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検討

Consideration of Search Function for Mobile Agent System

灘本 拓 † 尾崎 槙 † 東野 正幸 †† 高橋 健一 † 川村 尚生 † 菅原 一孔 †

Taku Nadamoto † Shin Osaki † Masayuki Higashino ††

Kenichi Takahashi † Takao Kawamura † Kazunori Sugahara †

† 鳥取大学大学院 工学研究科　†† 鳥取大学 産学・地域連携推進機構

1　はじめに

モバイルエージェントシステムとはネットワークに接続されたコンピュータであるノード間をエージェントと呼ばれるプログラムが移動しながら処理を行うシステムである．モバイルエージェントには分散処理における通信遅延および通信回数の削減，負荷分散，耐故障性の向上などの利点があり，これらの利点より分散処理システムの構築に有効な技術として注目され，様々な研究が行われてきた [1, 2]．しかし，モバイルエージェントシステムを利用したシステムが一般的に普及しているとは言い難い．その原因のひとつとしてモバイルエージェントシステムのデバッグが困難であることが挙げられる．我々はこの問題を解決するためモバイルエージェントシステムのためのデバッガを開発している [3]．このデバッガの機能の一つにエージェントの検索機能がある．モバイルエージェントは遠隔地のノードに移動して動作するため，デバッグの際にデバッグ対象の実行状態の把握が困難になる．検索機能はこのような遠隔地で動作しているエージェントの位置や状態などの情報を取得することで効率的なデバッグを可能にする．しかし，現状の検索機能では単体や同名のエージェントの検索，エージェントの状態の検索しか行えず，移動によって複雑に動作するモバイルエージェントのデバッグをより効率的に行うには複数のエージェントの位置や実行状態などを様々な状況に応じて整理・絞込みを行える，より詳細な検索が必要になる．そこで本研究ではモバイルエージェントシステムのデバッグの問題点を提起し，問題点を解決するためのエージェントの検索機能を提案する．

2　モバイルエージェントの移動

モバイルエージェントシステムではエージェントと称されるプログラムが複数の計算機に分散し，計算機間を移動しながら処理を行う．開発者は実行したエージェントの位置や状態を確認しながらシステムのデバッグを行う．しかし，多数のエージェントが複数の計算機に分散して動作しているため，監視すべきエージェントの数が多く，またエージェントは遠隔地において自律的に移動先を変えながら移動するため，個々のエージェントは常に位置や状態を変化しながら動作する．また，エージェントの移動は他の様々な要素に影響する．例えば，エージェントの移動の失敗はその後の処理や入出力，他のエージェントとの協調動作などに影響を及ぼす．このため，エージェントの実行状況が複雑に変化し，システム全体の把握がより難しくなる．

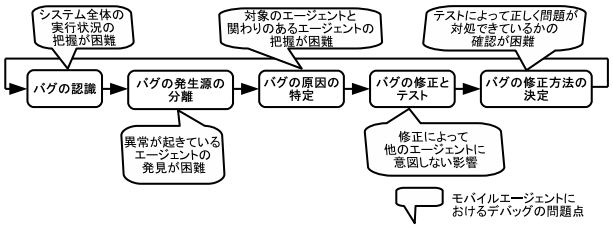


図 1: モバイルエージェントシステムのデバッグにおける問題点

3　モバイルエージェントシステムのデバッグの問題点

モバイルエージェントはエージェントが遠隔地に移動，分散して動作することからデバッグ対象が移動してしまいデバッグの際様々な問題点が生じる．図 1 に一般的なデバッグ手順を示し，以下にそれぞれのデバッグ手順の問題点を述べる．

1. バグの認識

システムにバグが生じた場合，システムの不具合やエージェントの異常動作がおきる．開発者はこれらの異常に気づくことでバグを認識するが，モバイルエージェントシステムは複数のエージェントが遠隔地に移動して動作することから，エージェントの動作状況を常に適切に把握するのが難しく，バグの発見が遅れる原因になる．

2. バグの発生源の分離

モバイルエージェントシステムのデバッグではまずバグの発生源であるエージェントを特定する必要がある．そのためにはバグの発生源を類推し，可能性のあるエージェントをひとつずつ確かめる必要がある．モバイルエージェントシステムは複数のエージェントによって成り立っており，分散して動作している多数のエージェントの中から可能性のあるエージェントを見つけ出し，実行状態や位置を確認しなければならない．しかし，これらのエージェントは遠隔地で動作しているため，特定のエージェントを探しだし調べるのは手間を要する．

3. バグの原因の特定

バグの原因が異常な動作をしているエージェント自身にあるとは限らない．エージェントの協調動作や入出力によって他のエージェントに影響が及び，バグの発見箇所とは別の場所に原因が存在する可能性がある．これらの原因はエージェントの移動性にあり，エージェントが移動することで様々なノードやエージェントなどに副作用を生じさせ，バグの原因が複雑化する．

4. バグの修正方法の決定

バグの原因を特定した後にその修正を行う．ただしその修正がシステムに意図しない影響を与えてしまい，新たなバグが発生する原因になることもある．特にモバイルエージェントシステムの場合，エージェント同士が協調して動作しているため，一部の変更による影響が他のエージェントにも及ぶことが考えられる．しかし，エージェントは各ノードに分散して自律的に動作しているため，エージェント同士の協調関係が把握しにくく，修正方法の決定が難しくなる．

5. バグの修正とテスト

ソースコードを修正し，テストする．テストする際にはその修正が問題に正しく対処しているか，その修正によって他の部分に望ましくない影響を与えていないかを確認する必要がある．

4 検索機能の検討

モバイルエージェントのデバッグは遠隔地に分散して動作しているエージェントの実行状態や位置を確認することで行う．多数動作しているエージェントの情報をその状況に応じて適切に把握するための検索機能を検討する．

4.1 モバイルエージェントの検索

前述したデバッグにおける問題点に対し，どのように対応するかを検討する．

1. バグの認識

モバイルエージェントシステムは複数のエージェントが遠隔地に分散して動作するため，すべてのエージェントの動作状況を適切に把握するのは難しい．この問題に対して，各ノードで動作しているエージェントの状態を開発者の手元に集約し，開発に応じてエージェントの状態を確認する必要がある．そこで，単体のエージェントや同種のエージェントを検索することで，各エージェントの動作状況を確認しながら開発を行うことができ，開発時のバグの発見が早まる．

2. バグの発生源の分離

遠隔地で動作しているエージェントの中から，動作や位置に間違いのあるエージェントを探し出さなければならない．しかし，これらのエージェントは遠隔地で動作しているため，特定のエージェントを探し出すことが難しい．この問題に対して，各ノードから必要なエージェントの情報を取得する必要がある．そこで，単体のエージェントや同種のエージェントを検索することで容易にエージェントを探し出し，動作や位置を確認できる．また，移動経路の検索を行うことで，エージェントの移動状況を確認でき，間違った移動を行っているエージェントの発見が可能になる．

3. バグの原因の特定

エージェントの協調動作や入出力によって，他のエージェントに影響が及び，バグの発生源とは別の場所にその原因が存在する可能性がある．しかし，エージェントは移動し実行状態が複雑になるためバグの発生源であるエージェントと関連しているエージェントを探すのは難しい．この問題に対して，バグの発生源であるエージェントと関わりのあるエージェントの情報をノードから取得する必要がある．そこで，エージェントの訪問履歴を検索し，どのエージェントが関連しているかを探すことで，バグの原因の特定が容易になる．また，指定した処理を実行したエージェントを検索することでバグの原因となる処理を行ったエージェントを探し出すことができる．

4. 修正方法の決定

エージェント同士が協調して動作しているため，修正によって他のエージェントに予期せぬ影響が及ぶ可能性がある．この問題に対して，バグの修正を行うエージェントと関わりのあるエージェントの情報をノードから取得する必要がある．そこでエージェントの訪問履歴を検索することで，修正するエージェントと関連しているエージェントを探し，修正によって望ましくない影響が及ばないか確認することができる．また，指定した処理を実行したエージェントを検索することで，同じような処理を行っているエージェントを探し，同様のバグが含まれていないか確認することができる．

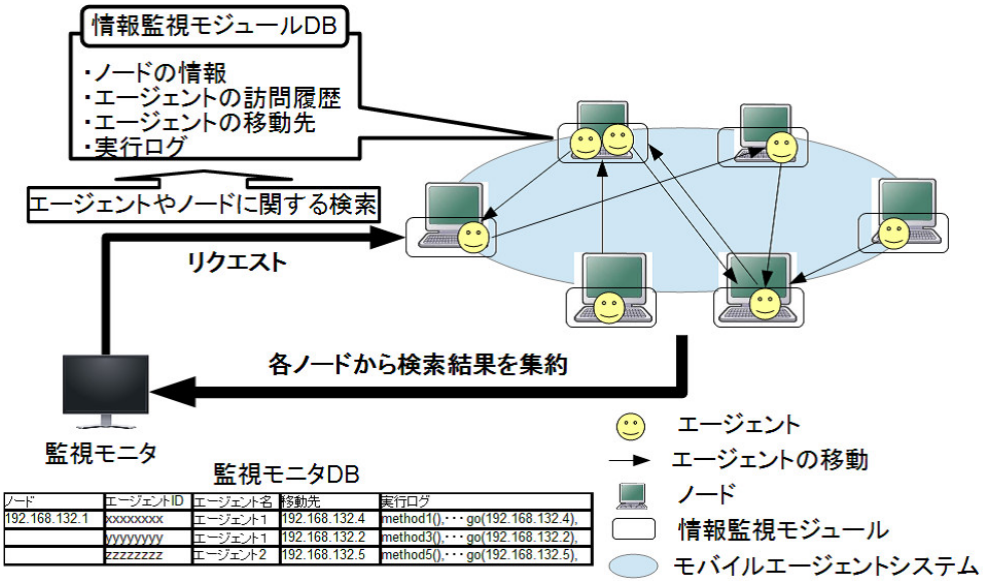
5. バグの修正とテスト

修正によって他の部分に望ましくない影響を与えていないか確認する必要がある．しかし，エージェントは遠隔地に分散して動作しているため，すべてのエージェントの動作状況を適切に把握するのは難しい．この問題に対して，テストを行ったエージェントの情報をノードから取得する必要がある．そこで，単体のエージェントや同種のエージェントの検索を行うことで，エージェントの動作状況を容易に把握することができ，テストが成功しているか確認することができる．

これらの問題に対応する検索方法を以下に示す．

特定のエージェントの検索

個別のエージェントに付与されたエージェント固



ノード	エージェントID	エージェント名	移動先	実行ログ
192.168.132.1	xxxxxxxx	エージェント1	192.168.132.4	method1(),・・・go(192.168.132.4),
	yyyyyyyy	エージェント1	192.168.132.2	method3(),・・・go(192.168.132.2),
	zzzzzzzz	エージェント2	192.168.132.5	method5(),・・・go(192.168.132.5),

図 2: 検索機能の概要

有の ID を検索することで特定のエージェントの情報を取得する．

同種のエージェントの検索
同じ属性を持つエージェントをエージェント名で検索することで同種のエージェントの情報をすべて取得する．

移動経路の検索
エージェント名やエージェント ID とそのエージェントの移動経路を検索することで，各ノードからエージェントの移動先を取得し，それをエージェントの移動順に整理することでエージェントの移動経路を取得する．

ノードの検索
ノードの IP アドレスとポート番号で検索し，指定したノードのエージェント訪問履歴を時系列順に取得する．

処理の検索
エージェントの実行ログで検索し，各ノードのログからその処理を実行したエージェントを取得する．

4.2 検索機能の実装方法

デバッグ対象のエージェントを指定するためには，すべてのエージェントを一意に識別できる必要がある．そのため各エージェントに対して UUID（Universally Unique Identifier）[4] を振り分け，個々のエージェントを識別するためのエージェント ID を付与する．図 2 に検索機能の概要を示す．各ノードには情報監視モジュールを付与し，そのノードで起きたイベントを監視する．情報監視モジュールによって，訪れたエージェントの名前と ID，移動先，実行ログを取得し，そのノードのデータベースに記録する．ユーザが検索を行う際，必要に応じた検索クエリを入力する．入力が完了すると監視モニタから各ノードに対してリクエストと共にエージェントやノードに関する検索クエリが送られる．各ノードは送られてきた検索クエリを用いてデータベースの情報を検索し，検索結果を監視モニタに送信する．監視モニタは各ノードから送られてきた情報を監視モニタのデータベースに集約し，ユーザに提示する．

5 おわりに

本研究ではモバイルエージェントシステムの検索機能を検討した．今後は今回検討した検索機能の実装と評価を行う．

参考文献

[1] A. R. Hurson, E. Jean, M. Ongtang, X. Gao, Y. Jiao., and T. E. Potok. *Recent Advances in Mobile Agent-Oriented Applications*, pages 106–139. John Wiley & Sons, Inc., 2010.

[2] A. Outtagarts. Mobile Agent-based Applications : a Survey. *International Journal of Computer Science and Network Security (IJCSNS)*, pages 331–339, 2009.

[3] M. Higashino, S. Osaki, S. Otagaki, K. Takahashi, T. Kawamura, and K. Sugahara. Debugging Mobile Agent Systems. In *Proceedings of the 15th International Conference on Information Integration and Web-based Applications and Services (iiWAS 2013)*, pages 667–670, 2013.

[4] M. Mealling P. Leach and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. *Request for Comments (RFC) 4122*, 2005.