マルチメディア通信と分散処理　102－6
（２００１．３．２１）

エージェントシステムにおけるスケジューリング機構の実装

伊藤　元晴　本多　弘樹
電気通信大学　情報システム学研究科

既存のエージェントシステムにおいては、個々のエージェントが自律的に動作を判断し、ネットワーク上を移動しながら処理を実行するのが一般的である。これに対し、本研究では、分散システム内の計算機環境を有効に利用し、個々のエージェントだけでなくシステム全体の処理効率を上げるためには、エージェントの動きをある程度全体的に制御をする機構が必要ではないかと考え、エージェントサーバにおいて、分散システム内における全ノードの計算機負荷情報の監視を行いながら、エージェントの動きを全体的に制御するための機構（エージェントスケジューリング機構）の実現を試みた。また、エージェントスケジューリング機構の有効性を確かめるために、例題エージェントを実装し、それを用いて実験と評価を行った。

Implementation of a Scheduling Mechanism for Agent Systems

Motoharu Itoh　Hiroki Honda
Graduate School of Information Systems,The University of Electro-Communications

In the agent systems, generally each agent autonomously makes decision to move on networks and perform some processings. In order to exploit the process capacity on a distributed system, we consider that a mechanism which could control entirely the motion of each agent is necessary. We propose an agent scheduler mechanism on a agent server that collects load-information from all nodes and controls all the motion of agents.We have implemented some sample programs of the agent systems, and used those samples to evaluate the effectiveness of our mechanism.

1. はじめに

次世代の分散処理技術として、エージェント技術が注目され、これを実現するソフトウェアが数多く、提案・公開されている[1][2][3][4][5][6][7][8][9][10][11]。

既存のエージェントシステムは、エージェントの作成を可能にする API 群とエージェントの起動・停止を管理するエージェントサーバから成り立っている。それらのシステムでは、エージェント自身がどこで処理をすれば良いのかを決定し、ネットワーク上を移動しながら処理を行うといった自律性をもつエージェントが実現されている。エージェントがより早く処理を終えてユーザの元に戻ってくるためには、エージェント自身が分散システム内の全てのノードの計算機負荷情報を取得し、それに基づいて移動経路の順番を考えて初めて実現可能となる。従来、エージェントサーバが、分散システム内にある各ノードの計算機負荷情報を監視し、エージェントの動きを全体的に制御するようなエージェントシステムはなかった。

本研究では、エージェントサーバにおいて、サーバのあるノードだけでなく、分散システム内における他ノードの計算機負荷情報の監視を行いな

がら、エージェントの動きを全体的に制御する機構（エージェントスケジューリング機構）を考案し実装を行った。また、例題エージェントにより、その有効性を評価した。

2. 従来のエージェントシステムの問題点

従来のエージェントシステムでは、エージェント自身が自律的に、分散システム内のノードを選択し、そこに移動して処理を行う。しかし、ユーザにエージェントの移動先が不透明になっているため、例えば、移動先のノードがダウンし、エージェントが消滅したことに気付かないでいるという事態が発生するという恐れがある。

また、従来のエージェントシステムでは、エージェントの処理を全体的に効率よく制御するための機構を備えていないため、分散システム内の計算機環境の有効利用という面では問題点もある。

分散システム内の計算機環境を有効に使い、かつ、エージェントが分散システム内の計算機環境において、より早く処理を終えて、安全にユーザの元に戻ることができるようにするには、エージェントサーバ内に、エージェントの動きを制御する機構が必要である。

3. エージェントシステムにおけるスケジューリング機構とその実装

3.1 スケジューリング機構の概要

本方式では、エージェントシステムのエージェントサーバにおいて、分散システム内における全ノードの計算機負荷情報の監視を行いながらエージェントの動きを全体的に制御する。

具体的には、それぞれのエージェントがなるべく早く処理を終えることができるようノードの計算機負荷に基づいてエージェントの移動先のノードを決定する。移送先の決定に際しては、分散システム内の全ノードの計算機負荷値だけでなく、そのノードの計算機性能やネットワーク性能も考慮に入れる。また、エージェントの実行順序の決定に際しては、エージェントの優先度を考慮する。

3.2 スケジューリングの構成と実装

本研究では、エージェントスケジューリング機構を、エージェントシステムは、Mercury[1]に実装した。

実装したエージェントスケジューリング機構は、「負荷情報収集エージェント」、「グローバルエージェントスケジューラ」、「ローカルエージェントスケジューラ」から構成される（図1）。

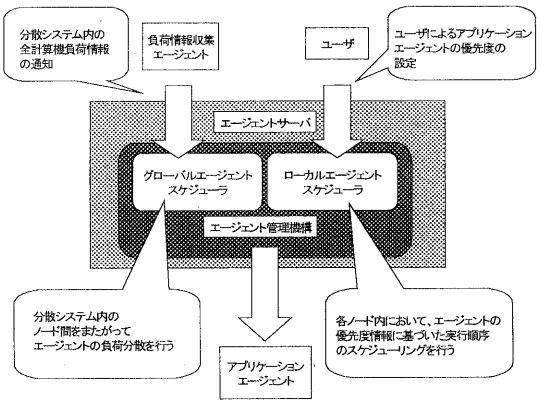

図1　スケジューリング機構の構成

3.2.1 負荷情報収集エージェント

計算機負荷情報を一定時間おきに収集する機能を負荷情報収集エージェントとして実装した。

負荷情報収集エージェントは、各ノードに計算機負荷情報の収集に行くワーカエージェントと、サーバのあるノードに滞在しワーカエージェントから分散システム内の各ノードの計算機負荷情報を受け取るマスタエージェントから構成される。

負荷情報収集エージェントの処理手順は、一定時間ごとに以下のステップを繰り返している。

(ステップ1：複製によりワーカエージェントを作成)

マスタエージェントは複製技術を用いることにより、各ノードの計算機負荷情報を収集するワーカエージェントを作成する。

(ステップ2：全ノードのワーカエージェントの送信)

マスタエージェントが、サーバのあるノード(マスタノード)から分散システム内の全ノードにワーカエージェントを送信する。

(ステップ3：ワーカエージェントによる計算機負荷情報の取得)

各ノードに、移動してきたワーカエージェントは、そのノードの計算機負荷情報(計算機性能、計算機負荷値、処理待ちのエージェントの個数)を取得する。計算機負荷情報の取得は、例えば、LinuxOS の場合では、proc/cpuinfo ファイルから、計算機性能を取得したり、proc/loadavg ファイルから、そのノードの計算機負荷値を取得する。処理待ちのエージェントの個数は、エージェントサーバから通知を受ける。

(ステップ4：取得した計算機負荷情報のマスターエージェントへの通知)

各ワーカエージェントは、計算機負荷値を取得した後、元のノード(マスタノード)に戻り、マスタエージェントに自分のラウンドトリップ時間と共に、各ノードの計算機負荷情報を通知する。

(ステップ5：グローバルエージェントスケジューラへの計算機負荷情報通知)

マスタエージェントは、グローバルエージェントスケジューラに全ノードの計算機負荷情報を通知する。

3.2.2 グローバルエージェントスケジューラ

ノード間をまたがってエージェントの移送先を決定するスケジューリング機構をグローバルエージェントスケジューラとして実装した。

グローバルエージェントスケジューラは、常に分散システム内の各ノード(現在いるノードの計算機負荷情報も含む)における最新の計算機負荷情報を負荷情報収集エージェントにより取得し、その得られた情報をもとにスケジューリングを行う。スケジューリングした結果情報を、アプリケーションエージェントに通知する。

グローバルエージェントスケジューラの処理手順は、以下のステップが繰り返される。なお、ステップ2では、計算機環境が均質か非均質かによって処理を場合分けしている。

(ステップ1：計算機負荷情報の受け取り)

グローバルエージェントスケジューラは、負荷情報収集エージェントにより、収集されてきた全ノードの計算機負荷情報(計算機性能、計算機負荷値、処理待ちのエージェント個数、ラウンドトリップ時間)を受け取る。

(ステップ2：計算機負荷情報をもとにスケジューリング)

各ノードにおける計算機負荷情報を元に後述のスケジューリングを行う。このとき、負荷情報収集エージェントにより取得してきた計算機負荷情報の中の、計算機性能を比較する。分散システム内の計算機環境によって2通りに場合分けしてスケジューリングを行う。

(ステップ2.1：計算機環境が均質の場合のスケジューリング)

各ノードの計算機負荷値を元に、最も計算機負荷値の小さいノードを決定する。

(ステップ2.2：計算機環境が非均質の場合のスケジューリング)

各ノードの処理待ちのエージェント個数と負荷情報収集エージェントのラウンドトリップ時間を元に、最も計算機負荷値の小さいノードを決定する。

（ステップ３：ノードデータ情報の通知）
グローバルエージェントスケジューラが、計算機負荷値の低い順に順位付けされたノードデータ情報をアプリケーションエージェントに通知し、そのノードにアプリケーションエージェントを移動する。

3.2.3 ローカルエージェントスケジューラ

各ノードのエージェントサーバで、エージェントを実行する順番を決定するスケジューリング機構をローカルエージェントスケジューラとして実装した。これにより、サーバのあるノード内に常駐するエージェントだけでなく、他のノードから送られてきたエージェントも優先度に応じて、より早く処理することができるようにした。計算機負荷情報の最新情報を各ノードのエージェントサーバが取得できるようにするため、負荷情報収集エージェントの優先度は、他のアプリケーションエージェントよりも先に処理を行うことができるようにするために最も高く設定している。

ローカルエージェントスケジューラは、処理すべきエージェントを要求パケットとして受け取る（図２）。

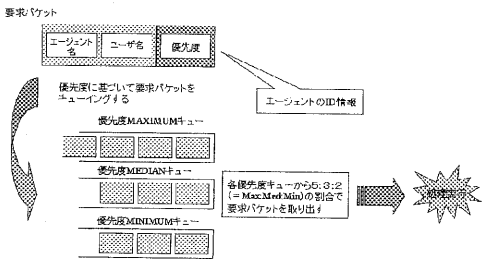

図2　優先度キューからの要求パケット取り出し処理

要求パケットには、エージェントを利用するユーザが設定したエージェント ID 情報が添付されているが、本研究では、優先度情報をエージェント ID 情報に追加した。この優先度情報をもとにエージェントの実行順序をスケジューリングする。

ローカルエージェントスケジューラの処理手順は、処理手順は、以下のステップが繰り返される。

（ステップ１：優先度情報の取得　）
優先度情報をユーザ ID から取得する
（ステップ２：要求パケットのキューイング）
優先度情報に基づいて優先度別にキューイングを行う。
（ステップ３：エージェントの処理の実行）
優先度別に要求パケットを取り出し、エージェントの処理を行う。

4. 評価

エージェントシステムにおけるエージェントスケジューリング機構の有効性を確かめるために、例題プログラムを実装し、実験を行った。実行環境を以下に示す（表１）。

表１　実行環境

ノード	CPU[MHz]	Memory[MB]	OS
ノード 0	(IntelCelron) 686	64	Linux
ノード 1	(IntelPentiumII) 400	263	Linux
ノード 2	(IntelPentiumIII) 801	662	Linux
ノード 3	(IntelPentiumIII) 501	263	Linux

例題プログラムとしては、以下のエージェントを実装した。

（1） エージェント間通信を用いず、協調動作を行なわないエージェント（文字列検索エージェント）

（2） エージェント間通信を用いて、協調動作を行うエージェント（地図作成移動エージェント）

実装したエージェントスケジューリング機構を使う場合と使わない場合でそれぞれ実験および評価を行った。

[文字列検索エージェントによる実験と評価]

あらかじめ、分散システム内の各ノードにそれぞれ同一のファイルを用意しておく。文字列検索エージェントはそのファイルの文字列検索を行う。

実験では1個の文字列検索エージェントをいずれかのノードに送って処理を終えて元のノードに戻ってくるまでの時間と、１０個の文字列検索エージェントを送りすべてのエージェントが処理を終えて元のノードに戻ってくるまでの時間をエージェントスケジューリング機構を使う場合と使わない場合で測定した（図3）。

エージェントスケジューリング機構を使わない従来方式の場合は、エージェントを決められた順序で各計算機に送る。

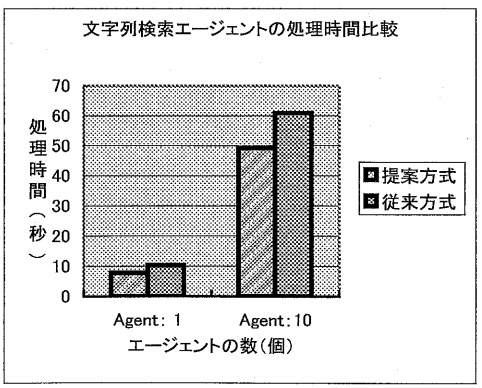

図3 文字列検索エージェントを用いた実験

実験は、エージェントが1個の場合と、１０個の場合で、それぞれ１０回ずつ行った。図3の結果は、それぞれの場合における１０回分の実験結果の平均値である。実験結果から、どちらの場合でも、提案方式の方が、従来方式よりもエージェント全体の処理時間が短く、エージェントスケジューリング機構が有効であることが確認できた。

[地図獲得ロボットエージェントによる
実験と評価]

地図獲得ロボットエージェントは、マスタ・ワーカモデルを用いて実装している。ワーカエージェントに与えられた処理は、一定時間内にできるだけ多くの地図情報を取得し、地図を作成することである。各ノードには、同じ地図環境が用意されている。ワーカエージェント（３０個）は、マスタエージェントに、エージェント間通信によって地図情報を通知する。

実験では、ワーカエージェントにより地図領域の情報量（地図情報５００個）を取得するのにかかった時間を測定した（図4）。

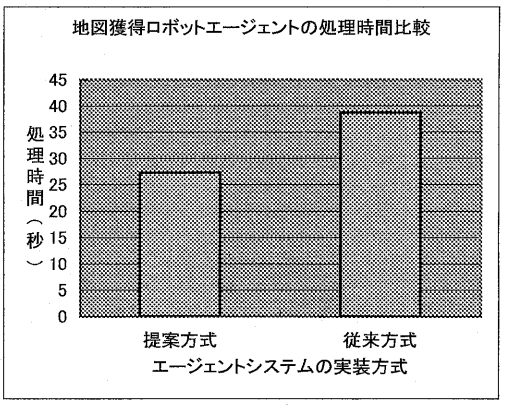

図4 地図獲得移動ロボットエージェントを用いた実験

図４の結果は、１０回分の実験結果の平均値である。実験結果から、提案方式の方が、従来方式よりも処理時間が短く、複数のエージェントがエージェント間通信を使って処理を行っている場合でも、エージェントスケジューリング機構が有効であることが確認できた。

5. まとめ

本研究では、エージェントサーバが、分散システム内における全ノードの計算機資源の監視を行いながらエージェントの動きを全体的に制御する機構を試みた。実装したエージェントスケジューリング機構は、グローバルエージェントスケジューラ、負荷情報収集エージェント、ローカルエージェントスケジューラの３つで構成されている。エージェントスケジューリング機構の評価実験から、スケジューリング機構の有効性を確かめることができた。

今後の課題としては、ローカルエージェントスケジューラにおいて必要な情報となるエージェントの優先度付けをどう行うかがあげられる。現在は、ユーザが自由に設定できるようになっているが、エージェントサーバがエージェントの使用目的に応じて判断して、優先度付けをする必要がある。また、グローバルエージェントスケジューラにおいては、均質・非均質環境においてより良い効果を得るために、計算機性能と計算機負荷値、そしてネットワークトラフィックなど各ノードの計算機負荷情報として必要な要素を、全て組み合わせてスケジューリングを行い、最適解を求めるスケジューリングアルゴリズムを考える必要がある。

参考文献

[1] 岩井俊弥, Java モバイル・エージェント, ソフト・リサーチ・センター.

[2] 佐藤一郎 ,AgentSpace Web Page (http://islab.is.ocha.ac.jp/agent) , 1997.

[3] 山本　学, Web アプリケーションのためのエージェントサーバ技術,日本情報処理開発協会,ネットワークエージェントに関するワークショップ資料, pp.111-114, 2001.

[4] HendroSubagyo, 高汐一紀, 動的な環境に適応する移動エージェントの実装と評価, 情報処理学会,コンピュータシンポジウム論文集, pp.139-146,2000.

[5] 大須賀昭彦, プランニングモーバイルエージェント, bit, Vol.31,No3.3, pp.88-95.

[6] 佐藤一郎, 高橋美奈子, モーバイルエージェントの階層的な構成と移動, 日本ソフトウェア科学会第１５回大会論文集, pp249-252.

[7] 高汐一紀, 他, 時間に依存した移動エージェントのための動的資源予約機構, 情処 マルチメディア, 分散, 協調とモーバイル(DoCoMo'99) シンポジウム, 1998.

[8] 熱田真弓, 他, 移動エージェントのためのチェックポイント管理機構, 日本ソフトウェア科学会第１６回大会論文集, 1999.

[9] 梅澤　猛, 他, センサネットワーク利用のためのモバイルエージェントプラットフォームの設計と実装, 日本ソフトウェア科学会第１６回大会論文集, 1998.

[10] 高汐一紀, 他, ユーザモビリティの実現を目的とした移動エージェントフレームワーク, 日本ソフトウェア科学会第１６回大会論文集, 1998.

[11] 鵜林尚靖, 玉井哲雄, ロール概念に基づく発展型移動エージェント, 日本ソフトウェア科学会第１６回大会論文集, 1998.