Javaを用いた分散型メールエージェントの設計*

2N－5

溝口文雄*　大和田勇人*　矢口敬正†

東京理科大学 理工学部‡

1　はじめに

最近のインターネットの普及によってネットワークに接続されるユーザの数は急速に増加している．また，近年のような分散したネットワークのもとでは，ユーザーは，好きな時に好きな場所から接続できる必要があり，ユーザーの利用頻度の高い電子メールにしてもそうした分散環境に対応したシステムが必要となる．

例えば，出張で社外からネットワークにアクセスした時に，社内にいるのと同じ環境でメールを管理できたら大変便利である．そのためには，メールを見るクライアントとメールの送受信を管理するメールサーバーが，それぞれ，独立した分散環境にある必要がある．

また，次世代のメールシステムとしては，データベースによる個人メールの管理や，ユーザーの振舞いを学習して，自動的に分類してくれる機能 [1] が望まれる．

このような多機能を備えたシステムにおいては，全ての機能を一度に呼び出すのではなく，必要な機能を必要に応じてダウンロードした方が，クライアント側の負荷は軽くなる．しかし，システムの処理速度を考えると，個々のサーバー内での処理の分散も重要となる．即ち，一つの処理を分散して同時に行なうことにより，処理能力の効率化を目指す．

本稿では，電子メールを対象として，このような分散型の電子メールエージェントの設計を行なう．

2　設計方針

本稿で提案する分散型は，以下の2点である．

- 処理の分散
  処理能力の向上を目指して少しでも早い作業を行なうために，これまで，特に時間がかかっていた学習の分野に着目し，学習処理内部の分散環境を構築．図1の点線矢印において，分散オブジェクトモデルで実装する．分散オブジェクトを使うことで，クライアントとサーバーの関係を意識せずにプログラムできる．また，同じようなデータを各マシンに渡す場合，各マシンに文字列を渡して，渡した先でそれを加工するよりも，分散オブジェクトモデルを使って，すでに加工された文字列（オブジェクト）を渡した方がより高速な処理が望める．

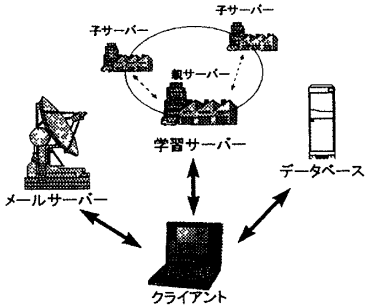

図 1: 分散のイメージ

- サービス（サーバー）の分散
  学習やデータベースなどのそれぞれのサービスが独立したホストにあっても，接続できるシステムの構築．メールサーバーやデータベースのような既存のサービスは，サービスそのものをオブジェクトにして渡すのは不可能である．そこで，図1の実線矢印においてクライアント・サーバーモデルで実装する．

3　システムの設計

本システムは，基本的に全てJava言語で実装する．Java言語を使うことの利点は アーキテクチャ中立であり，これにより，Javaの動く環境であれば，再コンパイルせずに，プログラムを実行することが出来る．

そこで，本システムで実装する分散環境は，Sunマイクロシステムズによって開発されたRMIを用いる．これは，分散環境に対応したネットワーク言語で，Javaで書かれているため，Javaのプログラムを簡単に分散させることが出来る．

3.1　サーバーの設計

以下に挙げるのがRMIで書くサーバーのプログラムであり，クライアントがサーバーとの接続の際に参

*Design of Distributed Mail Agent using Java
†Fumio MIZOGUCHI,Hayato OHWADA,Takamasa YAGUCHI
‡Faculty of Sci. and Tech. Science University of Tokyo

照する部分である.

```
try{
    System.setSecurityManager
       (new RMISecurityManager());
    // レジストリー立ち上げ
    LocateRegistry.createRegistry(8888);
    // サーバーの立ち上げ
    ServerImpl s = new ServerImpl();
    Naming.rebind("//:8888/ilpserver",s);
    System.out.println("ILP SERVER ready.");
} catch(Exception e){}
```

この作業で，プログラムは ilpserver という名前でサーバーを立ち上げる．この時，同時に，そのサーバー名がレジストリーに登録される．

3.2 クライアントの設計

ここで定義した ilpserver を呼び出す場合は，クライアント側のプログラムで以下のように宣言する．

```
try {
    String url
       = "rmi://hostname:8888+"/ilpserver";
    Server server = (Server)Naming.lookup(url);
} catch(Exception e) {}
```

これにより，サーバー側のメソッドの使用が可能となる．hostname には ilpserver が立ち上がっているホストの名前が入る．

4 システムの実装

本システムの分散構造は図 2に示す通りである．

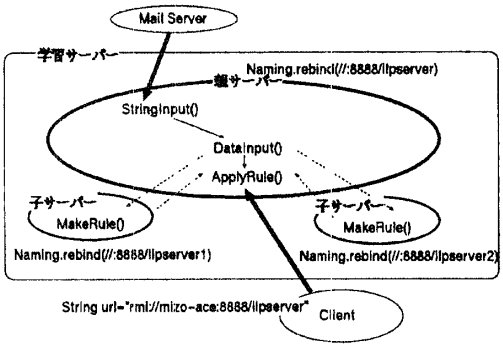

図 2: 分散構造

図の太めの実線は，他サーバーとのアクセス，点線は子サーバーとのアクセスを示している．

4.1 学習サーバーの実装

学習サーバーに準備されているメソッドは，DataInput, StringInput, MakeRule, ApplyRule の 4 つである．このうち，DataInput は，サーバー内部で使われるメソッドで，サーバーへのデータ入力に使われる．データ格納の様子は下に示した通りである．

```
private void DataInput(String data){
  StringTokenizer st1
    = new StringTokenizer(data,"\t");
  String MAILID = st1.nextToken();
  String SUBJECT = st1.nextToken();
  email.assert("subject", MAILID, SUBJECT);
}
```

DataInput メソッドは StringInput メソッドを通じてリアルタイムに送られてくるデータに対し，メール番号，ドメイン名，サブジェクト等に分割し，assert メソッドにより，email という名前でデータを格納する．

MakeRule メソッドと ApplyRule メソッドはそれぞれ，学習ルールの生成と適用に使われ，MakeRule はクライアント（ユーザー）から，また，ApplyRule はメールサーバーの方から呼び出される．

```
public void MakeRule(String rule,String target){
    ilp = new ILP(email, target, "+#");
    ObjectSave(ilp.induce(),rule);
}
```

学習ルールは新規メールの数にあわせて何度も使用するので，ここでは，ルール作成と同時にサーバー内部でオブジェクトとしてファイルにセーブされる．

ApplyRule メソッドは，MakeRule メソッドをで生成されたルールをもとに DataInput メソッドを通じてリアルタイムに流れてくるデータに対して，そのメールの分類先を提示する．

4.2 処理の分散

本システムにおける処理の分散化は学習エンジン部で実現する．本稿では学習のかけ方に注目し，各ターゲットごと，あるいは各属性ごとの分散学習を行なう．

これは，学習の親サーバーにデータを取り込む際に各属性ごとに振り分けてデータを格納し，データオブジェクトを子サーバーに渡すことで実現する．これにより，非常に時間のかかる作業である学習を短時間で完了させることを可能にする．

5 おわりに

Java の ライブラリーである RMI を用いて，分散ネットワーク型の電子メールエージェントを設計した．これにともなう処理やサービスの分散化により，マシンパワーに依存する必要がなくなった．Java 言語を用いることで，以前作ったシステム [1] のように，OS や，その環境に依存することがなくなった．

参考文献

[1] 溝口文雄，大和田勇人，矢口敬正，小川浩司，Personalized Mail Agent（その 1）- 設計方針 -，情報処理学会第 53 回全国大会,(1996)