# 大規模マルチエージェントシミュレーション環境のための クラスタ機構

## A Clustering Mechanism for Massive Multi-Agent based Simulation

**山本　学**[†]　　**田井　秀樹**[††]　　**水田　秀行**[††]
Gaku Yamamoto　　Hideki Tai　　Hideyuki Mizuta

## 1. はじめに

マルチエージェントシミュレーション(Multi-Agent base Simulation, MABS)は，人間の行動モデルを取り込んだシミュレーションのように数式モデルでは表現しにくい対象のシミュレーションで注目されている．しかしながら，多くの MABS ではエージェント数が数千程度であり，多数のエージェントを用いる大規模なシミュレーションへの適用が困難である．我々は 1999 年に数十万を超えるエージェントを管理する基盤技術[1]の研究を開始し，2005 年にこの技術を MABS へ適用し，数十万，数百万を超える数のエージェントを用いた大規模マルチエージェントベースシミュレーション(Massive Multi-Agent based Simulation, MMABS)を可能にする技術を開発している[2]．このような MMABS では性能が非常に重要となる．比較的単純なオークションシミュレーションでも，エージェント数が百万体となると全体の処理時間は 100 時間を越える場合もありえる．この時間を短縮するには，シミュレーションを複数のプロセッサ，または，計算機を用いたクラスタで実行することが要求される．MMABS で，このようなクラスタ機構を構成するための鍵となる基本技術は，エージェントプロキシ，メッセージング，時間管理，エージェントの移動である．本稿では，これらの技術に関して，我々が開発している Java 上での MMABS 基盤「ZASE」を例にして述べる．

## 2. 背景

多くの MABS では，エージェントの数が少ない状況で行なわれている．しかしながら，シミュレーションの結果が，エージェントが数十体の場合と数十万体では結果が異なる場合がある．

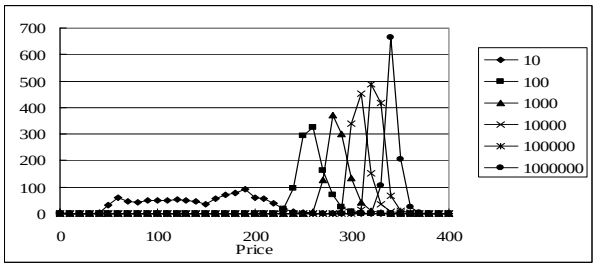


図 1　大規模オークションシミュレーションの最終落札額の度数分布

図 1 は，オークションシミュレーションで，エージェント数を 10 体から百万体まで変化させた場合の最終落札額とその度数分布を示したものである[2]．図 1 の結果は入札者数がオークションの結果に大きく影響を与えているということを示している．図 2 に 100 万体のエージェントを利用した場合の度数分布を得るのに要する時間の推測値と計算機台数の関係を示す．この図から処理性能の重要性とクラスタ機構の必要性がわかる．

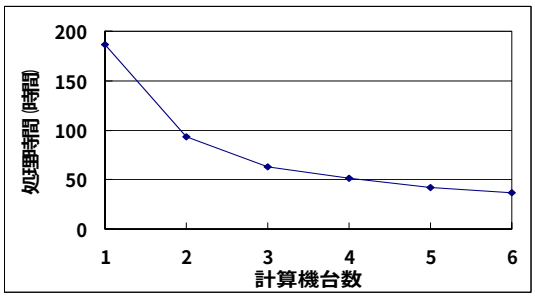


図 2　大規模オークションシミュレーションの計算機台数と処理時間

また，近年 MABS の交通シミュレーションへの適用も試みられている．MMABS の技術を用いて都市レベルのシミュレーションを行なうと，数十万から数百万体のエージェントが必要となる．例えば，京都市規模の道路網をシミュレーションする場合，我々の実験では，1 つのプロセッサを使うと実時間の 10 秒を 1 シミュレーション時間としても 1 時間分のシミュレーションを行なうのに 1 時間以上を要することが得られている．

## 3. MMABS の概要

本稿で議論する技術は，MMABS の基盤となる技術である．MABS には，オークションシミュレーション，交通シミュレーション，非難誘導シミュレーション，感染症シミュレーションなど様々なタイプがあるが，本稿で考える基盤とは，これら様々なタイプのシミュレータを構築するための基盤である．したがって，提供する機能は MMABS を実現するための基本的な機能である．

大規模化のための基本機構となるものは，一つのプロセスで数万から数十万のエージェントを活動させる機構[1]と複数計算機から構成されるクラスタ化されたシミュレータを構築するための機構である．図 3 にクラスタ化されたシミュレータの概要図を示す．シミュレータは，シミュレーション実行環境とエージェント実行環境から構成される．

前者は，エージェントが活動する環境のシミュレーションを行なう．例えば，オークションシミュレーションでは，入札処理のシミュレーションである．後者は，エージェントが実際に置かれる環境である．エージェントはシミュレーション実行環境が発生するイベントを受けて，意思決定を行い，その結果をシミュレーション実行環境に通知する．シミュレーション実行環境とエージェント実行環境は同じプロセス上に置かれる場合もあれば，異なるプロセス上に

† 日本アイ・ビー・エム（株）東京基礎研究所，
京都大学 社会情報学専攻
†† 日本アイ・ビー・エム（株）東京基礎研究所

置かれることもある．また，シミュレーション実行環境ならびにエージェント実行環境もそれぞれ複数のプロセスに分割されることもある．我々は典型的な分割パターンは次の２つであると考えている．

1. １つのシミュレーション実行環境プロセスと複数エージェント実行環境プロセス
2. シミュレーション実行環境とエージェント実行環境を同じプロセスに載せ，シミュレーション空間全体を複数のプロセスに分割する

1 はオークションシミュレーションのように，シミュレーション全体を調停するものが一箇所に存在する場合である．RoboCup-Rescue シミュレータも「カーネル」がシミュレーション実行環境であり，そこからエージェント群に情報を提供するという形態であり，1 の形態となっている[3]. 2 は交通シミュレーションのようにエージェントが地理的に分散し，シミュレーション空間全体を地理的に分割できる場合である．

実行環境を結合する鍵となる基本機構は，エージェントプロキシ，メッセージング，時間管理，エージェントの移動である．以降これらに関して述べる．

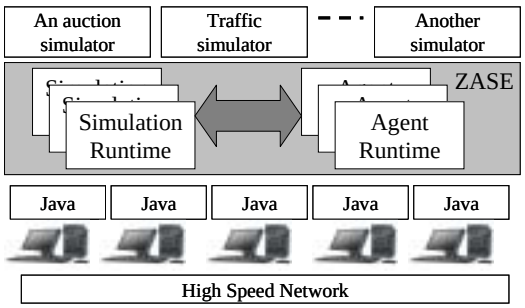


図３　クラスタ化されたシミュレータの概要

## 4. エージェントプロキシ

MMABS では，シミュレーション実行環境が生成したイベントに対して，各エージェントがアクションをとることで，シミュレーション全体が進行する．エージェントのアクションには，イベントに対して機械的に反応するアクションと何らかの意思決定が必要なアクションがある．例えば，交通シミュレーションの場合では，ドライバの振る舞いをエージェントが行なうことになるが，通常の運転では，進行速度は前方車両の速度や道路の制限速度から反応的に決定される．これに対し，交差点における進路決定は，ドライバの意思決定が必要である．

一方，シミュレーション実行環境からエージェントを呼び出す場合，メッセージ機構を用いる．シミュレーション実行環境がメッセージを生成し，そのメッセージをエージェントに送信し，そのエージェントが処理を行い，結果をメッセージとしてシミュレーション実行環境に返送する．この一連の処理のオーバーヘッドはメソッド呼び出しに比べ大きい．シミュレーション実行環境とエージェント実行環境が分離されている場合は，通信を伴うので，さらにオーバーヘッドが大きくなる．

このため，シミュレーション実行環境の状況によって反応的に決定可能なものは，メッセージ機能によりエージェントを呼び出すのではなく，エージェントの代理としてのエージェントプロキシをシミュレーション実行環境内に置き，シミュレーション実行環境からエージェントプロキシをメソッド呼び出す方が効率的である．この分離は，エージェントの意思決定と反応的処理を明示的に分離し，エージェントは意思決定のみを行なうという意味もあり，プログラミング・モデルの観点から見ても理にかなっている．

## 5. メッセージング

メッセージには，エージェント間メッセージ，シミュレーション実行環境とエージェント間のメッセージ，シミュレーション実行環境間メッセージがある．エージェント間メッセージは，クラスタ環境とは関係なく必要な機能であり，ほとんどのエージェントシステムで実装されている．また，シミュレーション実行環境間のメッセージは，宛先を特定した非同期メッセージと全シミュレーション実行環境に送信するマルチキャストメッセージで十分であり，特殊な形態ではない．一方，シミュレーション実行環境とエージェント間のメッセージに関しては，プログラミングの行ないやすさと性能の点から宛先特定メッセージやマルチキャストメッセージだけでは十分ではない．本稿ではシミュレーション実行環境とエージェント間のメッセージ機構を述べる．

### 5.1 シミュレーション実行環境とエージェント間のメッセージ

シミュレーション実行環境は，適宜イベントを生成し，エージェントはそのイベントを受けてアクションを実行する．このアクションには，エージェント内部の状態だけに影響を与えるものと，アクションの結果シミュレーション実行環境の状態に影響を与えるものがある．このため，シミュレーション実行環境からエージェントへメッセージを送信する機構としては，非同期メッセージと同期メッセージがある．非同期メッセージとは，シミュレーション実行環境が送信したメッセージに対してのエージェントからの返答メッセージを待たないものである．同期メッセージとは，シミュレーション実行環境が送信したメッセージに対するエージェントからの返答メッセージを待つものである．

一方，シミュレーション実行環境がエージェントにメッセージを送信する場合，宛先を特定して送信する宛先特定メッセージと，宛先を特定しない宛先不特定メッセージがある．宛先不特定メッセージの宛先は，エージェント実行環境により決定される．ZASE の場合では，エージェント実行環境にあらかじめ登録されている Resolver オブジェクトにより決定される．Resolver オブジェクトは，宛先となるエージェントの識別子のリストを作成する．エージェント実行環境は宛先不特定メッセージをこのリスト内のエージェントに渡す．例えば，オークションシミュレーションでは，オークションの最新価格を決定するシミュレーション実行環境が最新の価格をエージェントに送信し，エージェント実行環境にある Resolver オブジェクトが宛先を決定する．

同期メッセージと非同期メッセージは，宛先特定メッセージならびに宛先不特定メッセージの両方にある．ここで特殊なメッセージ形態が，宛先不特定の同期メッセージである．宛先不特定メッセージでは，エージェント実行環境の Resolver オブジェクトが宛先リストを生成する．この場合，それぞれのエージェントが返答メッセージをシミュレーション実行環境に送信するため，返答メッセージは複数

になる．シミュレーション実行環境は，送信されてくる返答メッセージを順次取得する．ZASE の場合，シミュレーション実行環境が使用する同期型宛先不特定メッセージ送信用アプリケーションプログラミングインタフェース(API)の戻り値として，java.util.Iterator オブジェクトが戻される．シミュレーション実行環境は，java.util.Iterator#hasNext()メソッドを用いて，次の返答メッセージの受信を確認する．このメソッドは，もし次の返答メッセージを受信済みであれば true を返し，また全ての返答メッセージを受信していないのであれば，次の返答メッセージの受信までして待ち，受信後に true を返す．これ以上返答メッセージがない場合は，即座に false を返す．シミュレーション実行環境は，java.util.Iterator#next()により，次の返答メッセージを取得する．

### 5.2 メッセージ集約機能

同期型宛先不特定メッセージの場合，宛先となるエージェントが大量になる．大量の返答メッセージがネットワークを介して送信されると性能が極端に低下する．我々の実験では，ネットワークを介してメッセージを送信する場合，毎秒 3000～5000 メッセージ程度しか送信できないことがわかっている．したがって，このメッセージを削減するために返答メッセージを集約する機構が必要である．シミュレーション実行環境は，すべての返答メッセージを受信した後に，何らかの集計処理を行ない，結果を計算する場合がある．オークションシミュレーションでは，シミュレーション実行環境が送信する最新価格情報を持つメッセージに対して，各エージェントが次の入札額を返答メッセージとして返送する．これを受けたシミュレーション実行環境は，全ての返答メッセージの中から次の価格を決定する．例えば，2 番目に高い入札額である．複数のエージェント実行環境が存在する状況では，シミュレーション実行環境は，各エージェント実行環境から最高入札額と 2 番目に高い入札額の情報だけを受信すれば，全体の中での 2 番目に高い入札額を決定できる．つまり，各エージェント実行環境に返答メッセージを集約する機能を入れることで，そのエージェント実行環境内の全返答メッセージから最高入札額と 2 番目に高い入札額を決定して，一つのメッセージとしてシミュレーション実行環境に返送することができる．ZASE の場合，シミュレーション実行環境は，同期型宛先不特定メッセージを送信する場合，返答メッセージの集約処理を行なう Aggregator オブジェクトを引数として渡すことができる．このオブジェクトは，メッセージとともにエージェント実行環境に送られ，返答メッセージの集約処理を行なう．

## 6. 時間管理

シミュレーションでは時間管理は重要な機能である．複数の実行環境から構成されるシミュレータでは，それらの間で時間の同期を行なう．エージェントシミュレーションでは，エージェントはシミュレーション実行環境からのイベントを受信して処理を行なう．そのため，時間管理はシミュレーション実行環境が行なう．シミュレーション実行環境が複数ある場合は，それらのどれかがマスターとなって時間管理を行なう．他の実行環境はマスターの時間に同期する．時間同期機構として，シミュレーション実行環境間時間同期機構とシミュレーション・エージェント実行環境間時間同期機構がある．

前者の機構では，マスターがある時刻の開始を全シミュレーション実行環境に送信することで，その時刻の処理が開始される．各シミュレーション実行環境がその時刻の処理が完了したらマスターに通知し，マスターは全シミュレーション実行環境からの完了通知を受けたら，全シミュレーション実行環境に時刻の完了を通知する．この一連の処理はそれ程複雑なものではない．

一方，シミュレーション実行環境とエージェント実行環境の時間同期はそれ程簡単なものではない．エージェント実行環境での，あるシミュレーション時刻の処理の完了の判定は，その時刻内に処理されるべきメッセージの処理が全て完了したかどうかで判定される．しかしながら，エージェントは他のエージェントにメッセージを送信する．ここで問題となることが，このエージェントのメッセージがいつ処理されるべきかということである．これは 1 シミュレーション時間が現実世界でどのくらいの時間に相当するか，ならびに，そのメッセージが現実世界でどのようなものに対応するかに依存する．例えば，人同士が直接会話している状況をメッセージで再現する場合は，同じシミュレーション時刻で処理されるべきである．一方，電子メールでの対話を再現するのであれば，そのメッセージを送信した後の適当なシミュレーション時刻で処理されるべきである．このようにエージェントが送信したメッセージをいつ処理すべきかは，アプリケーションに依存する．したがって，時間同期機構は，メッセージを処理するタイミングを固定的にするのではなく，いつ処理すべきかをアプリケーションで制御できるようにすべきである．

ZASE の場合，規定の振る舞いとして，あるシミュレーション時刻内でエージェントが送信した全メッセージの処理が完了したら次のシミュレーション時刻へ遷移する．メッセージを送信したシミュレーション時刻以降にそのメッセージを処理する必要がある場合，アプリケーションは該当するシミュレーション時刻になるまでメッセージを一時的に蓄積するコンポーネントを開発すればよい．

さらに，ZASE では，オプションの機構として，次の 3 点も提供する．

- 規定の振る舞いを停止する
- エージェントのメッセージ送信，メッセージ処理完了をモニタできるオブジェクトをエージェント実行環境に登録する
- エージェント実行環境に対して，次の時間に遷移してよいことを通知する

アプリケーションは，該当するシミュレーション時刻になるまでメッセージを一時的に蓄積するコンポーネントと上記 3 機能を併用することも可能である．

## 7. エージェントの移動

エージェントが移動する場合，そのエージェントに対応しているエージェントプロキシも移動する場合がある．交通シミュレーションの場合では，エージェントプロキシはシミュレーション実行環境に置かれ，道路を移動する．また，シミュレーション実行環境とエージェント実行環境は同じプロセスに置かれるため，エージェントプロキシの移動とともに，エージェントも移動する．しかしながら，こ

のようにエージェントとエージェントプロキシが常に一緒に移動するとは限らない．例えば，地図を管理するシミュレーション実行環境とエージェント実行環境が分離されるシステムで，シミュレーション実行環境を分離した場合では，エージェントプロキシは移動するが，エージェントが移動する必要はない．

アプリケーションに柔軟に対応するために，ZASE では，エージェントを移動させる API ではなく，エージェントを非活性化・活性化させる API とエージェントプロキシを非活性化・活性化させる API を提供する．これによりエージェントをバイト列に変換し，バイト列から復元することが可能となる．これらの API と実行環境間のメッセージング機能により，アプリケーションに適した移動機能を容易に実装できる．

## 8. 関連研究

MABS を行なうための基盤となるフレームワークやツールは，Swarm[4]， Ascape[5]， StarLogo[6]や NetLogo[7]などが知られている．しかしながら，これらの基盤は大規模化を目的としていない．現在，これらの研究を踏まえて，MABS をより現実の社会・経済の性質を反映した高度なものとするための一つの方向として，非常に多く(数千～数百万)のエージェントが現実的な速度で動作するシミュレーションフレームワークの開発がいくつか進められている．例えば，SugarScape モデルと同様のセルオートマトン型を基礎とした Repast[8]や MASON[9]がそれぞれ大規模化を目指している．MASON は多数のエージェントをシングルプロセスで管理するための機構を提供している．SOARS プロジェクト[10]では，CD ブートによるグリッド環境でのクラスタを容易に構築する機能を提供している．高橋ら[11]は超並列スーパーコンピュータ BlueGene で多数のエージェントを用いたシミュレーションを行うフレームワークを提案している．これは多数のノードが非常に高速なネットワークで結合されているシステムを想定している．我々が開発している ZASE もこのような流れの一つである．我々は，特に，MMABS のための基本機構の研究を対象としており，SugarScape モデルのような特定のモデルを前提としたものではなく，様々なモデルを実現するための基本機能を提供している．

## 9. まとめ

MMABS では，クラスタ機構が重要である．MMABS の実行環境は，シミュレーション実行環境とエージェント実行環境に分けることができ，それぞれも複数に分割される．この分割方法としては，一つのシミュレーション実行環境と複数のエージェント実行環境に分割する方式，シミュレーション実行環境とエージェント実行環境を同じプロセスに置き，そのようなプロセスを複数結合する方式がある．複数の実行環境を結合させる機能の鍵は，エージェントプロキシ，メッセージング，時間同期，エージェントの移動である．これらの機能により，様々なアプリケーションに応じた MMABS の実現が可能となる．

## 謝辞

本研究を行なうにあたり，京都大学社会情報学専攻石田亨教授より様々な助言をいただきました．ここに謝意を表します．また，本研究の一部は，総務省戦略的情報通信研究開発推進制度（SCOPE）「ユビキタスネットワーク社会におけるメガナビゲーション技術に関する研究」によるものです．ここに記して謝意を表します．

## 参考文献

1. G. Yamamoto and Y. Nakamura: Architecture and Performance Evaluation of a Massive Multi-Agent System, The Proceedings of Autonomous Agents 1999, ACM Press, 1999
2. G. Yamamoto, H. Mizuta, and H. Tai: A Platform for Massive Agent-based Simulation and its Evaluation, The First International Workshop on Coordination and Control in Massively Multi-Agent Systems, 2007
3. 北野宏明，松野文俊，田所諭，高橋友一，竹内郁雄，RoboCup-Rescue技術委員会: RoboCup – Rescue 情報科学の緊急災害対応問題への挑戦, IPSJ Magazine vol.41 No.4, 2000
4. N. Minar, R. Burkhart, C. Langton and M. Askenazi: “The Swarm simulation system: A toolkit for building multi-agent simulations”, 96-06-042 (1996). http://www.santafe.edu/projects/swarm/overview.ps.
5. “ Ascape”. http://www.brook.edu/es/dynamics/models/ascape/.
6. “StarLogo”. http://education.mit.edu/starlogo/.
7. “NetLogo”. http://ccl.northwestern.edu/netlogo/.
8. M. J. North, N. T. Collier and J. R. Vos: Experiences creating three implementations of the repast agent modeling toolkit”, ACM Trans. Model. Comput. Simul., 16, 1, pp. 1–25 (2006).
9. S. Luke, G. C. Balan, L. Panait, C. Cioffi-Revilla and S. Paus: “MASON: A Java multi-agent simulation library”, Proceedings of Agent 2003 Conference on Challenges in Social Simulation (2003). http://cs.gmu.edu/ eclab/projects/mason/.
10. H. Deguchi, H. Tanuma and T. Shimizu: “SOARS: Spot oriented agent role simulator - design and agent based dynamical system”, Proceedings of the Third International Workshop on Agent-based Approaches in Economic and Social Complex Systems, pp. 49–56 (2004).
11. T. Takahashi and H. Mizuta: “Efficient agent-based simulation framework for multi-node supercomputers”, Proceedings of the 2006 Winter Simulation Conference (2006).