マルチメディア通信と分散処理　97－10
コンピュータセキュリティ　8－10
（２０００．３．２１）

分散環境における透過的なプログラム記述法とD'Agentを用いた実行環境

本田 治　　多田 知正　　樋口 昌宏

大阪大学 大学院基礎工学研究科

スクリプト言語は，プログラムを簡潔に記述できることが利点である．分散環境で遠隔計算機上の資源を利用するスクリプトを記述する際，既存のクライアント/サーバモデルや移動エージェントモデルに基づいたプログラミングでは，資源が同一計算機上に存在する場合とは異なったプログラミングが必要である．このため，スクリプトの記述が煩雑になり，スクリプト言語の利点を損なっている．本稿では位置透過性を持つ分散プログラムの実行環境を提案する．提案する環境では，全ての資源が単一計算機上に存在する場合と同様のスクリプトを記述することができる．また，この環境を移動エージェントシステム D'Agent を用いて実装しスクリプトを記述し実行した．

Transparent programming in distributed computing systems and its implementation with D'Agent

Osamu Honda　　Harumasa Tada　　Masahiro Higuchi

Graduate School of Engineering Science, Osaka University

Ease of programming is the main merit of script languages. In distributed computing systems, we usually use Client/Server model or Mobile Agent model to write a script that accesses some remote resources. That is, we have to write a different script from that accesses only local resources. It brings some complexity which spoils the merit of script languages. In this paper, we propose writing scripts transperently in distributed computing systems. We implemented a system which executes such scripts. Using the system, we can write scripts which access distributed resources in the same way as those access only local resources. The system is constructed on the D'Agent mobile agent system.

1　まえがき

スクリプト言語は，比較的小規模のプログラムを記述する際に、すばやく簡潔に記述できることから広く用いられている．分散環境において，スクリプト言語を用いて遠隔計算機上の資源を利用する場合，クライアント/サーバ (C/S) モデルに基づくプロセス間通信が用いられてきた．しかしながら C/S モデルでは，クライアントとサーバの通信に関する煩雑な記述が必要であり，プログラムが簡潔に記述できるという利点が損なわれている．また，近年研究されている移動エージェントモデルでは，通信に関する記述は必要ないが，エージェントがいつどの計算機に移動するかをプログラムに記述しなければならず，資源とその資源を利用するプログラムが，同一計算機上に存在する場合とは異なるプログラミングが必要である．一方，分散環境全体を一つの仮想的な計算機とみなすモデルにおいては，プログラマは資源を保持する計算機を意識する必要がなく，プログラムを容易に記述できる．本研究ではこの利点に着目し，このモデルに基づいて記述されたスクリプトの実行環境を，既存の移動エージェントシステムの上に実装した．

実装したシステムでは，スクリプト中に明示的な移動のための命令を記述しない．遠隔計算機上の資源へのアクセスが発生した場合は，システムが資源へのアクセスの方法を動的に決定し実行する．この際，状況に応じてエージェントが資源の存在する計算機に自動的に移動する．このシステムの適用例として、遠隔計算機上のデータベースへのアクセスをとりあげ，C/S モデルや移動エージェントモデルと比べて，記述が容易に行えることを示した．また，記述したスクリプトの実行時間を測定し，実用的な時間で動作することを確認した．

以下，2節では分散環境とスクリプトについて，3節では，従来の分散プログラミング環境の問題点について，4節では，提案する分散プログラミング環境について，5節では，作成したシステムの評価について述べる．

2 準備

2.1 スクリプト言語

スクリプト言語は，比較的小規模なプログラムを短期間で開発することを目的としたプログラミング言語であり，通常インタプリタにより解釈実行される．Perl，Tcl，Python等が知られている．スクリプト言語は主に次の3つの特徴を持つ．

- プログラミングが容易
  変数の宣言(名前，型)が不要であり，プログラム中での自由な変数の作成や，変数の記憶領域を意識せずにプログラムが記述できる．また，UNIXのコマンド，豊富な組み込み関数，正規表現等が利用可能であり，便利な機能が多数存在する．
- 実行速度が遅い
  実行時にプログラムを解釈実行するため，機械語に翻訳したプログラムを実行するコンパイラ言語と比較して実行速度が遅い．しかしながら，今日の計算機の実行速度の向上により，多くの仕事は実用的な時間で実行可能である．
- 異なるアーキテクチャの計算機でも実行可能
  機械語に翻訳されていないため，計算機の種類やOSに依存する部分が存在せず，プログラムを変更することなく動作させることができる．

2.2 分散環境

複数の計算機が通信リンクで互いに接続されている環境を分散環境と呼ぶ．分散環境上の各計算機は，いくつかの資源 (resource) を保持している．資源とは，何らかのサービスを提供する物理的または論理的実体である．資源を直接利用できるのは，その資源を保持する計算機上に存在するプロセスのみであるとする．また，分散環境で遠隔計算機上の資源を利用するプログラムを作成することを，分散プログラミングと呼ぶ．

2.3 クライアント/サーバモデル

クライアント/サーバモデルは，分散環境で遠隔計算機上の資源を利用するための，一般的なモデルである．サーバとは，受動的に資源の機能を用いてサービスの提供を行うプロセスであり，クライアントとは，能動的にサーバに対してサービスの提供を促すプロセスである．クライアントは資源を利用する際，プロセス間通信を用いてサーバに対してサービスの要求を行う．サーバはそれを受けて資源にアクセスし，結果をクライアントに返す．

2.4 移動エージェントモデル

近年多くの移動エージェント技術に関する研究が行われている [1, 2, 3, 4, 5, 6, 7, 8]．移動エージェントとは，計算機間の移動が可能なプログラムであり，移動先の計算機で，移動前の計算機で行っていた仕事の続きを行うことができる．移動エージェントの移動の仕組みとその性質は次の2種類が存在する [5]．

- 強移動性
  プログラムのコードとプログラムの状態 (変数の値，プログラムカウンタ，スタックの内容等) を伴って計算機を移動し，移動先の計算機で移動に関する命令の次の命令から実行を続ける．
- 弱移動性
  プログラムの状態を移動する代わりに，プログラマが指定した情報を持って移動する．移動後は，プログラムの先頭から実行が再開され，移動時に伴った情報をもとに仕事の続きを行う．

移動エージェントには次に挙げる利点が指摘されている [3, 5]．

- プログラムの実行時間の短縮
- 通信量の削減
- 回線を維持する必要がない
- 通信に関する記述が不要

3 従来の分散プログラミング環境

3.1 クライアント/サーバモデル

スクリプト言語を用いて分散プログラミングを行う場合，通常クライアント/サーバモデルに基づくプロセス間通信が用いられている．PerlやPythonでは，ソケットを用いたプロセス間通信の仕組みを提供しているため，クライアント/サーバモデルに基づくプロセス間通信を用いた分散プログラミングが可能である．クライアント/サーバモデルに基づいて遠隔計算機上の資源を利用する場合，クライアントとサーバのプログラムを別々に記述しなければならない．また，クライアントとサーバ間のプロセス間通信のための

表 1: 既存のエージェントシステム

システム名	組織名	エージェント記述言語
D'Agent[1]	Dartmouth 大学	Tcl
TACOMA[3]	Tromosø大学	Tcl
Mole[6]	Stuttgart 大学	Java
Ara[4]	Kaiserslautern 大学	C,C++,Tcl
Telescript[7]	General Magic 社	Telescript
Sumatra[8]	Maryland 大学	Java

記述が必要である．プロセス間通信を行うためには，ソケットの作成，ポートの設定，ソケットとポートのバインド，バッファの処理等が必要になる．これらによってプログラミングが煩雑なものとなり，スクリプト言語の利点が損なわれている．

3.2 移動エージェントモデル

クライアント/サーバモデルとは異なる分散プログラムのためのモデルとして，移動エージェントモデルが存在する．移動エージェントモデルは近年活発に研究されており，多くの移動エージェントシステムが開発されている．これまでに実装されている主な移動エージェントシステムを表 1に示す．これらのシステムのうち，移動エージェントを記述するための言語がスクリプト言語であるのは，Tcl を採用している TACOMA，D'Agent，Ara と，Telescript を使用している Telescript である．移動エージェントモデルに基づいてプログラムを記述する場合，プロセス間の通信に関する記述が不要なため，スクリプトの記述が，クライアント/サーバモデルの場合に比べて簡潔になっている．しかしながら，プログラマは，プログラムコードのどの部分がどの計算機で実行されるかを意識する必要があり，また，エージェントのシステムへの登録などの処理も記述しなければならない．さらに，弱移動性をもつエージェントシステムでは，プログラマが移動後に実行の続きを行うための処理を記述する必要があるなど，移動エージェントモデルに基づいてスクリプトを記述する場合も，プログラミングが煩雑になり，スクリプトの利点が損なわれている．

4 位置透過性を持った分散プログラムの実行環境

3 で述べたように，従来の分散プログラミング環境は，スクリプトの記述に適しているとは言えない．分散計算機環境において，スクリプト言語を用いたプログラムを記述する際，スクリプト言語の利点である記述の容易さを損なわない分散プログラムの実行環境の作成を目標とする．

4.1 仮想単一計算機モデル

分散環境のモデルとして分散環境全体を 1 つの仮想的計算機とみなすモデルを考えることができる．以降では，このモデルを仮想単一計算機モデルと呼ぶ．仮想単一計算機モデルでは，ユーザは，分散環境内の資源の位置を意識する必要がない．このような性質を位置透過性という．位置透過性を持つ分散環境では，遠隔計算機上の資源を利用するプログラムを記述する場合に，単一計算機上に全ての資源が存在する場合と同様のプログラムを書くだけでよく，分散プログラムを容易に記述することができる．

4.2 実行環境

仮想単一計算機モデルにおいては，遠隔計算機上の資源へアクセスを行うプログラムを記述する場合にも計算機や資源の位置とアクセス方法を指定しないため，遠隔計算機上の資源へのアクセス方法を，スクリプトの実行系が決定する必要がある．資源へのアクセス方法の決定については，スクリプトの実行前に静的に決定する方法と，スクリプトの実行時に動的に決定する方法が考えられる．本研究では，資源へのアクセス方法の決定を，システムが動的に決定する方を採用した．これにより，資源へのアクセスが発生した時点での分散環境の状況またはスクリプトの実行状況に適した資源アクセスの方法を実行系が選択でき，より効率のよい処理を行うことが可能となる．

4.3 スクリプトの記述

3 台の計算機上のデータベースにアクセスし，アクセス結果がある基準を満たすまでアクセスを続けるプログラムを，D'Agent で記述した場合と，本研究で作成したシステムで記述した場合の例を表 2と表 3に示す．D'Agent では，エージェントの登録や，一度移動したエージェントは標準入出力が使用できないために，エージェントが実行を開始した計算機上に，結果出力用エージェントを残さなければならないなど，煩雑なプログラミングが要求される．一方，本研究で実装したシステムではこれらの記述が必要なく，簡潔なプログラムになっている．

表 2: D'Agent を用いたプログラム例

```
#!/bin/sh
# restart with the Agent Tcl interpreter \
exec /usr/src/dagent/agent2.0/agenttcl/bin/ \
agent "$0" "$@"
proc subagent {m1 m2 m3} {
   global agent
   set list ""; set jump_machine $m1
   while 1 {
      agent_jump $jump_machine
      access database "Query"
      if {[judge $result] == 1} {
         set jump_machie $m2
      } elseif {[judge $result] == 2} {
         set jump_machie $m3
      } elseif {[judge $result] == 3} {
         set pre_m $agent(actual-server)
         agent_jump $m1
         catch {access Database "Query"} result
         set list "$list $result"
         agent_jump $pre_m
      } elseif {[judge $result] == 4} {
         set list "$list $result";
      } else {break;}
   }
  return $list
}
puts "first machine name";gets stdin m1
puts "secound machine name";gets stdin m2
puts "third machine name";gets stdin m3
seet code catch {security signatures on}result]
if {[catch {agent_begin}]} {
   return -code error "ERROR"
}
   agent_submit $agent(local-ip) -vars m1 m2 m3 \
-procs subagent -script {subagent $m1 $m2 $m3}
   agent_receive code message -blocking
   puts\ $message
}
agent_end
```

4.4 実装

4.4.1 アーキテクチャ

分散環境全体を 1 つの仮想的計算機とみなすモデルに基づいたシステムを実現するため，移動エージェントの仕組みを利用した．既存の移動エージェントシステムには移動に関する命令が存在する．しかし，本システムにおいてユーザの記述するスクリプトには，移動に関する命令は存在しない．エージェントの移動は，あらかじめ与えられた移動方針に従って，システムが自動的に移動の必要性の有無と移動先を判断して行う．実装した移動方針については後述する．ユーザーは，あらかじめ与えられた移動方針の中から，必要な移動方針のみを選んだり，その設定を変更することができる．また，あらかじめシステムに与えられている移動方針以外にも，新たな移動方針をシステムに追加することができる．

表 3: 実装したシステムを用いたプログラム例

```
puts "first machine name";gets stdin m1
puts "secound machine name";gets stdin m2
puts "third machine name";gets stdin m3
set list "";set now_machine $m1
while 1 {
   access database "Query" at $now_machine
   if {[judge $result] == 1} {
      set now_machie $m2
   } elseif {[judge $result] == 2} {
      set now_machie $m3
   } elseif {[judge $result] == 3} {
      access database "Query" at $m1
      set list "$list $result"
   } elseif {[judge $result] == 4} {
      set list "$list $result";
   } else {break;}
}
puts $list
```

本システムでは，ユーザの記述したスクリプトが直接エージェントとして実行されるのではなく，スクリプトの実行系によって移動方針と共にデータとして保持され，移動エージェントであるスクリプトの実行系が，移動方針に従って動的な判断を行いながらスクリプトの内容を順次取り出して実行する．

4.4.2 基礎となる移動エージェントシステム

本システムの基礎となる移動エージェントシステムとして，D'Agent を採用した．D'Agent は Dartmouth 大学の R. S. Gray らによって開発された移動エージェントシステムである．本研究で D'Agent を採用した理由を次に挙げる．

- 強移動性を持つ
  弱移動性を持つ移動エージェントシステムでは，エージェントの移動の際，ユーザが実行の続きを行うための処理を記述する必要がある．よって，目標のシステムを実装するためには，システムの実行系が強移動性をシミュレートしなければならず，強移動性をもつ移動エージェントシステムの方が，実装するシステムの基礎として適している．
- エージェント記述言語が Tcl である
  スクリプトを記述するためのシステムの実装が目標である．よって，基礎となるシステムが持つスクリプトのインタプリタを利用することで，システムの実装の容易に行える．

4.4.3 エージェントの移動方針

実装した移動方針は次の2つである.

1 遠隔計算機上の資源にアクセスする場合
一度目の資源アクセスでは，移動エージェント自身が移動する代わりにサブエージェントを作成して計算機を移動させ，資源にアクセスさせ結果を送らせる．その計算機上の資源へのアクセスが連続すると，計算機を移動する．これにより，資源に対し交互にアクセスが発生した場合，移動エージェント自身が交互に移動する場合に比べ，サブエージェントはサイズが小さいため，通信量を削減することができる.

2 計算機のCPUが過負荷な場合
現在，エージェントが存在する計算機でCPUの利用率が一定以上になると，最初にエージェントの実行を開始した計算機へエージェントを戻す．これにより，計算機への負荷の集中を防ぐ.

4.4.4 実行系の記述

システムは，D'Agentのプログラミング言語である拡張版Tclで記述されており，スクリプトの実行部分となる移動エージェントと，その移動エージェントを作成する親のエージェントから成る．D'Agentでは，一度移動したエージェントはコンソールとの接続が切れるため，標準入出力を扱うことができない．よって，親エージェントが，最初にスクリプトが実行された計算機上で，コンソールへの標準入出力を移動エージェントの代わりに行う.

このシステムのプログラムは，全体で1200行程度(50KB)である.

5 評価

5.1 資源の指定

本研究で実装した実行環境では，スクリプト中で資源を指定する際には，資源の名前と資源を保持している計算機の名前を記述する必要がある．この場合，プログラマは分散環境上の資源の位置を意識する必要があるため，厳密な意味では位置透過性を持つとはいえない．しかし一般に分散環境においてスクリプトを記述する際，アクセスする資源を保持している計算機はあらかじめわかっていることが多く，計算機名をスクリプト中に記述することによって，プログラマの負荷が大きく増大することはないと考えられる．また，資源を保持する計算機をスクリプト中に記述しても，スクリプトの構造そのものは変化しないため，スクリプトを簡潔に記述できるという利点を損なうことはない．将来，分散環境で場所などを意識することなく資源を利用するための仕組みが実現されれば，完全に資源の位置を意識することなくスクリプトを記述することができ，真に位置透過性を持つプログラミング環境が実現できる.

表 4: 実験環境

CPU	PentiumII 300Mhz
ネットワーク	イーサネット (10bps)
OS	Linux
データベース	PostgreSQL

表 5: プログラムの実行時間とその内訳

処理内容	時間 (ミリ秒)
全体	1243
エージェントの登録	249
エージェントの移動	495
データベースへのアクセス	98
他の全ての命令	401

5.2 実験内容

4で述べたシステムを用いて，遠隔計算機上に存在する関係データベースにアクセスを行うプログラムを作成し，そのプログラムの実行時間を測定した．作成したプログラムは，遠隔計算機上の関係データベースに対してアクセスを行い，結果を出力するプログラムである．また，実験環境は表4に示すとおりである．値の測定は，一つの項目に対し100回の実験値の平均を取った.

5.3 実験結果

得られた実験結果を表5に示す．また，実験に使用したスクリプトと同じ動作をするプログラムを移動エージェントとして記述し，実行時間を測定した．その結果，281msであった.

5.4 考察

実際の実行時間を測定した結果，当初予測したよりも実行速度の低下は大きかった．D'Agentのエージェント移動機構をそのまま利用するために，実行系そのものをインタプリタ言語であるD'Agentのエージェントとして記述したことが，実行時間の大幅な低下の原因となっている．また，同じデータベースへのアク

セスを，D'Agent のエージェントとして直接記述した場合も 281ms の時間を要するように，D'Agent 自体の実行速度も Perl 等と比べて決して早いとは言えない．実行系の移動時の通信量も問題である．実行系の大きさは 50KB 程度あり，これだけの大きさのプログラムがエージェントとして移動するのはそれ自体かなりの通信量となる．データベースへのアクセスを行うスクリプトでは，データベースへのアクセスが連続した場合に，データの転送にかかる通信量を削減するために，スクリプトを含んだ実行系そのものが移動するが，一般にテキストデータの場合には，実行系の移動のための通信量がデータの転送のための通信量を逆に上回ることが多い．このような場合はデータベースへのアクセスが連続する場合にも実行系は移動しない方が望ましい．しかしながら，画像や映像，音声といったマルチメディアのデータでは，50KB を大きく上回るものが数多く存在する．PostgreSQL でマルチメディアデータを扱うのが難しいため，実験はテキストデータを対象として行ったが，マルチメディアデータを扱う状況では，実行系が移動することが通信量の削減につながることが多いと思われる．

本研究で作成したシステムの最大の目的は，分散環境において，スクリプトを容易に記述することである．作成したシステムでは，4.3で示す様に，全ての資源が同一計算機上に存在する場合と同様のプログラムを記述するだけでよく，システムの目的は達成されているといえる．実際に実行時間が問題とならないようなある種のアプリケーションにおいては，例えば，自動回覧板プログラムなどでは，十分に実用に耐えるものであると思われる．作成したシステムの実用性を向上するためには，スクリプトの実行速度をあげることが必要であると考えられる．具体的な方法としては，D'Agent のエージェント移動機構そのものを変更して，記述したスクリプトを直接実行できるようにすること．あるいは実行系を D'Agent から分離して実装し，必要最小限のエージェントを実行時に動的に生成して D'Agent に渡すようにすることなどがあげられる．

6 まとめ

本論文では，分散環境で計算機や資源の位置を意識せずにスクリプトを記述できる，位置透過性を持った分散プログラミング環境とその環境について述べ，それを実現するためのシステムを作成した．実装したシステムでは，プログラマは，遠隔計算機上の資源にアクセスする際に，全ての資源が単一計算機上に存在する場合と同様のプログラムを記述するだけで，遠隔計算機上の資源が利用できるため，分散環境でのスクリプトの記述が容易になる．また，スクリプトの実行速度は低下するものの，実用的な時間でスクリプトが実行されることを示した．

今後，より分散環境に適した移動方針の考案と実装を行う予定である．例えば，移動先で使用するかどうかわからない巨大なデータや手続きは実行系と一緒に移動させず，データ管理用サブエージェントにデータを管理させて，必要になれば取り寄せるという手法により，通信量を削減することができると思われる．別の課題として，システムの実用性を向上させるため，セキュリティを考慮にいれたシステムの実装や，スクリプトを記述する言語として，Tcl 以外の言語を実装することなどが挙げられる．

参考文献

[1] R. S. Gray : "Agent Tcl: A transportable agent system. In Proceedings of the CIKM Workshop on Intelligent Information Agents", Fourth International Conference on Information and Knowledge Management, 1995

[2] D. Chess, B. Grosof, C.Harrison, D. Levine, C. Parris, and G. Tsudik : "Itinerant Agents for Mobile Computing",IEEE Personal Communications, pp.22-49, 1995

[3] D. Johanson, R. V. Renesse, and F. B. Schneider : "An Introduction to the TACOMA Distributed System Version1.0", Technical Report 95-23, Depth. of Computer Science, Univ. of Tromosø and Cornell Univ., Tromosø, Norway, June 1995

[4] H. Peine : "An Introduction to Mobile Agent Programing and the Ara System", ZRI-Report, january 1997

[5] A. Fuggetta, G. P. Picco, and G. Vigna : "Understanding Code Mobility", IEEE Tansactions on Software Engineering, Vol. 24, No. 5, May 1998

[6] M. Straßer, J. Baumann, and F. Hohl : "Mole - A Java Based Mobile Agent System", Object-Oriented Programming ECOOP'96, M. Mühlaüser, ed., pp. 327-334, july 1996

[7] J. E. White : "Telescript Technology : Mobile Agents", Software Agents, J. Bradshow, ed. AAAI-Press/MIT Press,1996

[8] A. Acharya, M. Ranganathan, and J. Saltz : "Sumatra : A Language for Resource-Aware Mobile Programs", Vitek and Tschudin, 1997