計算機資源の意味を基本とした分散環境での資源管理

3E-1

中村 修
(株)リコー
ソフトウェア研究所

昭和63年 6月 14日

1 はじめに

コンピュータネットワークの普及に伴い、ソフトウェアやハードウェアなどの計算機資源に対する利用者のアクセスは、TSS などの場合に比べて遥かに複雑さを増している。TSS の場合には、計算機資源は、すべて1つのホスト計算機上で管理され、利用者は計算機資源の名前さえ覚えていればアクセスできるような環境であった。しかし、ネットワークを基本とした分散環境では、計算機資源の存在している場所は、複数の計算機システムに分散し、計算機資源の名前以外に資源の実体が管理されているホスト計算機名を知らなければ利用できない。また、機能的に同じような計算機資源が複数の場所に存在し、利用者は何が同じで何が違っているかを意識しなければ、これらの計算機資源を利用できなくなってきている。特に異機種間接続をしている場合などは、システムのアーキテクチャの違いによってオブジェクトを実行できるホスト計算機が制限されるなどの資源を取り巻く環境の違いを意識しなければならなくなってきている。

本論文では、まず分散環境での資源管理における重要な概念である意味の透過性を提唱する。次に意味の透過性を提供した資源管理システムの構築方法について述べる。

2 意味の透過性

分散環境における計算機資源の管理を行なう場合には、一般的に以下のような透過性が重要である。

- 場所の透過性
- アクセスの透過性

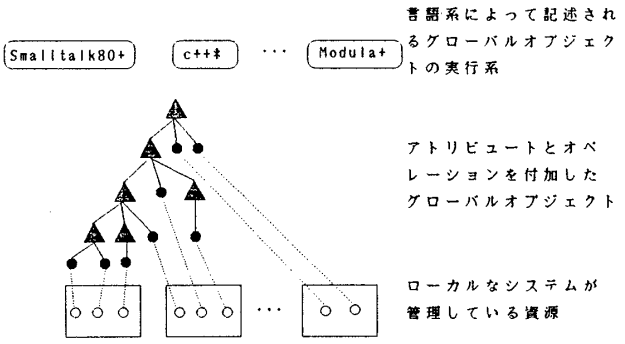
場所の透過性とは、利用者は計算機資源が管理されている場所を意識せずに利用できることであり、アクセスの透過性とは、計算機資源の管理されている場所に依存しないアクセス方法によって計算機資源がアクセスできることである。
例えば、ファイルのキャッシングによって実際にファイルがどこに存在しているかを利用者が意識しないですむ事を場所の透過性が提供されているといい、同じコマンドやシステムコールによってローカルなファイルもリモートにあるファイルもアクセスできることをアクセスの透過性が提供されているという[1].

現在のネットワーク環境では、NFS に代表されるようなネットワークファイルシステム[2][3] などによって場所の透過性やアクセスの透過性が提供されている環境は一般的になっていきているが、各々の計算機資源が持っている意味を捉えた資源管理は行なわれていない。しかし、実際に分散環境で計算機資源を管理する場合には、計算機資源をいろいろな角度から管理し利用者に提供する必要がある。

例えば、NFS では、ASCII で書かれたテキストファイルを異なったシステム間でファイルシステムの一部として共有し、互いにアクセスしても両者にとってファイルの内容が異なるというような問題はないが、オブジェクトファイルなどのようにシステムのアーキテクチャに依存したものは、ファイルシステムの一部として共有することは可能だが、実行することはもちろんできないし、利用者にとってそれが実行できるかできないかを知るすべはない。また、内容が同じでも異なった漢字コードで書かれたファイルは、今までのシステムでは異なったファイルとして扱われていた。しかし、何等かのコード変換フィルタを使って相互に変換し合うことができれば、内容という観点からは同じファイルとして扱うことができる。そこで、計算機資源の持っている意味を捉えた資源管理が重要となる。

計算機資源をいろいろな角度、すなわち、資源の持っている意味を捉えながら参照できるような環境のことを“意味の透過性” が提供されていると定義する[4]。例えば、ある角度から計算機資源を捉えたときに意味的に同じ物は、1つのものとして扱えるような環境が意味の透過性が提供されている環境ということになる。

"Semantic Based Resource Mamagement in Distributed Enviornment"
Osamu Nakamura
Ricoh Co., Ltd.



図一1 システムモデル

3 システムモデル

意味の透過性を提供した資源管理方法の1例を以下に示す。

各ローカルなサイトが管理している計算機資源を分散環境で扱えるようにするために各ローカルな資源をそのアトリビュートを内包したグローバルオブジェクトとして管理する。そして、このグローバルオブジェクトを扱う環境としてオブジェクト指向型言語[5]やModula-2[6]などの言語系を設定することによって分散環境で意味の透過性が提供された資源管理をおこなうことができる(図一1)。

グローバルオブジェクトは、ローカルな計算機資源がもっているアーキテクチャや利用者などをそのアトリビュートとして管理するとともに各資源に対するオペレーションも管理する。すなわち、グローバルオブジェクトは、ローカルな資源とそのアトリビュートをタイプとして、資源へのアクセス関数をオペレータとした1つのオブジェクト指向におけるオブジェクトとして捉えることができる。例えば、ローカルな計算機資源のアトリビュートとしてアーキテクチャという属性を管理することによって異機種な計算機環境でのオブジェクトファイルが持っている"実行できる"という意味やデータファイルのバイトオーダなどによるデータファイルの意味を捉えた資源の管理が可能となる。また、資源に対するオペレーションを個々の資源毎に定義することによって、テキストファイルを参照するときに、単純なバイト列としてアクセスする場合と、テキストとして意味を保持したアクセスを区別することが可能となる。

このようなグローバルオブジェクト環境は、上位レベルとしてこれらのグローバルオブジェクトを操作する実行系の基板となる。この実行系は、グローバルオブジェクトを基本構成要素とした言語系として実現することによってより良い環境を提供することができる。

4 おわりに

本論文では、分散環境における計算機資源の管理方法として計算機資源が持っている意味を捉えた資源の管理方法について述べた。各ローカルな資源が持っているアトリビュートとその資源へのオペレーションを管理することによって、ローカルな計算機資源をグローバルオブジェクトとして分散環境で統一的にアクセスできるようになる。そして、グローバルオブジェクトを操作する環境では、グローバルオブジェクトが持っているアトリビュートを利用することによって計算機資源が持っている意味を失わずにアクセスできるような環境を構築することが可能となる。

現在各ローカルな計算機資源をグローバルオブジェクトにするためのプロトタイプをSun Workstation上に構築中である。このプロトタイプでは、グローバルオブジェクトは、Unixのファイル毎にスレッドプロセスとして実現している。ローカルな資源が持っているどのような属性をアトリビュートとして利用すれば最も効果的に資源を管理でき、グローバルオブジェクト環境上の実行系がうまく動作できるかを現在研究中である。

参考文献

[1] J. Prais and W. F. Tichy, "STORK: An Expreimental Migrating File System for Computer Networks," *Proc. of IEEE INFOCOM*, pp 168-175, April 1983

[2] R.Sandberg, et al, "Design and Implementation of the Sun Network File System," *Proc. of USENIX Summer Conf.*, June 1985

[3] A. P. Rifkin, et al, "RFS Architecture Overview," *Proc. of USENIX Summer Conf.*, June 1986

[4] O. Nakamura and N. Saito, "Resource Mahagement Scheme in Dsitributed Environment," *Proc. of SIGCOM Conf.*, Oct 1987

[5] A. Goldberg, D. Robson, "Smalltalk-80 The Language", *Addison-Wesley*, 1983

[6] N. Wirth, "Programming in MODULA-2", *Speringer-Verlag*, 1983