成蹊大学理工学研究報告
J. Fac. Sci. Tech., Seikei Univ.
Vol.42 No.1 (2005) pp.27-32

自律分散処理モニタと分散処理エディタの開発

小川　大介*[1]，甲斐　宗徳*[2]

Implementation of Monitor and Editor for Autonomic Distributed Processing System

Daisuke OGAWA*[1], Munenori KAI*[2]

ABSTRACT：In autonomic distributed processing, it is difficult because of the autonomy for an administrator or a user to grasp the behavior and state inside the system correctly, and even if a certain obstacle is encountered in a system, intervention of an administrator is difficult. So, in this research, a monitor of autonomic distributed processing and an editor for distributed processing are designed and implemented. An autonomic distributed processing monitor offers the interface which acquires the information inside distributed processing system, and displays them, or operates a mobile agent. A user can grasp and control the behavior and state inside a system through this monitor. The editor for distributed processing can support description of the application program for distributed processing using communication. At the time of communication command creation, a sending command and a receiving command, corresponding to each other, can be described simultaneously and a communication error by user's easy mistake can be reduced.

Keywords：Autonomic Distributed Processing, System Monitoring, Editor, Communication, Programming

(Received March 25, 2005)

１．はじめに

近年，IT 化が進んでおり，それと共に情報も複雑・多様化されてきている。また，パソコン自体が，一般家庭にも広く普及し，個人でも複数のマシンを LAN などで接続して同時に利用する人が増加している。この様なことから，マシンに解かせる問題も，さまざまな形となって現れてきた。高性能のマシンであれば，様々な問題に対応することも可能であろうが，そのようなマシンは個人に限らず，企業等でもコストがかかりすぎてしまう。そこで,複数のマシンを使うことによって,低コストで,誰にでも簡単に使うことができる分散処理システムの実現が望まれる。当研究室では特に，システム全体を管理しながら，成長を続けていく，自律分散処理システムの構築を目指して開発してきた。[1]

自律分散処理システムとは，自律的にシステムの状況の変化に対応しながら分散処理を行うシステムである。一般に分散システム上で，タスクの負荷分散や停止したタスクの再生処理等の作業にはシステム全体の状況把握や管理が必要であり，それらのコントロールをすべて含めてユーザがアプリケーションプログラムを記述するのは非常に困難であると考えられる。システムの自律性を満たすためには，システム内で動く各ソフトウェアが，他からのメッセージに応じた行為と自ら取得した外部の情報に応じて判断を行い，その結果によって行動を計画し，その計画に基づき行動できる必要がある。そこで，自律分散処理システムを実現するにあたって，モバイルエージェント[2]を利用することにした。システム上には,タスクエージェントをはじめ様々な機能を持つ独立したモバイルエージェントが存在し，それら全体でひとつのシステムを形成する。不足しているとわかった機能は,その機能を持ったモバイルエージェントをシステムに投入することで実現することが出来る。モバイルエージェントは，ネットワークを通じてマシン間を移動できるソフトウェアである。またエージェントとは，人間の代理

*[1] 情報処理専攻大学院生
*[2] 情報処理専攻助教授(kai@st.seikei.ac.jp)
Associate Professor, Dept. of Information Sciences

人となってシステム環境を認識し与えられた目標を達成するため自律的に行動するソフトウェアのことである。

分散処理内部の動きは不透明で，システム内部でトラブルがあった場合でもユーザの介入が困難である。そこで，各マシンの情報を視覚的にとらえ，エージェントを多少なりとも操作できるインタフェースをもったものを作る。それを分散処理モニタと呼ぶ。ここでのモニタは，分散処理の手助けまたは監視に使いたいものなので，分散処理に大きな影響を及ぼし，分散処理システムへの妨害はもちろん，それ自身が大きなオーバーヘッドになることがあってはならない。

分散処理において，障害が最も起こりやすいのは，データ転送時における通信である。通信を分散処理システムの中で高速性や効率，自由度を考えて実装するのは最も困難な作業の１つである。さらに通信の記述は送信側と受信側の対応を考えて，引き数の種類や数，どの相手のどのメソッドと通信するかといった点に注意せねばならず，容易ではない。記述後も，エラーが起きたときなど，通信命令がどのソースの，どの通信命令と通信を行っているのが分かりづらい。そこで，通信の記述をサポートし，コーディング時，およびデバック時にも通信相手の対応する通信コード部分がどこに入ればよいか，簡単に分かるような機能を持った，分散処理用エディタの開発を行った。

２．AgentSpace

本システムでは国立情報学研究所の佐藤一郎氏が開発した AgentSpace[2] というモバイルエージェントシステムを利用している。幾つかある既存のモバイルエージェントシステムの中から AgentSpace を選んだ理由は，AgentSpace はエージェント移動時にエージェントの情報を圧縮して送るため，エージェントの移動が速いという点と，オープンソースであるためにシステムを拡張しやすかったらからである。

本システムを構築する上では高性能なエージェントの機能はそれほど必要ではなく，我々が目的とする分散処理に特化し易いモバイルエージェントであることが望ましい。従ってエージェントの生成，高速な移動といった，基本機能を満たしているモバイルエージェントシステムであればよい。分散処理を行う上では通信部分，つまりエージェントの移動部分がボトルネックになる可能性があり，そのためエージェントの移動が高速であるということは重要と考えられる。

３．システムの特色

まず，開発したシステムの目的は「分散処理の手法について知らない人でも簡単に分散処理の恩恵を受けられるシステムを構築すること」である。そこで，本システムは GUI を用いて操作を簡略化することにより初めて使う人でも容易に使用できるようにする，システムの稼動状態を分かりやすく表示するなど，初心者にも使いやすいように工夫することにした。

システムのもうひとつの特徴として，「様々な種類のプログラムを実行可能な汎用性の高いシステム」を目指すことにした。当初このシステムは「ユーザにとって書き易く分散処理する事により処理時間の短縮が可能な SIMD 形式の Java プログラムを分散処理できるシステム」を目指していた。しかし「SIMD 形式 Java プログラム」のみを分散処理の対象としたのでは汎用性が高いとは言えない。せっかくシステムをモバイルエージェントで実装しているのだから「モバイルエージェントとしての機能をユーザが自由に利用できる高度なタスクプログラム」も記述して分散処理できるようにすべきである。そこで，SIMD 形式の Java プログラムだけでなく，エージェント同士が自由に通信を行い，データをやり取りできるような MIMD 形式のプログラムにも対応できるようにする。SIMD 形式のプログラムは Java の class ファイルで作成し，MIMD 形式のプログラムは jar ファイルにまとめてタスクファイルとして使用できるようにする。図１は本自律分散処理システムの構成例を示している。

システムは LAN 環境で接続された複数のマシンからなる。

エージェントは AgentSpace 上でしか存在できないので，利用したいマシンで AgentSpace を起動させる必要がある。

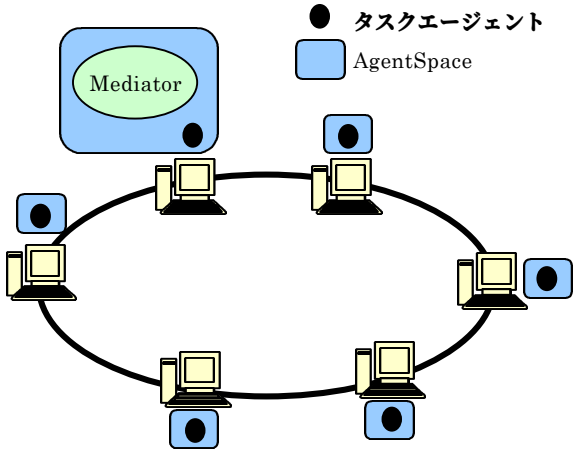

図１　自律分散処理システム構成例

ユーザは本システムを用いて分散処理を行いたいと思ったとき任意のマシンのAgentSpace上でMediatorというプログラムを起動する。Mediatorはユーザとのインタフェースとなるエージェントで，ユーザからタスクファイルなど必要な情報を入力されるとその情報に従いタスクプログラムを分散処理するタスクエージェントを生成する。

タスクエージェントはMediatorによってうまく負荷分散されるようにネットワーク上のマシンに振り分けられて移動する。それぞれのマシンに移動後，タスクエージェントはタスクプログラムを実行して解を求める。解を求めたタスクエージェントはMediatorのあるマシンに返り，Mediatorに解を教えて消滅する。最後にMediatorが全ての部分解を統合して全体解を出力した後，一連の処理は終了する。

また，佐藤氏が開発したオリジナルのAgentSpaceは，他のマシン上に起動しているAgentSpaceの情報を知ることはできない。そこで，システム全体に関する動的な情報の取得・更新を行うために，まずAgentSpaceが定期的に性能値を更新して他のAgentSpaceにブロードキャストする方法が挙げられる。しかし，この更新間隔を調節することが困難であり，さらにAgentSpaceの数が増加するにつれて，ブロードキャストによる通信回数が増加してしまう問題点がある。

そこで，すべてのAgentSpaceを巡回する管理エージェントを設けて，その管理エージェントがAgentSpaceから情報を受け取り，他のAgentSpaceに伝えて回ることで，AgentSpace同士でのブロードキャストが必要なくなり，通信回数を削減することができる。

その結果，負荷バランスを考慮したタスク分散，使用可能なマシンの識別や，ダウンしたタスクの再送信，再配置など，本来ユーザが管理しなければならない事象を自律的に管理することが可能となり，システムの効率性・安全性が向上すると考えられる。

４．分散処理モニタ

4．1　分散処理モニタの目的

分散処理モニタとは，現在システムがどのように動いているのかシステムの振る舞いをわかりやすく表示するためのものである。

管理エージェントの役割は，分散処理が効率よく行われるようにシステムを管理したり，また，分散処理が進行しなくなったときに自動的に処理を復旧することである。しかしながら，システムに何らかのトラブルがあったときに，管理者が介入して対処したいときがある。そこで，そのための情報を管理者に伝えることが，分散処理モニタの役割である。

分散処理モニタの画面を図２に示す。

図２に示すとおり，ボタンが8個用意されている。その内訳は，

- 情報表示に関して1個(更新)
- 他のマシン情報に関して2個(派遣情報，滞在情報)

図２　分散処理モニタの画面

- エージェントに関して3個(生成，派遣，消去)
- 管理エージェントに関して2個(存在指定，全体停止)となっている。

4．2　情報の表示について

分散処理モニタは以下の情報を表示するとともに，ユーザが操作をする時は，常にその操作を行った時刻の表示をするようにしてある。

① 自分のマシン（IPアドレス，性能値）
② 自分のマシンに存在しているエージェントの情報（名前，エージェントID）
③ 他のマシンリスト（IPアドレス，性能値）
④ 他のマシンの静的情報（IPアドレス，性能値，名前，エージェントID ）
⑤ 他のマシンの動的情報（エージェント出発時間，到着時間など）や新しくリストに加わったマシン，リストから外れたマシンの情報

①～③は，「更新」ボタンで，④は，「派遣表示」ボタンで，⑤は，「滞在表示」ボタンで表示できる。

管理者が分散処理中に，異常の疑いがあるマシンに気がついた時，そのマシンにエージェントを直接送りこんで，情報を取得できるように以下のものを実装した。

- 派遣情報表示：目的のマシンにエージェントを派遣し，情報を取得後すぐに自分のマシンに帰ってきて，そのマシンの情報を表示する。
- 滞在情報表示：目的のマシンに指定時間滞在し，

時間内にそのマシンで全てのエージェントがどのような動きをしているのか，動的な情報を取得し，その情報を表示する。

5．分散処理用エディタ

5．1　分散処理用エディタの目的

分散処理用エディタは，分散処理用アプリケーションを記述するときに，通信命令を記述しやすいようにサポートし，作成後も，通信命令がどのソースの，どのメソッドと通信しているかが簡単に見てとれる機能を提供する。

5．2　AgentSpace での通信

AgentSpace でサポートされているエージェント間通信は同一 AgentSpace 内のエージェント同士でしか通信が行えない。しかし，あるエージェントがタスク実行中に，他の AgentSpace にある別エージェントへデータを送ったり，あるいは相手エージェントの動作を要求する必要が起こったとする。エージェントが移動したのではタスクの実行が中断されてしまう。よって，エージェントがどこにいようとお互いに通信ができる機構が必要となる。

そこで通信要求の生じたエージェントはエージェント間通信専用のエージェントを生成し，送りたいメッセージと，送りたい相手先のエージェント ID やエージェント名を渡す。送信元エージェントが自分で移動して相手先を探すのではなく，通信専用のエージェントに仲介してもらうことで，タスク実行の中断を防ぐことができる。通信専用のエージェントは他のエージェントからの呼び出しによって生成されて送信データを受け取った後，別マシンへ飛び通信相手先エージェントを探索する。通信相手を探し当てた通信専用エージェントは，データを送信し，その場で消滅する。

5．3　通信方法

通信方法に関しては，今年度，当研究室の田久保が実装を行った。実装した通信方法は以下の通りである。

① １対１エージェント間通信
　グループ内で各エージェントを特定して通信する方法
② エージェント ID 指定通信
　エージェントの ID を指定して通信する方法
③ グループ名通信
　グループ内のエージェント全てに送信する通信方法
④ エージェント名指定通信
　エージェント名を指定して通信する方法

5．4　分散処理用エディタ画面

Java の Swing を利用したもので，GUI となっている。もちろんエディタとしての基本機能は全て備えていて，保存などのファイル操作や，切り取りや，貼り付けといった編集も使用可能である。

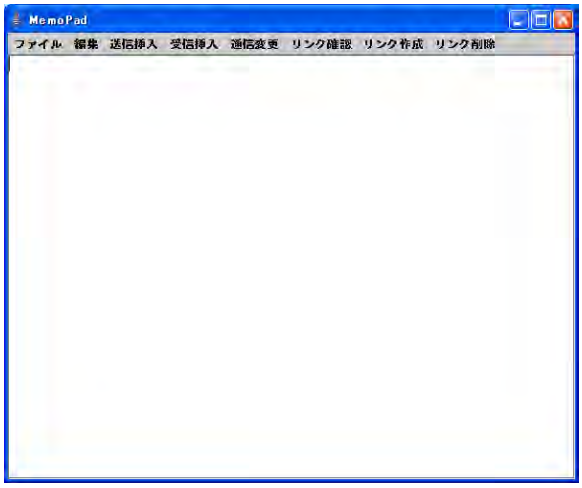

図３　分散処理用エディタ画面

5．5　通信命令挿入

通信命令挿入というのは，通信命令を記述するものである。ここでは，開発者のコーディング時のミスをできるだけ予防できるように，ダイアログにしたがって記述していくだけで命令が完成するようになっている。例えば「エージェント名による送信命令」を記述する場合に

図４　通信方法選択画

図５　AgentContext 入力画

図６　型決定画面

は，まず図４のようなダイアログで通信方法を尋ねられる。次にインターフェースとなり，通信の手助けとなるAgentContext 変数を入力（図５）する。そして，図５と同様のダイアログ画面が，表示され，通信相手のエージェント名，通信相手の受信メソッド名を入力する。ここまでが通信に必要な条件の設定である。

次に，通信を行うデータの入力になる。
図６の画面で扱うデータの型を決定し，その後，図５のダイアログで変数，もしくは扱うデータそのものを入力する。これが基本的な通信命令記述方法である。

挿入方法は，用途に合わせて送信・受信ともに３種類の挿入方法を用意した。

(1) 通信同時挿入

送信（受信）命令を記述するときに，同時に受信（送信）命令を作成し，通信に使う必要な情報，すなわち，受け取りメソッド名，データの変数や数など通信相手同士を，対応して記述できるようになっている。対応して記述できるので，アプリケーション実行時の通信の保証も成される。出来上がった送信(受信)命令は，ソースのカーソルの位置に挿入でき，相手の受信（送信）命令は，挿入するソースを選択した後，貼り付けボタンを使って挿入することができる。同時挿入では，通信するデータの数は，取消しボタンが押されるまで，通信したいデータの個数分入力できる。また，この場合にのみ，受信命令の変数も設定するように図５のダイアログが表示される。

(2) 既存通信命令対応挿入

既に通信情報に登録されている受信（送信）命令に合わせて，送信（受信）命令を挿入する方法である。受信命令が，Receive という名で，データ数が３個であれば，送信命令はそれに対応するように受け取りメソッド名を Receive で，データ数を３個と記述ができる。この挿入方法では，通信相手のデータがもし，String であれば，String で入力するようにダイアログが出る（図７）。

図７　対応するデータが String の場合

(3) 単独挿入

通信相手を考えずに送信（受信）命令だけをそのまま記述するものである。送信命令作成時には図６の画面は表示されず，変数，もしくはデータを入力する画面だけが出てくる。

5．6　リンク作成

単独作成で通信命令を記述した場合や，後に紹介するリンクを削除する操作が行われた後などには，通信相手が設定されていない。しかし，通信命令自体は通信情報として登録されている。リンク作成はこの通信情報に登録されている通信命令同士を，リンクさせるものである。これはアプリケーションには実際に何の影響もないが，リンクを作成することで，いつでも，通信相手が分かるようになる。通信したい命令を選択すると，ソースファイル選択画面が開かれる，リンクをさせたい命令があるソースを選ぶと，先ほど選択した通信命令と，条件が合うものが一覧表示される。

条件とは，送信命令に記述されている通信相手のメソッド名が，受信命令のメソッド名と合致するかどうか，送りたいデータの数が一緒かどうかを検索している。

5．7　リンク確認

通信挿入時やリンク作成時に設定した，通信相手を確認する機能である。リンクを確認したい通信命令を選択すると，相手ソースファイルが開かれ，通信相手命令のところまでカーソルが移動する。通信命令は１対１とは限らず，複数通信対象がある可能性がある，その場合には，リンクしている全ての通信ファイルが開かれる。リンクが設定されていない場合には,「リンク設定されていません」と表示する。

5．8　通信情報登録

通信命令挿入で記述すると，通信情報が，記述しているソースがあるディレクトリに専用のディレクトリが生成され，その下に保存される。この情報を使用すれば，リンクが確認でき，更新することでリンクの作成や，リンクの削除を行っている。図８はその送信命令側の通信情報の例である。

```
ac.Send_ToAID(AID,"rm",vec,2)↓
NL↓
NC↓
ac.Send_ToName("ran","uketori","word",1)↓
sample2.java↓
uketori(String str,Integer num)↓
```

図８　送信命令情報

３行で１セットになっていて，１番目が送信命令，２番目が通信する相手ファイル名，そのファイルの通信命令が３番目にある。ＮＬとＮＣは１セットで，通信相手命令がないことを示している。

ファイル名が sample.java である場合，送信命令情報は sampleS.txt に，受信命令は sampleR.txt に保存されている。

この情報は通信命令の作成や，リンク作成などの機能を使うごとに更新される。

5．9　その他の機能

上記の他にも以下の機能を実装した。

- リンク削除
  リンク作成とは逆に，リンクを切る機能
- 通信命令変更
  登録した通信命令を，変更する場合に使う。リンクは全て切れるが，登録は残る。
- 保存時の命令確認
  キーボード操作で通信命令を削除したときに，通信命令がないことを確認し，自動的に登録情報から削除する機能
- 全リンク削除
  通信命令を変更したときや，命令がなくなったときにリンクを全て消去する機能

６．評　価

分散処理モニタのメリットは以下の４点である。

① 分散処理モニタを通じて行った操作は，全ての履歴が時刻とともに残り，過去の操作も確認することができるようになっている。
② 情報の表示，エージェントの操作ができ，ユーザにとって使いやすいものとなっている。
③ マウスですべて操作できるので，ユーザにとって，とても使いやすい設計になっている。
④ ある期間，マシン上の全てのタスクエージェントが異常な動作をしていないかを見ることが可能となっている。

以上のことから，システム使用者が，モニタを使用することで，情報得て，エージェントの操作を行えるといったように，分散処理に介入しやすくなったといえる。

分散処理用エディタの利点は以下の３点である。

① 通信の記述が，ダイアログにしたがっていくだけで簡単に作成できる。
② 通信相手の命令を同時に記述したり，通信相手に対応するように通信命令を記述数することで，アプリケーション実行時の通信の保証ができる。
③ 通信情報をいつでも確認できる。

ユーザは関数の仕様を知らなくても，通信命令の記述が行え，通信時における実行エラーを未然に防げる可能性は大きくなった。

７．おわりに

分散処理モニタでは，分散処理内部でどんな情報が保たれているかがわかり，また，マシンにどんなエージェントが存在するのか，いつ生成されて，いつ削除されたのかというように，ある程度エージェントの動きを把握できるようにもなった。それを見てエージェントの操作が行え，ユーザの介入がしやすくなった。

通信部分では通信部分の記述をサポートすることで，実行エラーを未然に防ぐことができようになり，また，実行エラーが起きたとしても，通信命令がどの通信命令と対応しているかが分かり，その後の対処がしやすくなった。

今後は分散処理モニタに実行エラーを，分散処理モニタに表示できるように改良していくべきである。その際にはエディタでデバッグをサポートできるような形を取れることが望ましい。

また，分散処理用エディタでは，このエディタを使って書いたものではないファイルの通信を自動検出し，登録できるようにすれば，より便利なものになるだろう。

参考文献

[1] 佐々木竜介，小川大介，坂井功，甲斐宗徳「モバイルエージェントを用いた自律分散処理システムの構築　－自律分散処理システムにおける性能とユーザビリティの向上－」2004 年度電気学会電子・情報・システム部門大会論文集，pp.1036-1040，(2004-9)
[2] 佐藤一郎　モバイルエージェントシステム
http://islab.is.ocha.ac.jp/agent/index.html