様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　元 年　 ６ 月 １１ 日現在

機関番号：１５１０１
研究種目：若手研究(B)
研究期間：2015～2018
課題番号：１５Ｋ１５９８２

研究課題名（和文）分散環境におけるモバイルエージェントの動的デバッグ手法に関する研究

研究課題名（英文）Research on Dynamic Debugging Methods of Mobile Agent Systems on Distributed Environments

研究代表者
東野　正幸（HIGASHINO, Masayuki）

鳥取大学・総合メディア基盤センター・助教

研究者番号：７０７３６０９０

交付決定額（研究期間全体）：（直接経費）　　3,100,000円

研究成果の概要（和文）：モバイルエージェントは自律性を持ったソフトウェア部品である。複数のエージェントを協調させることで人間社会に似たモデルで複雑な分散システムを構築できる利点がある。しかし、システムの規模が大きくなるにつれてエージェントの動作や協調関係が複雑化しデバッグが困難になる問題がある。本研究では、モバイルエージェントのデバッグの困難性を分析し、分散システムを構成するエージェント集合の分割と統合を自由に行ってもシステムが正しく動作することを保証するフレームワークを提案した。開発者がシステムを観測しながら自由に構成を変更可能とすることでデバッグの困難性を軽減できると考えられる。

研究成果の学術的意義や社会的意義
システム開発にはデバッグは必要不可欠であり、本研究はモバイルエージェント技術の実用化に大きく貢献できる。モバイルエージェントのデバッグの困難性を軽減することで、モバイルエージェント技術に基づく分散システムのデバッグに必要な作業工数の削減が期待できる。モバイルエージェント技術を実用化に近づけることで、大規模で柔軟性が要求される分散システムをさらに高度に発展させることにも貢献が期待できる。

研究成果の概要（英文）：A mobile agent is an autonomous software component. A mobile agent has the advantage that can construct complex distributed systems with models similar to human society. However, there is a problem that the distributed system constructed with mobile agent technology becomes difficult to debug belongs to the scale of the distributed system increases. This project surveys the difficulty of debugging a mobile agent system and proposes a framework which guarantees that the system works correctly even if the sets of agents are split and integrated freely. The difficulty of debugging of a mobile agent system can be reduced by freely dividing and concatenating the system with observing the performance of the system.

研究分野：情報ネットワーク

キーワード：モバイルエージェント　分散システム　デバッガ

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９、ＣＫ－１９（共通）

１．研究開始当初の背景

モバイルエージェントとはネットワークに接続された計算機間を移動できる自律的なソフトウェア部品である。複雑な分散システムを、モバイルエージェント同士の協調により実現することで、人間が理解しやすい人間社会のようなモデルで設計可能となることから、モバイルエージェントは分散システムの構築技術として有用性が示されてきた。しかし、モバイルエージェントは、自律的に計算機間を移動可能であることから、自律性に基づいた柔軟なシステム構築が可能となる反面、ネットワークやソフトウェアの規模が大きくなるにつれて、モバイルエージェントがどの計算機でどのような処理を行なっているのかを把握することが難しくなり、デバッグが難しくなる問題がある。

２．研究の目的

本研究はモバイルエージェントのデバッグの困難性を分析し、その困難性を軽減するための動的なデバッグ手法を開発することを目的としている。

３．研究の方法

近年の情報システムにおける分散環境のパラダイムにおいては、高い独立性を持つ低粒度のサービスを組み合わせて情報システムを構成するマイクロサービスとよばれる構築手法が急速に普及しつつある。

マイクロサービスは、モバイルエージェントに類似した性質をいくつか有しており、モバイルエージェントの動的なデバッグ手法を開発・適用する上で重要な応用対象となる。マイクロサービスの文脈においては、モノリスアーキテクチャからマイクロサービスアーキテクチャへシステムを分割したり逆に統合したりするといった構造変更の難しさが課題となっている。これは、分散環境におけるモバイルエージェントシステムでエージェント間の通信到達性を担保したままエージェントを分割・統合することと類似した課題である。

このことから、モバイルエージェント間の通信方法をプッシュ方式またはプル方式と同期方式または非同期方式を組み合わせた４パターンに分類し、これらのエージェント間における通信方式の制約下において、モバイルエージェントの集合をネットワーク上で分割・統合可能にすることで、モバイルエージェントシステムを既存のマイクロサービスの分散環境として扱う手法を提案する。これによりモバイルエージェントのデバッグの困難性を緩和することができると考えられる。

４．研究成果

(1) フレームワークの要件：分散システムにおいては、構成するサービスの粒度が小さく、数が多いほど、各サービスの仕様を把握して正確に実装するためのコストが高くなる。分散システムを分割・統合する場合、分割前と分割後に仕様を検証する手法として、大きく分けて静的検証と動的検証の手法があるが、静的検証は低粒度かつ多数のサービスから構成される分散システムにおいては計算量が大きくなる。このため、程粒度かつ多数のサービスから構成される分散システムを動的に分割・統合する場合、開発された仕様や実装を検証するのではなく、一定の制約下で、自由に動的な分割・統合を可能とする仕様や実装を開発することで、システムの構造変更に対するデバッグが容易となる。そこで、モノリシックサービスから動的にサービスを切り出し、それらをマイクロサービスとして動的に再構成する方法を開発する。この手法を実現するためには、そのフレームワークに要求される２つの性質がある。

① システムを自由に分割・統合するためには、モノリシックサービスまたはマイクロサービスの任意の部分を新しいサービスとして分割したり、任意のサービスを既存のサービス群に統合したりすることができ、かつシステムとして正しく機能することを担保したフレームワークが必要となる。これは、既存の仮想化技術を組み合わせることで実現できると考えられる。ただし、仮想化の粒度は、開発に使用するプログラミング言語の実行環境におけるインスタンスのレベルから物理マシンで動作する仮想マシンのレベルにまで及ぶ。多くのウェブサービスは３層アーキテクチャといった層型のシステム構成を採用しているが、これらは層ごとに個別の仕様や実装を持っている。サービスを自由に分割・統合する箇所が、異なる層を横断している場合に、複数の仕様や実装の整合性を確保する必要があり大きなコストとなる場合がある。このため、本フレームワークでは、既存の層で採用されている仕様や技術に依存しないシステム全体で統一的に利用できる制約を持っている必要がある。

② モノリシックサービスまたはマイクロサービスから新しくサービスを分割・統合した場合の性能変化を評価できることが必要になる。フレームワークによりシステムを分割・統合できたとしても、システムの性能は基盤となるハードウェアおよびソフトウェアの制約に依存するため、マイクロサービスアーキテクチャとしての優位性を分割によって得たとしても、

分割により性能が低下した場合には、低下した分の性能を向上させるための対策が必要になる。このため、本フレームワークでは、マイクロサービスをモノリスサービスから分割する際に、そのサービスを元のシステムに統合できることを担保した上で、サービス毎の統計情報の収集と分析を可能にすることで、システムの構造変更と性能の確保を両立する状況を動的に試行錯誤により発見可能にする必要がある。

(2) アーキテクチャの設計

① モバイルエージェントの構造:モバイルエージェントの内部データ構造を定義する。図 1 は、モバイルエージェントシステムのアーキテクチャを示す。ARE (Agent Runtime Environment) では、複数のエージェントが同時に動作し、ネットワークを介して ARE 間を移行できる。エージェントは、実行時状態領域とアプリケーション領域、およびプログラムコードを含むプログラムコード領域から構成される。実行時状態領域は、コールスタックやプログラムカウンタなどの処理を実行中のエージェントの状態に関する情報を持つ。アプリケーション領域は、エージェントの開発者によって設定された任意のデータを持つ。プログラムコード領域は、エージェントの動作を記述したプログラムコードを持つ。この構造は、OMG (Object Management Group) による MASIF 仕様および FIPA (Foundation for Intelligent Physical Agents) によるエージェント管理仕様と競合しない。

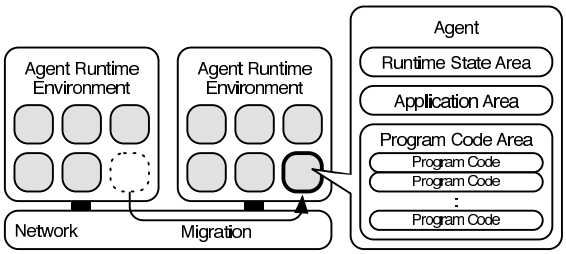

図 1 モバイルエージェントの構造

② エージェント間の通信方式:個々のエージェント間の通信方式として以下の 4 種類を定義する。

1. Push with Asynchronous : この通信方式は、送信元のエージェントから送信先のエージェントへ片方向にメッセージを送信する (図 2)。この方式は、メッセージが送信先に到達したという信頼性保証が不要な場合に使用する。
2. Push with Synchronous : この通信方式は送信元のエージェントから宛先のエージェントへ片方向のみにメッセージを送信する (図 3)。Push with Asynchronous との違いとして、メッセージが宛先に到達したという信頼性保証が必用な場合に使用する。
3. Pull with Asynchronous : この通信方式は、送信元エージェントから送信先エージェントへ要求メッセージを送信し、送信元エージェントは送信先エージェントからの応答メッセージを待たない (図 4)。
4. Pull with Synchronous : この通信方式は、送信元エージェントから送信先エージェントへ要求メッセージを送信し、送信元エージェントは送信先エージェントからの応答メッセージを待つ (図 5)。

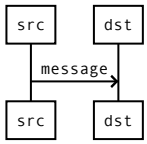

図 2 Push with Asynchronous

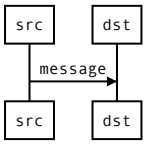

図 3 Push with Synchronous

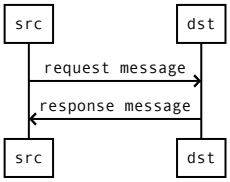

図 4 Pull with Asynchronous

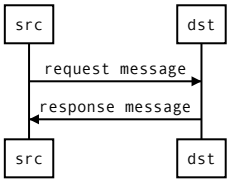

図 5 Pull with Synchronous

(3) フレームワークの設計

① プロセスの分割と統合:図 6 にプロセスの分割と統合の概念図を示す。開発者は任意のプロセスを任意の数プロセスに分割し、他のノードにプロセスを移動できる。プロセスが他のノードに移動された場合、ローカル通信とリモート通信の切り替えはフレームワークを介して透過的に行われる。一般に、通信遅延は、ローカル通信よりもリモート通信の方が大きく、システムの分散化による冗長性とリモート通信による通信遅延との間にはトレードオフがある。システムの構成を変更する際には、ドメインの性能の品質のバランスを取ることが重要となる。このアーキテクチャでは、プロセスが任意のノードに移動されても、プロセス間通信はローカルでもリモートでも維持される。これにより、通信の種類を意識することなく、プロセスの分散と性能のバランスを取れる境界を開発者が動的に変更しながら発見しやすくなる。この特性はドメイン駆動による開発において重要であり、より良いドメインを見つけることを容易にし、システムの変更容易性とデバッグの容易性を改善することができる。

② データの分割と統合：図 7 にデータの分割と統合の概念図を示す。開発者は、任意のリソース（データ）を複製したり任意の数に分割したりすることができる。本フレームワークは、分割されたリソース間の接続と、分割されたリソースの検索性を担保する必要がある。分割されたリソースがマスタスレーブモデル、マルチマスターモデル、分散合意プロトコルのための Paxos などのいずれの手法をとるかは、フレームワークの１つ上の層によって実現される。このアプローチでは、プロセスの分散とデータの分散を自由に行えるようにすることで、開発者はプログラムのドメイン設計とデータの整合性の保証方法を分離して考えることが可能となり、ドメインの構造変更の容易性とデータの整合性に関するデバッグの容易性を両立可能とする。

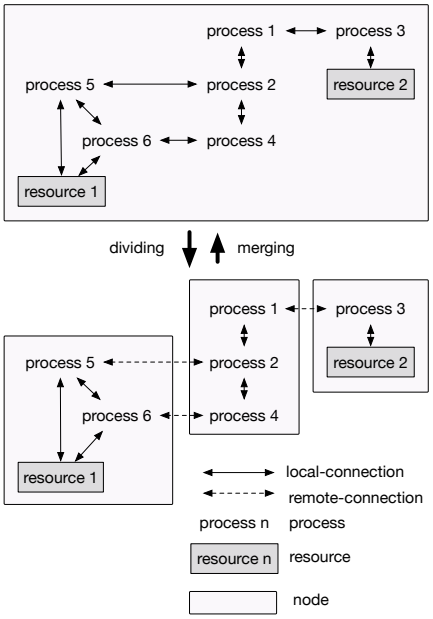

図 6 プロセスの分割と統合

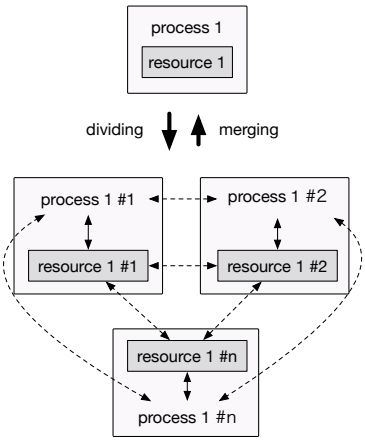

図 7 データの分割と統合

５．主な発表論文等

〔雑誌論文〕（計 2 件）

① Masayuki Higashino、Toshiya Kawato、Takao Kawamura、A Design with Mobile Agent Architecture for Refactoring a Monolithic Service into Microservices、Journal of Computers、査読有、13 巻、10 号、2018、pp. 1192-1201、DOI：10.17706/jcp.13.10.1192-1201
② Masayuki Higashino、Toshiya Kawato、Takao Kawamura、A Design for Application of Mobile Agent Technology to MicroService Architecture、International Journal of Computer, Electrical, Automation, Control and Information Engineering、査読有、12 巻、2 号、2018、pp. 53-57

〔学会発表〕（計 6 件）

① 東野 正幸、モノリシックサービスからマイクロサービスへの構造変更を容易化するためのウェブアプリケーションフレームワークの検討、情報処理学会研究報告、2018-DPS-174 巻、34 号、pp. 1-4、2018
② Masayuki Higashino、Application of Mobile Agent Technology to MicroService Architecture、In Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services、pp. 526-529、2017
③ 東野 正幸、川村 尚生、モバイルエージェント技術によるウェブサービスのマイクロサービスアーキテクチャ化に関する一考察、合同エージェントワークショップ＆シンポジウム 2017 予稿集、pp. 344-345、2017
④ 東野 正幸、川村 尚生、論理的モバイルエージェントフレームワークの開発とマイクロサービスアーキテクチャとの調和、合同エージェントワークショップ＆シンポジウム 2016 予稿集、pp. 235-236、2016
⑤ 東野 正幸、灘本 拓、高橋 健一、川村 尚生、菅原 一孔、モバイルエージェントシステムのデバッグに関する研究の展望、合同エージェントワークショップ＆シンポジウム 2015 予稿集、pp. 185-186、2015
⑥ 灘本 拓、高橋 健一、東野 正幸、川村 尚生、菅原 一孔、モバイルエージェントシステムのためのエージェント検索機能の提案、合同エージェントワークショップ＆シンポジウム 2015 予稿集、pp. 187-188、2015