マルチメディア通信と分散処理 71−28
（1995. 7. 14）

ネットワーク管理システムにおける分散オブジェクト管理

藤崎智宏　蔭山克禎　荒野高志
NTT ソフトウェア研究所 広域コンピューティング研究部

現在、分散オブジェクト指向プラットフォーム上でコンピュータネットワーク管理システムを構築中である。このネットワーク管理システムでは、分散環境上で複数のオブジェクトが協調して動作する。このため、ホストの障害などでオブジェクト間の通信が途絶えると、システム自体の動作に不都合を来たす。そこで、システムを安定動作させるため、管理システムを構成するオブジェクトの存在管理を行う必要がある。本システムでは、オブジェクトの存在を見張っているサーバオブジェクトを置くことでオブジェクトの存在管理を行なっているが、サーバオブジェクト自身の障害に対応できない。そこで、サーバオブジェクトがセカンダリサーバオブジェクトを管理する、という手法を用いた。これにより、サーバオブジェクトを管理対象オブジェクトとして同一視することが可能になり、さらにシステムの信頼性を向上させることができた。

Distributed object management in network management system

Tomohiro Fujisaki　Katsuyoshi Kageyama　Takashi Arano
NTT Software Laboratories, Global Computing Laboratory

In this paper, we proposes a new strategy about distributed object management. In a distributed object environment, checking for object existance is important especially there are any objects which act as server. Simple scheme for object management is to place an object-observation server which is responsible for objects existance. But in this scheme, maintainance for server itsself is a problem. We solve this problem by adding secondary-object observation server into primary's maintenance target. We have made object-observation server, and make sure that this shceme is useful.

1　はじめに

コンピュータネットワークは急激な発展を続けており、それに伴ってネットワーク構成機器が多様になる、異なるプロトコルを持つネットワークを相互に接続するなど、ネットワーク構成の複雑性が増している。複雑な構成のネットワークを容易に管理するために、ネットワーク管理システム (以下 NMS とする) を用いることができる。しかしながら、現状の多くの NMS が採っている、ネットワークを中央で集中的に管理する、という構成では大規模なネットワーク管理を行う際に性能的、安全性的に問題点が多い。更に NMS に対しては、個々の組織の環境に合わせてカスタマイズ、独自の拡張行う、ということが重要であるが、市販の NMS ではこういった要求に十分に答えることはできない。

上記の要求は NMS を“分散オブジェクト”技術を用いて構成することで解決することが可能である。この技術はオブジェクト指向の持つ拡張性、ソフトウェア作成への即応性、分散環境での実行メカニズム等を特徴とする。分散オブジェクト指向技術を用い、オブジェクトをネットワーク中に分散して配置することにより、負荷の分散を行うことができ

る。更に、分散オブジェクト機構の持つオブジェクトの位置透過性を利用することで耐故障性を上げることができる。

現在、分散オブジェクト指向プラットフォーム上でネットワーク管理システム“NetKeeper”を構築中である[Arano95]。このネットワーク管理システムでは、分散したホストに配置された複数のオブジェクトが協調して動作する。このため、ホストの障害などでオブジェクト間の通信が途絶えると、システム自体の動作に不都合を来たす。そこで、システムを安定動作させるため、管理システムを構成するオブジェクトの存在管理を行う必要がある。

本稿ではNMS“NetKeeper”において採用している、分散オブジェクトの管理手法について述べる。第2章では分散オブジェクト技術と、NMSに分散オブジェクト技術を適用する利点について、第3章では本NMSの構成と特徴について、第4章では本NMS中でのオブジェクトの管理機構について述べる。第5章ではオブジェクト管理上の問題点について述べ、最後に第6章にてまとめと今後の展開について述べる。

2 分散オブジェクト指向とNMS

本章では分散オブジェクトについて述べ、分散オブジェクト技術をNMSに適用することについての利点を述べる。

2.1 分散オブジェクト指向

分散オブジェクトとはオブジェクト指向の概念を分散環境に拡張したものである。従来のオブジェクト指向実行環境ではオブジェクトが一つの計算機中の一つのプロセス内に存在していた。分散オブジェクトの実行環境ではオブジェクトをネットワーク中に分散し、オブジェクトどうしがネットワーク中でメッセージを送受信することによりシステムが動作する。オブジェクトどうしのメッセージ送受信は、オブジェクトの“名前”とそのオブジェクトに対し動作を指示する“メソッド”を指定することにより行われ、オブジェクトが実際にどのマシン上にあるかを気にしなくてよい。このオブジェクトの位置透過性は、耐故障性の高いシステムの構築、動的なオブジェクト配置による負荷分散に利用できる。また、既存のsocekt等を用いたプロセス間の通信に比べ、非常に簡便に使用できるので、開発も用意である。

次節では、分散オブジェクト技術をNMSに適用する利点について述べる。

2.2 分散オブジェクト指向NMS

既存の多くのNMSでは、コンピュータネットワークの管理を集中的に行っている。また、カスタマイズできる幅が少なく、拡張することが難しい、といった問題点がある。

NMSに対しては、以下のような機能が求められる。

1. NMSをカスタマイズ・拡張したい
   組織の都合に合ったNMSを作りたい。
2. 分散環境を有効活用して管理したい
   ネットワークを有効活用し、特定の場所やマシンにとらわれずに管理をしたい。

分散オブジェクト指向技術の次に挙げる特徴を用いることで、上記1、2を実現できる。

1. オブジェクト指向による変更容易性
   オブジェクト指向で用いられるMVCモデル[Cuningham86]の利用により、管理アプリケーションとそのGUIを分離することでGUIのカスタマイズが容易である。GUIを作成する際には、あらかじめ用意されている部品を用いることができる。また、GUIのみでなく、部品を組み合わせることにより既存アプリケーション本体の変更も容易に行える。更に、アプリケーションとGUIの間は分散オブジェクト呼出という高レベルな通信機構を使うことができ、プログラミングが容易である。
2. 分散オブジェクトの位置透過性
   オブジェクトに名前だけでアクセスできるため、分散オブジェクト指向環境がカバーするネットワーク中のどこにいても、同じ環境でのオブジェクトアクセスが可能である。

他に、通信の最適化、システム構成の動的な変更等に対応、などの利点がある[福井95]。次章では、分散オブジェクト指向技術を用いて構築したNMS“NetKeeper”の構成について述べる。

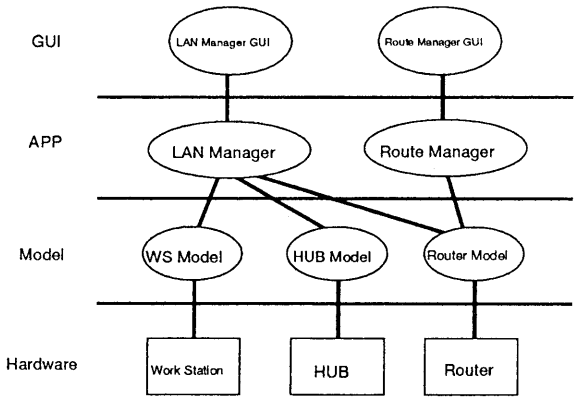

図 1: Structure of the NetKeeper

3 NMS NetKeeper の構成

NetKeeper の構造を図 1 に示す。

NetKeeper では管理対象であるネットワークを 4 層構造に分けてモデル化している。特徴的なのは、管理対象機器に対しその代理エージェントである“モデル”を構築し、モデルに分散オブジェクト呼出によるインタフェースを持たせるという構成をとっていることである [藤崎 95]。モデルは管理対象機器に 1 対 1 に対応して存在し、複数のプロトコルを用いて実際にネットワーク機器にアクセスするが、複数プロトコルの存在をを分散オブジェクト呼出で隠蔽する。

以下、各層について詳述する。

Hardware 層
ネットワーク上に存在するルータ、ハブ、ワークステーションなどの管理対象機器そのものの層である。

Model 層
管理対象となるネットワーク機器からの情報の取得、機器の制御を行う“モデル”の層である。“モデル”は各機器のエージェントであり、管理対象機器一台に対し一つ存在する。機器からの情報を保持し、各種設定の指示を機器に送る。

APP(Application) 層
Model 層に存在する管理対象機器のモデルにアクセスし、ネットワークの管理をするアプリケーションの層である。管理対象機器のモデルを組み合わせ、組織のネットワーク管理ポリシーにあったアプリケーションを構築できる。

GUI 層
管理情報の表示、機器への指示を人間が行う部分である。GUI とアプリケーションを切り離したことにより、ユーザに合わせて容易に GUI を変更できる。また、GUI を複数の場所で実行し、同じアプリケーションにアクセスすることで離れた場所で同一の情報を共有することができる。

Hardware 層と Model 層の間の通信は管理対象機器により使用するプロトコルが違う。更に、機器によっては複数のプロトコルによって通信する。このようなプロトコルの違いを吸収するために、モデルはプロトコルを選択して機器と通信する Protocol Handler Controller を持つ [蔭山 95]。

Model-APP, APP-GUI 層間の通信は分散オブジェクト呼出の機構を用いて行われる。これにより、各オブジェクトに対しオブジェクト名とメソッドの指定でアクセス可能であるため上位層からは下位層のオブジェクトがどのマシンで動作しているかを気にしなくてもよい。例えば、APP 層からは、Model 層にある各モデルはどこのマシンで動作するかは気にせずに通信できる。モデルが動作するマシンに障害が発生した場合でも、そのモデルを他のマシンに移し変えることにより、APP 層中のアプリケーションからは何事もなかったかのように管理を続けることができる。NetKeeper では、モデル、アプリケーションを管理対象オブジェクトとし、存在管理を行うことでシステムの耐故障性を上げている。

次章では、NetKeeper に導入した分散オブジェクト管理の手法について述べる。

4 NetKeeper におけるオブジェクト管理

NetKeeper を用いてネットワーク管理を安定して行うためには、オブジェクトの存在管理が必要である。本節では NetKeeper でのオブジェクト管理について述べる。

4.1 NetKeeperによるネットワーク管理の例

NetKeeper を用いてネットワークの管理をする例を示す。第3章で述べたように、NetKeeper では管理対象ネットワーク機器に1対1に対応したモデルを置き、管理アプリケーションがモデルにアクセスし、動作する。ネットワークの監視/管理者は、専用の GUI にアクセスしてネットワークの監視/制御を行う。それぞれのモデル/アプリケーション/GUI はネットワーク中のどこに存在してもよい。NetKeeper を LAN 管理に適用した場合の、モデル/アプリケーション/GUI の配置例を図2に示す。

個々のモデル/アプリケーション/GUI はぞれぞれオブジェクトとして認識することができ、分散オブジェクト呼出によりアクセス可能である。

4.2 NetKeeperでのオブジェクト管理

NetKeeper が使用する分散オブジェクトプラットフォームではネットワーク中に存在するオブジェクトに対し、オブジェクトの名前のみでアクセス可能であり、オブジェクトがどのホストで動作しているかを気にしなくてよい。この位置透過性により、ホストの障害等でモデルがダウンした場合でも別なホストでそのモデルを立ち上げることにより、管理アプリケーションはモデルに対してアクセスを続けることが可能である。

NetKeeper ではこの性質を利用し、ネットワーク中にオブジェクトの存在を管理する“オブジェクト管理サーバ”を置き、対故障性を向上するという手法を用いていた。このオブジェクト管理サーバはオブジェクトの存在を常に見張っており、必要なオブジェクトがネットワーク中から消えると、あらかじめ指定された優先度にしたがって決定したホスト上でオブジェクトを再起動する。オブジェクト管理サーバの実現により、オブジェクトの存在管理を行うことができるようになったが、この管理サーバでは以下のような問題点が発生した。

1. 管理プログラム自身が障害でダウンすることを考慮していない。
2. 管理対象オブジェクトを再起動する際に静的なリストを参照しているため、オブジェクトが特定ホストに集中しがちである。
3. “モデル”オブジェクトの管理に特化している。

上記の問題点を解決するため、

1. サーバ障害時の対策
2. 動的な情報に基づいたオブジェクト配置
3. “モデル”のみでなく、分散オブジェクトの管理を行うサーバ形式にする(管理対象オブジェクトの動的な変更の実現)

を目標として新たにオブジェクト管理サーバを設計し、実装した。次節で、新オブジェクト管理サーバについて述べる。

4.3 オブジェクト管理サーバ

サーバの障害対策

サーバクライアントシステムにおいて、サーバの障害への対処は大きな問題である。Domain Name Sytem (DNS) では、primary サーバに対して secondary サーバを置くことで障害対策をしている。primary サーバに障害が発生した場合には、secondary サーバに問い合わせ先を切り替えることでシステム全体は動作を続けることができる。同様の手法は、Network Information System (NIS) などでも用いられている。DNS, NIS では、secondary サーバは primary サーバの機能を完全に肩代りするものではなく、一部の機能を代理するようになっている。

NetKeeper でのオブジェクト管理にも同様の手法を用いることは可能であるが、この手法であるとクライアント側がサーバが落ちたことを検知し、アクセス先を secondary に切り替えることになる。これは、クライアント側が secondary サーバの名前を知らねばならず、更に、secondary サーバが複数あった場合の対応が難しい。すなわち、名前でオブジェクトにアクセスできるというオブジェクトの位置透過性の利点を損ねることになる。

そこで、新オブジェクト管理サーバでは primary サーバが落ちた場合、secondary サーバが primary サーバになりかわる、という手

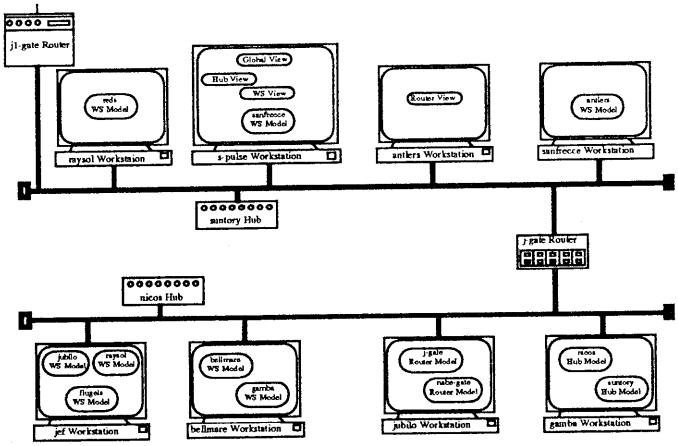

図 2: Example of object arrangement in NetKeeper

法を採った。この手法では、クライアント側はサーバが複数いることを認識する必要はない。さらに、primary サーバの管理対象オブジェクトに secondary サーバを加えた。primary サーバが立ち上がる際に分散オブジェクト名前空間をチェックする。サーバが管理対象オブジェクトで存在しないものを検知した場合には、そのオブジェクトを起動する。secondary サーバも管理対象オブジェクトとして指定されているため、障害で secondary サーバが primary サーバになった場合、secondary サーバが不在となるので再起動される、ということになる。これにより、primary サーバに障害が起こった場合の復旧をより確実にした。

新オブジェクト管理サーバの動作を図 3に示す。

secondary サーバは複数立ち上げることができる。この場合、最初に primary サーバの不在を検知した secondary サーバが primary サーバになる。分散オブジェクト機構の名前空間管理を利用し、同時に複数の secondary サーバが primary サーバにはならないことを保証している。

オブジェクトの動的配置

新オブジェクト管理サーバでは、管理対象オブジェクトを立ち上げるホストのを以下の情報を用いて決定する。

- ホストの CPU パワーの情報 (静的)
- オブジェクトごとに指定される優先度付の実行ホストリスト (静的)
- ホストの load average(動的)
- ホストに存在するオブジェクト数 (動的)

上記のうち、動的なデータは管理サーバが定期的に収集する。管理サーバは管理対象オブジェクトの不在を確認すると、上記データより実行ホストを決定してオブジェクトを起動する。

サーバとしての実装

管理対象オブジェクト登録 / 削除要求の受け付けを行う機構を実装し、動的に管理対象オブジェクトの変更を行えるようにした。

5 考察

- オブジェクトを起動する時に、ネットワークのトポロジや回線容量等を考慮することで、システム全体としての性能を向上させることができる [青野 95]。特に、大規模なシステムでは、オブジェクトの配置によって性能に大きな差が出て来ると考えられる。オブジェクト起動時のオブジェクト配置をよりインテリジェントにしていく必要がある。
- 本手法では、オブジェクトの不在の検知、競合の制御を分散オブジェクトプラット

```
main()
{
    if (primary サーバが存在する)
        secondary サーバ _main_loop();

    primary サーバ _main_loop();
}
secondary サーバ _main_loop()
{
    while (1) {
        if (primary が存在しない) {
            primary になる;
            break;
        }
        sleep;
    }
}
primary サーバ _main_loop()
{
    while (1) {
        管理オブジェクト
                追加 / 削除要求受付 &&
        管理対象オブジェクト
                存在チェック / 立ち上げ;
    }
}
```

図 3: Object-Observation server

フォームに大きく依存している。現在使用している分散オブジェクトプラットフォームは非常に安定しているが、NMS の信頼性をより高めるためにプラットフォーム自身の障害の管理を考える必要がある。

- 現在管理対象にできるオブジェクトには、オブジェクト自身が内部状態を管理しているもの、という制限がある。“モデル”では、その内部状態を起動時に対象ハードウェアから得る、という方法をとっている。今後、データベースとリンクした永続的オブジェクトの導入を行うことで、管理対象オブジェクトを容易に構築できるようにする必要がある。

- オブジェクトをホスト間で移動できれば、あらかじめわかっているホストのダウンの時に有用である。また、動的な負荷の分散をより細かく行うことができる。これも、永続的オブジェクトを導入することで、実現可能である。

6 まとめ

分散オブジェクトプラットフォーム上のアプリケーションが安定に動作するには、オブジェクトの存在管理が必要である。オブジェクトの存在管理の手法として、オブジェクト管理サーバを実装した。このサーバでは、primary サーバに障害が起きた時に secondary サーバが primary サーバになりかわる、という方式をとった。更に、存在管理対象オブジェクトに secondary サーバを加えることで、分散オブジェクトの存在管理の信頼性を向上させた。

今後、より進んだオブジェクト配置機能の実現、永続的オブジェクトの導入、オブジェクトマイグレーションの実現を行っていく。

参考文献

[Arano95] Takashi Arano et al. “A Computer Network Management System Platform based on Distributed Objects”, IFIP/IEEE DSOM ’95 (投稿中)

[青野 95] 青野 他 “性能を最適にするための分散オブジェクトの配置法 — 分散オブジェクト指向プラットフォーム上の NMS の実例より —” 電子情報通信学コンカレント工学研究会 95.5.19

[蔭山 95] 蔭山、他 “ネットワーク管理システムにおけるマルチプロトコルハンドリングについて”, 95 年電子通信学会総合大会

[Cuningham86] Ward Cunningham “Smalltalk-80 によるアプリケーションプログラムの作り方” bit, Vol.18 No.4, pp.379-395,1986. ソニー・テクトロニクス (株)AI 営業部 抄訳

[Rose94] MARSHALL T.Rose “THE SIMPLE BOOK second edition” Prentice Hall, 1994 ISBN 0-13-177254-6

[福井 95] 福井 他 “並行オブジェクト指向言語のユーザはだれか - ネットワーク管理への応用 -” WOOC ’95

[藤崎 95] 藤崎 他 “分散オブジェクト指向プラットフォームに置けるルータの運用・管理” 情報処理学会分散システム運用技術研究グループ研究会 95.01