移送することが出来ることが提案されている [1]．文献 [2] ではクラウドサービス内でサーバを移動させる時に，各サーバが公平に移動され，トラヒックをできるだけ小さくするような，移動アルゴリズムの提案を行っている．文献 [3] では限られた物理マシン群中に，できるだけ効率良く仮想マシンの多数配置を行うかなどについて，サービス提供側に求められる要件について整理したものである．文献 [4] では，低負荷な物理サーバー上に動く VM を移送し物理マシンを統合する回数と，消費電力の削減量についての提案である．

一方，災害時において，データセンタのシステムを他の遠隔地のデータセンタに退避し，システムを維持するため，AP の実行環境を移送する取り組みも行われている．前述の研究は，主にデータセンタを対象とし，高速なネットワークで繋がったストレージデバイスを用いて，移送を行うことを前提としている．このため，広域分散環境での移送には利用することが出来ない．広域分散システムを対象とした移送手法では，AP とその実行環境を移送する代表的な手法として，仮想マシンを用いる手法がある [5][6]．しかし，広域分散システム内で仮想マシンを用いて移送を行うときには，仮想マシンのディスクイメージを全てを移送する必要がある．このことにより，移送処理を行うために，多大な時間が必要となることが問題である [7]．また，広域分散環境では回線種別にもよるが，細い帯域をを使用していることもあるため，巨大なファイルを移送することは，多くの帯域を長時間占有する問題が存在する．

そこで，本研究では，AP が利用する資源の特定し，AP とその AP が利用する資源のみを移送する手法を提案する．具体的には，AP が発行するシステムコールを監視し，利用する資源の把握を行う．また，AP が利用する資源は，他の AP からも利用されている可能性がある．そのため，資源の共有関係に着目し，一緒に移送しなければならない資源の追跡を行う手法を提案する．

2. 移送モデル

2.1 AP とその実行環境を移送するときの問題点

通常，AP は単独で 1 つの資源を占有し，実行していることは少なく，それぞれの AP が 1 つのソフトウェア資源を共有し，参照を行っている．このため，AP と，その実行

[1] 高知工科大学
Kochi University of Technology

Vol.2015-DPS-164 No.11
Vol.2015-EIP-69 No.11
2015/9/11

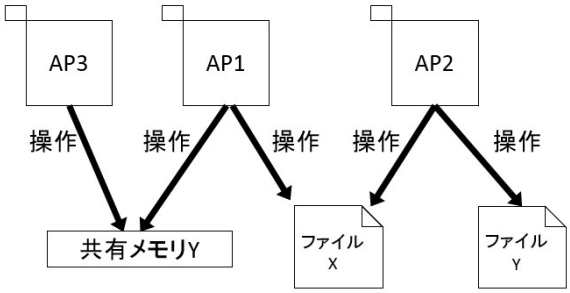

図 1 AP の資源共有の例

環境を移送するためには，移送対象となる AP がアクセスしているソフトウェア資源に対して，他にアクセスを行っている AP を特定する必要がある．この時，必要ならば，他の AP を移送対象とし，アクセスを行っている AP と，その実行環境も移送する必要がある．AP が実行を行うために，必要な資源は，(1) ファイル操作，(2) プロセス間通信，(3) ネットワーク通信の 3 つに分類できる．

(1) ファイル操作

ファイル操作は AP がファイル資源に対してアクセスを行った時に発生する．この時，移送を行う AP がアクセスしているファイル資源に対して他の AP がアクセスを行っていた場合は，他の AP の実行への影響を考慮して，移送をする必要がある．つまり，必要であれば，ファイル資源のコピーを行ったり，移送したい AP にアクセスを行っている他の AP について，その実行環境を一緒に移送を行う必要がある．

(2) プロセス間通信

プロセス間通信は同一計算機内で情報の共有を行っている．代表的な技法として，共有メモリが挙げられる．仮に，移送したい AP がプロセス間通信を行っていた場合は，移送対象の AP も他の AP もプロセス間通信を通じて，情報の授受を行っている可能性がある．つまり，プロセス間通信を行っている場合は，他の AP を移送対象に含め，一緒に移送を行う必要があると考えられる．

(3) ネットワーク通信

ネットワーク通信は外部の計算機と情報の授受を行っている場合である．この時，移送したい AP とアクセスを確立した場合は，移送対象の AP と情報の授受を行っている可能性があるため，外部の AP を必要な資源であるとみなす必要がある．

図 1 に，3 つの AP がそれぞれ資源を共有し実行している例を示す．AP1 と AP2 は，共にファイル X に対して操作を行っている．また，AP2 はファイル Y の操作を行っている．AP1 とファイル X を他の計算機に移送する場合は，ファイル X を移送すると移送元の計算機で AP2 が実行できなくなるため，実行できるように AP2 とファイル Y を移送する必要がある．また，AP1 と AP3 は共有メモリ Y を用いたプロセス間通信を行っている．この場合，AP 1 と AP3 は情報を共有している可能性があるため，AP1 のみを移送することが出来ない，よって，AP3 も一緒に移送する必要がある．この例で示したように，ある AP を移送するとき，AP とそれが利用する資源だけを移送すれば良い訳ではないことが分かる．よって，資源を共有している他の AP と，その実行環境を特定し，その関係性を把握する必要がある．つまり，移送する AP が利用する資源にアクセスする他の AP を特定し，その共有関係から，移送する資源であるか，決定する必要がある．本研究では，利用する資源をファイルに限定した場合について述べ，AP が利用している資源との共有関係を特定し，追跡する方法を提案する．

2.2 ファイルの共有関係に基づく移送モデル

前節で示した通り，移送する AP とその実行環境だけでなく，移送する実行環境にアクセスする他の AP と，その実行環境も状況に応じて一緒に移送しなければならない．本稿では，ファイル操作のみを対象とし，ファイルの共有関係に着目した移送対象の特定方法を示す．ファイル操作を行ったかどうかの監視については，open システムコールを監視することで実現できる．しかし，open システムコールがファイルに読み書きで open したとしても，対象ファイルに読み書きを行ったかは把握することができない．ここでは，open システムコールで監視した内容は，AP が該当の資源に対して，本当に操作を行っているか行っていないかに関らず操作を行ったと判断する．AP のファイル操作には，読み込みのみ (以降，ReadOnly と略す) と読み書き (以降，ReadOnly と略す) の二種類に分類できる．この操作種別により他の AP を移送するかどうかの判断を行う．

まず，移送対象の AP のみがアクセスしている場合について説明を行う．図 2 は AP1 のみがファイル X に対して open を行っているときである．この場合は，ファイル X に対して情報を書き込み，もしくは，読み込みを行っているので，動作に必要な資源である．よって，ファイル X を移送対象とする必要がある．

次に，移送対象である AP1 のみがファイル資源を利用しているのではなく，他の AP が AP1 が利用するファイル資源を利用している場合である．図 3 は AP1 と AP2〜APi がそれぞれファイル X にアクセスをしていた場合である．

(パターン 1) AP1 と AP2〜APi 全ての open が ReadOnly の場合．

AP1 と AP2〜APi はファイル X に対して読み込みを行っているだけで，ファイル X を通して AP1 と AP2〜APi は互いに影響を与えない．よって，AP1 とファイル X を移送する．しかし，ファイル X を移送した場合，移送元にある AP2〜APi が動作が出来なくなる可

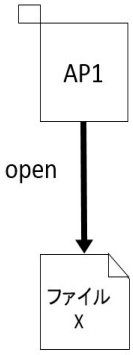

図 2 単独の AP のみが open している場合

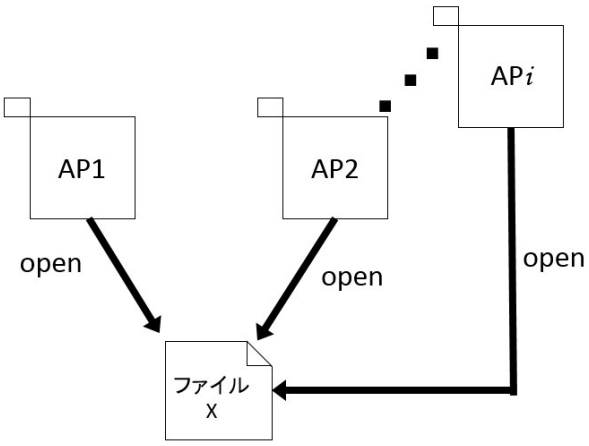

図 3 複数の AP が open している場合

能性があるため，ファイル X をコピーして残置する．

(パターン 2) AP1 の open が ReadWrite，AP2〜APi の open が ReadOnly の場合．

AP1 はファイル X に対して読み書きを行っている．AP2〜APi は読み込みを行っているので，AP1 が書き込んだ内容をファイル X を介して読み込み実行している可能性がある．従って，AP1 が AP2〜APi の動作に影響を与える可能性があるため，ファイル X と AP2〜APi を一緒に移送する．

(パターン 3) AP1 の open が ReadOnly，AP2〜APi のどれか一つ以上の open が ReadWrite の場合．

AP1 はファイル X に対して読み込みを行っている．AP2〜APi のどれか読み書きを行っている．AP2〜APi のどれかが書き込んだ内容をファイル X を介して AP1 が読込み実行している可能性がある．つまり，AP2〜APi のどれが書き込んだ内容が AP1 の動作に影響を与える可能性があるため，依存関係にあると言える．従って，ファイル X と AP2〜APi を移送する．

(パターン 4) AP1 と AP2〜APi の open が共に ReadWrite の場合．

AP1 と AP2〜APi は共にファイル X に対して読み書きを行っている．AP1 が書き込んだ内容をファイル X を介して，AP2〜APi が読み込み実行している可能性がある．また，AP2〜APi が書き込んだ内容をファイル X を介して AP1 が読み込み実行している可能性がある．従って，AP1 と AP2〜APi は共に依存関係があるため，ファイル X と AP2〜APi を移送する．

以上をまとめると，以下の 3 つのパターンに整理できる．

(1) 移送対象の AP が単独でファイルにアクセスを行っている場合，AP の操作種別に関係なくアクセスを行うファイルを移送対象とする．

(2) 移送対象の AP が open するファイルに，移送対象 AP と他の AP が ReadOnly のみで open している場合，移送対象の AP とアクセスを行うファイルの移送をする．この時，他の AP が動作可能とするため，ファイルのコピーを残置する．

(3) 移送対象の AP が open するファイルに，移送対象の AP もしくは他の AP が一つで ReadWrite で open している場合，移送対象の AP とアクセスを行うファイルの移送をする．アクセスを行う他の AP も移送対象に加えられる．

もし，他の AP が移送対象として追加された場合，他の AP を移送対象の AP として注目することで，この 3 つのパターンを適用でき解決できる．

3. 提案手法の実現

提案手法を実現するためには，AP が操作するファイルとその共有関係を把握する機能と，共有関係を追跡し，移送対象になる他の AP と，その実行環境を特定する機能が必要である．

3.1 ファイル操作と共有関係の把握方法

AP がファイルを操作する際は，操作するファイルを指定し open システムコールを発行する．ここでは，その open システムコールが発行するシステムコールを監視し，AP が利用しているファイルとその共有関係を把握する．UNIX 系 OS の open システムコールの仕様を示す．

int open(const char *pathname, int flags, mode_t mode);

この時，操作するファイルは open システムコールの引数である pathname により特定できる．ファイルとの共有関係は，flags を監視することによって特定できる．flags には 3 種類のフラグがある．

- O_RDONLY
- O_WRONLY
- O_RDWR

O_RDONLY が指定された場合は，ReadOnly に分類できる．O_WRONLY・O_RDWR が指定されている場合は ReadWrite として分類できる．

3.2 移送対象ファイルの追跡アルゴリズム

2.2 節で示した移送対象とするかの判断基準に基づき，共有関係の追跡を行うアルゴリズムを示す．追跡アルゴリズムは利用者が移送したい AP を探索対象として指定することで，開始される．

(ステップ 1) 探索対象の APi が open する全てのファイル Fj に対して以下の処理を実行する．

(ケース 1) 探索対象の APi が ReadOnly でファイル Fj を open していた場合

(a) 探索対象の APi のみがファイル Fj を open していた場合は，探索対象の APi と open しているファイル Fj を移送対象とする．

(b) 探索対象の APi が open するファイル Fj に対してアクセスしてる他の APk の全てが ReadOnly で open している場合は，探索対象の APi とファイル Fj を移送対象とする．ただし，ファイル Fj については，コピーを残置する．

(c) 探索対象の APi が open するファイル Fj に対してアクセスしている他の APk が 1 つでも ReadWrite でファイル Fj を open している場合は，探索対象の APi とファイル Fj を移送対象とする．また，APk を新たな探索対象とする．

(ケース 2) 探索対象の APi が ReadWrite でファイル Fj を open していた場合

(a) 探索対象の APi のみがファイル Fj を open していた場合は，探索対象の APi と open しているファイル Fj を移送対象とする．

(b) 探索対象の APi が open するファイル Fj に対してアクセスしている他の APk が 1 つでも ReadOnly か ReadWrite でファイル Fj を open している場合は，探索対象の APi とファイル Fj を移送対象とする．また，APk を新たな探索対象とする．

(ステップ 2) 新たに探索対象に加わった AP についてステップ 1 のアルゴリズムを適用し実行する．

上記のアルゴリズムの動作を図 4 に示す例を用いて説明する．AP1 と AP2 は，ファイル A を ReadOnly で，AP3 は ReadWrite で操作している．AP2 と AP4 はファイル B を ReadOnly で操作している．AP3 はファイル C を ReadWrite で，AP4 は ReadOnly で操作している．

(i) AP1 を利用者が移送したい AP とする時, AP1 と AP2 は ReadOnly で，AP3 は ReadWrite でファイル A と共有を行っている．よって，(ステップ 1) の (ケース 1) の (c) が適用され，ファイル A が移送対象となり，AP2 と AP3 は新たな探索対象となる．

(ii) AP2 と AP3 が新たに探索対象に加わったため，(ステップ 1) が適用される．

(iii) AP2 はファイル B を操作している．ファイル B は AP2 と AP4 からそれぞれ ReadOnly で共有を行っている．よって，(ステップ 1) の (ケース 1) の (b) が適用され，ファイル B はコピーすること明記して移送対象に加える．

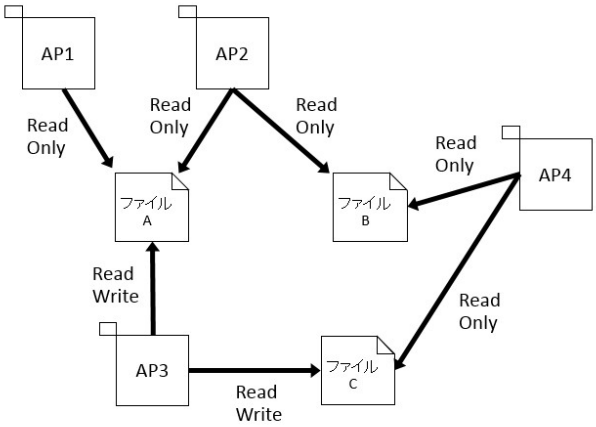

図 4 アルゴリズム適用例

表 1 評価環境

CPU	Intel Corei3-3240 3.4Ghz
メモリ	4GB
OS	FreeBSD9.2

(iv) AP3 はファイル C を操作している．ファイル C は AP3 から ReadWrite，AP4 から ReadOnly で共有されている．よって，(ステップ 1) の (ケース 2) の (b) が適用され，ファイル C を移送対象とし，AP4 を新たな探索対象となる．

(v) AP4 が新たに探索対象に加わったため，(ステップ 1) が適用される．

(vi) AP4 はファイル B を操作している．しかし，(iii) で追跡済みであるため，追跡を行わずに終了する．

4. 評価

評価を行った環境を表 1 に示す．利用するファイル資源と AP の対応関係を取得するために，open システムコールの監視を行った．このため，監視時のオーバーヘッドが，どの程度プログラムの実行時間に影響を与えるか，計測を行った．また，追跡アルゴリズムの特定時間を評価するために，AP 数が増加した場合の観点と，1 つのファイルにいくつのプロセスがアクセスを行っているかの共有度の観点から評価を行った．最後に，提案手法が実行する AP を特定することで，既存手法よりも移送時間が短くなることを示すために，Apache の実行環境の移送を対象に評価した．

4.1 open システムコール監視時のオーバヘッド

open システムコールを監視した時のオーバーヘッドを計測するために，実験用のファイル A と AP1 を 1 つずつ用意した．AP1 はファイル A に対してアクセスを行う操

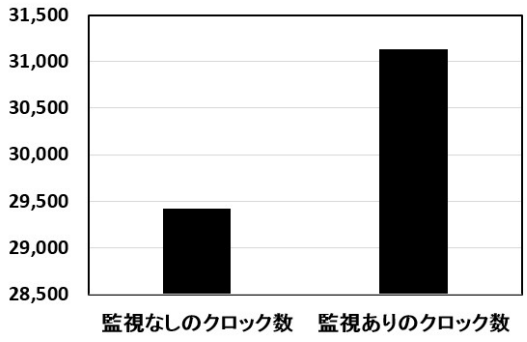

図 5 open システムコール監視時のオーバーヘッド

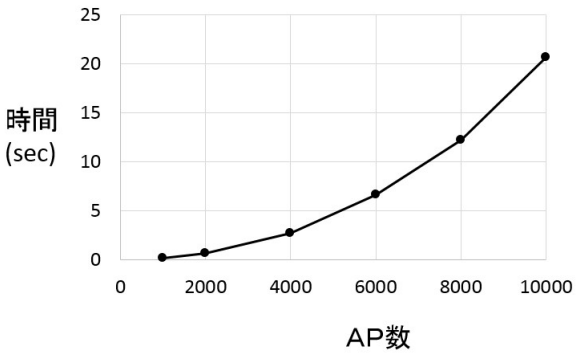

図 6 追跡アルゴリズムの実行時間

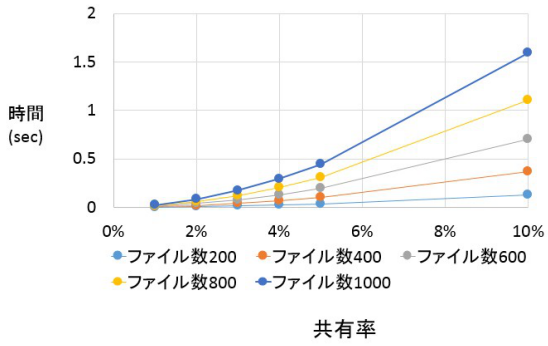

図 7 プロセス数 200 の場合の変化

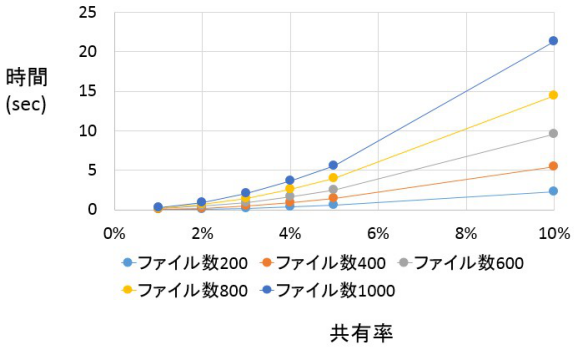

図 8 プロセス数 600 の場合の変化

作を行う．この時，open システムコールの監視ありと監視なし場合を，それぞれ 5000 回行い，平均を取った．あらかじめ，AP1 は測定前にファイル A に対して 1 回アクセスを行っておき，キャッシュヒットがある状態である．測定には Intel のアセンブラ命令である RDTSC(read-time stamp counter) を用いて，クロック数の取得を行い比較を行った．その結果を図 5 に示す．監視なしの場合だと 29,400 クロックであり，監視ありの場合だと 31,100 クロックである．監視ありのオーバヘッドは監視なしに比べて約 6 %増加であり，十分に小さい．

4.2 追跡アルゴリズムの評価

ここでは，AP の数を変化させた場合，AP 数によって追跡時間が，どのように変動するかの計測を行った．計測を行うために，評価用の AP を用意した．1 つの AP は，別のファイルに対して open を 5 回操作を行う．この時，AP 数を 2000，4000，6000，8000，10000 と変動させ，実行時間の測定を行った．追跡に要した時間を, 図 6 に示す．このグラフから，AP 数が 10000 の時，追跡時間が約 20 秒である．広域分散環境化における仮想マシンイメージの移送を考えた場合，回線種別にもよるが，数時間かかることが想定される．よって，20 秒という時間は無視できる時間であると考えられる．

4.3 ファイル共有率の評価

ここでは，各ファイルがいくつ AP からアクセスされているかを変化させた場合の評価を示す．具体的には，あるファイルが AP 全体の何%からアクセスされたかを示す「ファイル共有率」を可変にする．例えば，あるファイルにアクセスしている AP が 20 あり，全体の AP 数が 100 の場合，そのファイルの共有率は 20%である． ファイルの共有率を 1%から 10%の間で可変にし，評価した結果を図 7 と図 8 に示す．図 7 は，AP 数が 200，ファイル数を 200，400，600，800，1000 の場合について，ファイル共有率を変化させた結果である．図 8 は，AP 数が 600 の時の評価結果である．図 7 と図 8 から，ファイル共有率が高くなるにつれて，探索時間は長くなる．また，ファイル数が増えると，探索するファイルが増加するため，探索時間も長くなる．

今回の結果では，AP 数が 600，ファイル数が 1000，ファイル共有率が 10%のとき，約 21 秒の探索時間であった．実際のプロセス数は，数十から 200 程度であり，図 7 に示す数秒以内の探索時間と想定している．

4.4 仮想マシンと提案手法の比較

移送する AP と，その実行環境を特定することで，移送時間が短縮される．このことを示すために，提案手法と従来

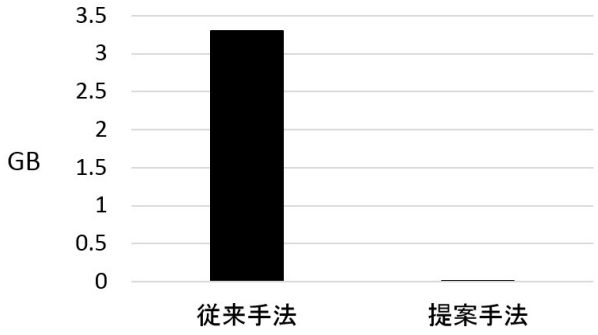

図 9 既存手法と提案手法の移送量の比較

表 2 既存手法と提案手法の移送時間の比較

通信帯域	既存手法	提案手法
64kbps	412,500sec	625sec
128kbps	206,250sec	312sec
1.5Mbps	17,600sec	26sec

手法との比較を行った．従来手法には，仮想マシンイメージを用いた．仮想マシンイメージは FreeBSD9.2 を OVF 形式にしたものを，移送する容量として定義した．提案手法は Apache2.2.29 をインストール時のページのみを配置し，起動，Web ページのアクセス，終了するまでに監視して得られる資源を使用する．また，提案手法では Apache を監視したときに得られるデータを用いて，追跡アルゴリズムの実行を行った．図 9 より，従来手法は 3.3GB，提案手法は 0.05GB となった．このことから従来手法よりも提案手法のほうが，移送量が減少していることが分かる．図 2 は移送にかかる時間を測定したものである．提案手法で Apache を監視した時に取得されたデータを用いて追跡アルゴリズムの実行時間の測定を行った結果，実行時間は 0.000166 秒 (約 0 秒) と非常に短い時間で特定することが出来た．これを元に移送時間が短縮しているかどうかを評価する．広域分散環境では 64kbps や 128kbps，1.5Mbps の専用線を用いられることが多い [8]．表 2 から，既存手法と提案手法を比べた場合，仮に 1.5Mbps の専用線を用いた場合，従来手法の移送時間は，17,600 秒である．提案手法の移送時間は 26 秒，追跡アルゴリズムの実行時間 0 秒を合わせると，26 秒となる．よって，移送時間が 99%短縮と大幅に短縮されることが分かる．

5. おわりに

広域分散システムを対象に，AP とその実行環境を効率的に移送する方法を提案した．本研究では移送時間を短縮するために，AP が利用する資源に着目を行い，他の AP からのアクセスを考慮した場合の，移送方法を示した．提案手法は，ファイルに限定した場合についての提案を行い，AP とファイルの依存関係を ReadOnly と ReadWrite の二つのパターンから関係性を明らかにした．それを元に移送する AP とその実行環境を特定する，追跡アルゴリズムの実装を行った．提案手法を用いて，Apache と仮想マシンイメージの移送時間を比較したところ，仮想マシンイメージを移送するに比べて，移送時間を 99%短縮する結果となった．また，移送時間も 1.5Mbps の時，26 秒なので，十分実用的な時間であった．

参考文献

[1] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield: Live Migration of Virtual Machines, Proceeding NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2,pp. 273-286(2005)
[2] 神崎康治，福島行信，村瀬勉，横平徳美，須田達也，“サーバ移動サービスにおけるサーバ追い出しアルゴリズム”，電気情報通信学会，IN2010-171，(2011 年).
[3] 小川 琢也，他，“仮想マシン群の効率的な配置に関する一検討”，電子情報通信学会，ICM2009-30，(2009 年)
[4] 橋本雅至，池辺実，岡本慶大，猪俣敦夫，藤川和利，“アプリケーションへの影響を考慮した動的なサーバ統合手法の提案”，電子情報通信学会，IA2012-68，(2012 年).
[5] 広渕崇宏，他，“仮想マシンの超広域ライブマイグレーションにむけたベストエフォート型状態同期機構の試作”，情報処理学会研究報告，2012-OS-121(11)，(2012 年).
[6] 直井由樹, 山田浩史, “GPU を活用した仮想マイグレーション”，情報処理学会研究報告，2014-OS-129(3)，(2014 年).
[7] 穐山空道, 広渕崇宏, 高野了成, 本位田真一, “ページキャッシュの復元による遠隔地ライブマイグレーションの高速化”，情報処理学会研究報告，2012-OS-123(9)，(2012 年)
[8] 一般社団法人 電気通信事業者協会，情報通信サービス利用状況，http://www.tca.or.jp/databook/pdf/2014chapter_2j.pdf, 最終閲覧日: 2015/8/6.