分散システム運用技術　8－12
（1997．11．21）

自律移動型プログラムを用いた計算機管理支援の枠組について

川上 貴士＊　　石田慶樹†　　古川 善吾‡　　牛島和夫＊

＊九州大学大学院システム情報科学研究科
†九州大学大型計算機センター
‡九州大学情報処理教育センター

分散システムの導入により、利用者に対する利便性が向上する一方で、計算機を管理している管理者に対する負荷が増加している。管理者の負荷を軽減する方法として、作業代行プログラムによる管理作業の自動化と集中管理による作業の簡略化とがある。これらの両方の方法を採る際、管理者の手間の増加、通信・処理の負荷の集中が問題となる。

解決策として計算機間を自律的に移動するプログラムの利用を検討した。問題点がどの程度改善されるかを確認するために、移動するプログラムを利用したシステムを実際に試作した。その結果、作業内容を変更する際の管理者の手間が減少すること、作業実行時の通信・処理の負荷が分散すること、が確認できた。

A framework for managing distributed computer systems with an autonomous mobile program

Takashi KAWAKAMI*, Yoshiki ISHIDA†, Zengo FURUKAWA‡ and Kazuo USHIJIMA*

*Graduate School of Information Science and Electrical Engineering,Kyushu University.
†Computer Center,Kyushu University.
‡Educational Center for Information Processing, Kyushu University.

This paper discusses managing distributed computer systems and introduces a prototype of a management support system using an autonomous mobile program. The distributed computer systems may have some kinds of computer architectures. An administrator and a managing computer processes many works and the traffic between the managing computer and other managed computers may be a huge amount. A management support system using an autonomous mobile program may solve these problems. The management system is constructed of a management part on a managing computer, an autonomous mobile program and execution environments on managed computers. A prototype system is developed on a Java programming environment and an experiment shows effectiveness of the prototype for managing distributed computer systems.

1. はじめに

計算機の普及と計算機間の通信手段の発展により、計算機をネットワークで接続した分散システムが広く利用されている。分散システムの導入により利用者に対する利便性が向上する一方、システム管理者に対する負荷が増加している。

分散システムの管理では作業を簡略化するために、集中的な管理を行なうことが多い[1]。集中管理の場合、管理情報や負荷が管理者あるいは管理用計算機へ集中することが問題となる。また、計算機による管理作業支援として、作業代行プログラムの実行により作業を自動化する方法が考えられる。しかしながら、この方法には作業内容を変更する際の手間が大きいという問題がある。

そこで本研究では、作業代行プログラムを利用して集中的な管理を行ないつつ、前述の問題点を解決する方法の一つとして、自律的に管理対象間を移動するプログラムを利用して計算機管理を支援するのための枠組を提案している[2]。本稿では、提案する方法により分散システム管理時の問題事項がどの程度改善されるかをシステムの試作を通して検討する。

2. 分散システムの管理と問題点

2.1 分散システムの特徴

分散システムには以下のような特徴がある。

- 複数の計算機が接続されている
  アーキテクチャの異なるPC/WSから構成されることが多い。
- 多様な構成をとる
  利用する目的や用途により、システムの規模や機種の構成が異なる。

管理者の立場からいうと、日常的な管理作業には手間がかからないことが望ましい。また、利用者やシステムの構成が変化し、管理すべき項目に変更が生じた場合、管理を行なうために必要な作業に対する変更点は極力少ない方が望ましい。そのため、管理支援の枠組としてはシステムの変化にあまり左右されないものが良いと考える。

2.2 管理の手順

管理する対象となる計算機を管理対象と呼び、管理情報の保持や管理対象への操作の実行を行なう管理者や計算機を管理主体と呼ぶ。

計算機管理に必要な作業は次の3つである。

1. 情報の入手
   管理主体から管理対象へ問い合わせ、管理対象に関する情報を管理主体側で把握する。
2. 判断
   入手した情報を元に、管理対象へ操作を行なう必要があるか、あるならばどのような操作を行なう必要があるか、を管理主体側が判断する。
3. 操作の実行
   管理主体から管理対象へ操作を実行する。

これらの手順を繰り返すことによって計算機管理を行なっている。この管理の手順を分散システムの管理に適用した場合、分散システムは複数の管理対象が存在しているため、管理主体から一つの管理対象に対して行なった一連の手順を他の管理対象についても同様に行なう必要がある。そのため、管理対象数が増加するにつれ、管理主体の負荷が増大するのは明らかである。

2.3 管理者の手間の軽減法と問題事項

管理者の負荷を軽減するには次の方法が考えられる。

- 作業代行プログラムの利用
  作業を代行するプログラムを管理対象側や管理主体側で実行し、作業を自動化することで負荷を軽減する方法。
- 管理対象の集中的な管理
  管理情報の把握や操作の実行を管理主体側で集中的に行ない、管理工数を削減することで負荷を軽減する方法。

しかしながら、これら両方の方法を採る場合には、次に挙げる問題事項が出てくる。

- 作業内容変更時の管理者の手間が大きい
  既存の作業内容を変更する必要がある場合や、新たに管理する内容を追加する必要がある場合、各々の管理対象上のプログラムを変更・追加する必要がある。
- 物理的な負荷が集中する
  集中管理を行なうため負荷の集中が起こる。管理主体側のプログラムから各管理対象側のプログラムへ情報の問い合わせや操作の実行を行なう際に、通信の負荷が集中する。また、管理対象に操作実行の通知をしたり、管理対象からの通知を受けるための処理の負荷が集中する。

- 同一のプログラムの利用が困難
  分散システムはアーキテクチャの異なる計算機から構成されるため、同一のプログラムを用いて管理するのは困難である。

3. 自律移動型プログラムの利用

前節で述べた分散システムの集中管理における問題事項への対策として、自律移動型プログラムの利用について検討する。

3.1 自律移動型プログラム

自律移動型プログラムとは、ある計算機から他の計算機へ自律的に移動し、移動先の計算機で動作可能なプログラムのことをいう。一般的にはmobile agent[3]やmobile code、ワーム[4]と呼ばれている。が、本稿では以下、移動型プログラムと呼ぶことにする。RPC(Remote Procedure Call)が他の計算機上ののプログラムを実行するのに対し、移動型プログラムはプログラム自体を他の計算機に送って実行する。

移動型プログラムの特徴を以下に挙げる。

- 実行コードと必要なデータが移動する
- 移動先計算機で実行される
- 移動先計算機の資源へアクセス可能である

3.2 計算機管理への適用

この移動型プログラムを計算機管理に用いた場合、前述の問題事項に対してどのような解決となり得るかを示す。

- 作業内容変更時の管理者の手間が大きい
  作業内容の変更・追加をするためには、1つの移動型プログラムを変更・作成するだけでよい。そのため、変更・追加時に要する管理者の手間を軽減することができる。
- 物理的な負荷が集中する
  管理対象の情報の入手・判断・操作の実行を移動型プログラムとして記述する。移動型プログラムは管理対象上で実行されるため、情報の入手や操作の実行のために発生する通信をなくし、処理の負荷を分散することができる。プログラムの移動のための通信が発生することになるが、管理対象間を巡回するように移動させることにより、通信の負荷を分散させることが可能である。
- 同一のプログラムの利用が困難
  移動先の計算機で動作可能な移動型プログラムを用いることで、同一のプログラムの利用が可能である。

4. システムの試作

移動型プログラムを利用することで、実際に問題事項が改善されることを確認するため、管理支援システムの試作を行なった。

4.1 実装方針

システムを試作する上でまず問題となるのは、移動型プログラムを記述する環境である。今回の試作では、言語レベルで実行コードの可動性を持つVirtual Machineによる実行方式を採り、実行コードがアーキテクチャに依存しないという特徴を持つ言語であるJavaを用いた。

次に、プログラムの移動にあたって、どの管理対象に移動すれば良いかを決定する必要がある。移動先の決定法として、各移動先で次の移動先を決める動的な決定法と、移動前にあらかじめ指定しておく静的な決定法が考えられる。動的な決定法の場合、操作を実行する必要がある管理対象全てでの実行がどの時点で終了したかを検知するのが困難である。そのため、静的な決定法を採ることにする。あらかじめ移動先を管理主体で把握しておくため、各管理対象から管理主体へ通知を行なうようにする。

また、巡回移動の形態に次の2つが考えられる。

(a) 実行が終了してから次の計算機へ移動
(b) 実行を始める前に次の計算機へ移動

(a)の形態の利点は、実行結果を保持して移動できることである。移動元での実行結果に基づく動作を移動先でとったり、実行結果をまとめて通知することができる。しかしながら欠点として、管理対象数が増加すると全管理対象での実行終了に時間がかかることが挙げられる。

(b)の形態の利点は、作業の実行を並列化し全管理対象での実行時間を短縮できることである。しかしながら欠点として、実行結果を個別に通知する必要があることが挙げられる。

今回の試作では、プログラムの記述の仕方により両方の移動形態をとることを可能にした。

4.2 構成

システムは図1に示すように3つの部分から構成される。

【移動部】 管理対象間を実際に移動する部分。
管理対象の計算機上で実行される実行コー

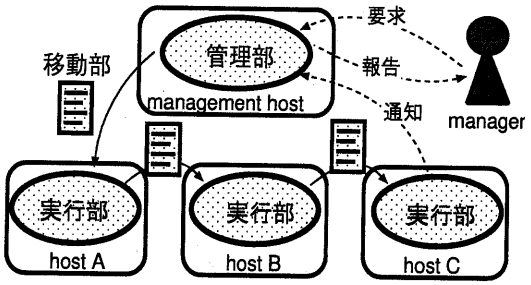

図 1: システムの構成

ドとデータから構成される。

【実行部】 管理対象の計算機上にあらかじめ配付されているプログラム。移動部を受け入れ、実行する。実行部は始動時に管理部へ自分の存在を通知する。

【管理部】 管理用計算機にあるプログラム。移動部と実行部の管理を行なう。管理者からの要求を受けつけ、また、管理者に管理部が持つ情報を提供する。

システム全体の動作を図 2に示す。図では縦の矢印が管理者およびプログラムの間のメッセージ通信を表し、横線がそれぞれの処理を表している。管理者が作業を行なう際は、作業を実行するプログラムを記述し、移動部として生成するよう管理部へ要求を出す。生成された移動部は移動を行ない、各管理対象上の実行部で実行され、実行結果を管理部へ通知する。管理部はこの結果を管理者に報告する。この報告には現在のところ E-mail を用いている。

また、データ部分の移動とプログラム部分の移動とは別々に行なわれる (図 2中 *DISPATCH* と *FETCH* に相当)。データ部分は毎回異なることが多いが、プログラム部分は同じものを用いることがあるためである。これにより無駄な通信の発生を抑えるようにしている。

4.3 実行例

次にこのシステムを用いた例として、管理用計算機上のあるディレクトリ下のファイルと管理対象上のファイルを比較して古ければ更新するという作業を行なった場合を見てみる。

作業を行なうためには、まず目的の作業を実行するプログラムを記述しなければならない。この場合「ディレクトリ下のファイルを調べ、ファ

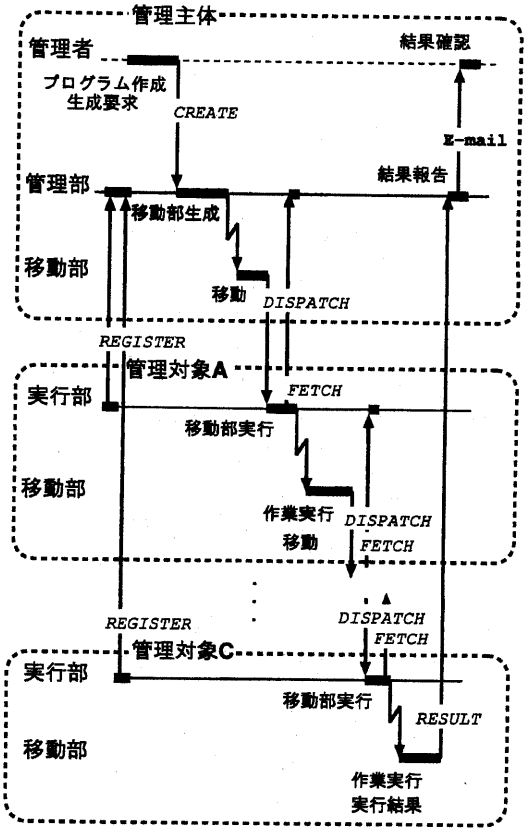

図 2: 動作の流れ

イルが古いか存在しないならば、該当するファイルを更新するという作業を対象となるファイル全てについて繰り返し実行し、次へ移動する」というプログラムを記述する (図 3は実行部によって実行される移動部のメソッドの記述である)。

```
public void doAction(){
  File file;
  for(;e.hasMoreElements();){
    file = (File)e.nextElement());
    if(compare(file)){
      update(file);
    }
  }
  setResult(s);
  next();
}
```

図 3: プログラムの記述例

管理部へ生成要求を出し作業を開始する。この後は移動部が各管理対象で作業を実行してま

```
<Id> 18
<From> kawakami@csce.kyushu-u.ac.jp
<Class> Distribute

<Results>
krishna.csce.kyushu-u.ac.jp:
Updated classes\dummy\Dummy3.class
Updated classes\dummy\Dummy2.class
Updated classes\dummy\Dummy1.class

crest.csce.kyushu-u.ac.jp:
Nothing to do

juno.csce.kyushu-u.ac.jp:
Updated classes/dummy/Dummy2.class
Updated classes/dummy/Dummy3.class

square.csce.kyushu-u.ac.jp:
Updated classes/dummy/Dummy3.class
Updated classes/dummy/Dummy2.class
```

図 4: 実行結果の報告

わり、結果を報告する。図 4は、 dummy の下のファイルを更新して回った実行結果の E-mail による報告である。

実行した作業はファイルの配付だが、移動型プログラムを用いて配付を行なうことで管理対象の状況に応じた動作、ここではまだ配付されていないファイルの配付を行なうという動作、をとることが可能になり、無駄なファイルの配付によって生じる通信を削減することができる。

5. 議論

5.1 問題事項の解決

どの程度問題事項が解決されているかについて検討する。

- 作業内容変更時の管理者の手間が大きい

  既存の作業に対する変更

  実行例に挙げたファイルを更新して回るプログラムについて、作業内容を変更する際の手間を考える。更新するファイルを変更する場合は、プログラムを実行する時のパラメータを変更するだけでよい。作業を実行する管理対象を変更するのに必要な作業は、移動部を生成する時に指定する移動先を変更するだけで済む。また、この例ではファイルの最終更新時間を比較して更新するかどうかを判定しているが、もっと詳細な情報を元に判定を行なうように変更することも考えられる。例えば、判定条件としてファイルサイズを追加する場合、変更に要した作業は、初期化部分のメソッドに1行、判定部分のメソッドに1 行、の合計2行を追加するだけであった。実行時のパラメータの変更や、1つのプログラムの部分的な変更で作業内容を変更できるので、管理者の手間は軽減できる。

  新しい作業内容の追加

  新しい作業を行なうためには、プログラムを記述する必要がある。しかしながら、何もない状態からプログラムの記述を行なうのは管理者にとって手間のかかる作業である。記述量を削減するためには、プログラムを再利用することが有効である。今回の試作で利用した Java はオブジェクト指向言語であるため、クラスの継承という形で再利用性の高いコードを記述することが可能である。有用なクラスを揃えることにより、新しい作業に対するプログラムの記述量を削減することができる。

- 物理的な負荷が集中する

  図 2の動作の流れから、作業実行時の計算機処理の負荷および通信の負荷は管理主体側に集中せずに分散していることがわかる。しかしながら、実行部が一斉に始動した場合、登録要求 (図 2中 *REGISTER*) の集中が発生する。また、実行結果の通知 (図 2中 *RESULT*) を個々に行なう場合にも、通知が集中する可能性がある。登録は1度行なえば良いため、2度目以降は通知しないことで回避可能である。実行結果の通知は必要時以外行なわない、後から結果を集めて回る、という対策を講ずることで集中を避けることができる。

- 同一のプログラムの利用が困難

  計算機のアーキテクチャ依存の実行コードではなく、Virtual Machine により実行される中間コードの利用により、異機種環境でも同一の実行コードで動作可能なようにした。しかしながら、そうすることによるデメリットも存在する。動作内容として最大公約数をとることになるため、プログラムとして記述可能な内容が限られてくることである。記述が困難な内容については外部のプログラムを利用し、そのプログラムへのインターフェースを用意することによって、共通の枠組を提供できる。

5.2 特徴に基づく用途の分類

作業代行プログラムとしてここでは移動型プログラムを議論したが、他に、事前に管理対象に配布したプログラムを利用する方法が考えられる。そこで、固定配置したプログラムを利用して管理作業支援を行なう方法(固定型)と、移動型プログラムを利用して管理作業支援を行なう方法(移動型)の違いを検討する。プログラムの存在する場所に着目し、固定配置を以下の2つに分類した。

(a) 管理対象側のみにプログラムが存在する形
(b) 管理主体側と管理対象側の両方にプログラムが存在する形

それぞれの形の特徴を以下に挙げる。

(a) の形 (例: cron の利用)

- あらかじめ指示しておいた契機(時刻等)に作業を実行する
- 作業を実行する際に管理主体側との通信を行なう必要がない
- 実行する作業内容が固定である

(b) の形 (例: SNMP の利用)

- 作業実行の契機は管理主体から任意に指示できる
- 管理主体側のプログラムと管理対象側のプログラムが通信を行ない作業を実行する
- 管理対象側のプログラムが実行可能な作業の中から必要な内容を選択して実行することができる

以上の特徴から、どのような用途に適しているかを考える。まず (a) の形は、管理主体と通信する必要がなく作業内容が固定である日常的な作業に適していると言える。(b) の形は、管理対象からの情報入手および入手した情報に対する操作が固定である作業に適している。しかしながら、管理主体と集中的な通信を行なうため、一度に多数の管理対象への操作の実行には不向きである。

一方、移動型の場合は、管理主体と各管理対象との間の通信が削減できる、同じ実行コードを各管理対象で実行可能である、という特徴から、一度に多数の管理対象へ同じ内容の非日常的な作業を行なう場合に適している。例えばソフトウェア構成の管理は、柔軟な対応が求められるため、移動型を用いるのが良い。

5.3 課題

今後の課題を以下に列挙する。

- セキュリティ対策
  移動型プログラムにとってセキュリティ機能は必須である。悪意を持ったプログラムの実行による計算機上の資源の盗用や破壊を防ぐため、認証機構を実現する必要がある。これは公開鍵暗号による電子署名を用いるのが良い。また、悪意のない場合でもプログラムが暴走する可能性がある。暴走を制御するためのメッセージ通信機構を実現する必要がある。
- ユーザインターフェースの提供
  現状では管理者と管理部との間のインターフェースとしては、コマンドラインからのインターフェースしか備えていない。コマンドラインからでもネットワーク経由の操作を行なうことは可能であるが、他にもWWWやE-mailによるインターフェースの提供もネットワーク経由の操作の実現に有効である。
- 通信量の削減
  移動型プログラムは、実行コードの移動が起こるため通信量が多い。実行コードの通信を抑えることだけでなく、通信を圧縮して行なうことも通信量削減の対策として考えられる。

6. おわりに

移動型プログラムという、計算機間を移動可能なプログラムを利用して管理作業を行なうための枠組の試作を行なった。今後は課題として挙げた項目を実現していくとともに、ソフトウェアのインストール / バージョンアップ作業への適用について検討を行なっていく。

参考文献

[1] 川又滋, 有野康仁, 入倉信治, 池田和弘: 遠隔 PC 操作ツール WinShare の背景と実装, 情報処理学会研究報告 96-DSM-4,pp.61-66,1996.

[2] 川上貴士, 石田慶樹, 古川善吾, 牛島和夫: 移動型プログラムを利用した分散システムの管理について, 情報処理学会第 55 回全国大会講演論文集,pp.3-755-756,1997.

[3] 服部文夫: エージェント言語, コンピュータソフトウェア, Vol.14,No.4,pp.3-12,1997.

[4] J. Reynolds:The Helminthiasis of the Internet, RFC1135,1989.