マルチメディア通信と分散処理 58−18
（1992. 11. 20）

リソース指向分散環境の設計と実現

谷林 陽一　今井 功　佐藤 文明　勝山 光太郎　水野 忠則

三菱電機(株) 情報電子研究所

分散システムでの処理の分散と資源の集中管理はそれぞれ排反する側面を持ち，従来独立に検討が進められてきた．本論文では，処理の分散と資源の管理を協調させた分散環境としてリソース指向分散環境を提案する．この環境の特徴は，分散環境上で動作するオブジェクトに二つのインタフェースを持たせていることである．一つは，サービスインタフェースであり，従来の分散環境で実現されているオブジェクトが持つインタフェースである．もう一つが管理インタフェースであり，このインタフェースによって資源管理を含むアプリケーションの開発を容易にすることができる．本論文では，リソース指向分散環境のアーキテクチャ，基本設計，および実現と評価結果について論じる．

An Implementation of Resource Oriented Distributed Environment

Youichi TANIBAYASHI　Isao IMAI　Fumiaki SATO
Kotaro KATSUYAMA　Tadanori MIZUNO

Computer & Information Systems Laboratory,
Mitsubishi Electric Corporation.

5-1-1 Ofuna, Kamakura, Kanagawa 247, Japan

Distribution of processing and centralized resource management are researched separatedly. However, it is difficult to realize efficient systems without cooperation of the distributed processing and resource management. In this paper, we propose an environment for object-oriented distributed processing cooperating with resource management. The feature of this environment is an interface and the management facility of objects. The interface are divided to the service interface and management interface. The management interface is used for testing, monitoring, and collecting status or load of the object. In this environment, therefore, the object is a manager as well as a server. It makes easy to develop the application with resource management facilities.

1 はじめに

近年の情報関連技術の発展により，計算機システムの形態は集中処理から分散処理へ移行しつつある．分散処理では，資源を共有したり処理を幾つかの計算機に分散することで，パフォーマンスや信頼性の向上を可能としている．分散処理システムの研究では，分散ＯＳ [1][2]，分散処理言語 [3][4]，そして分散システム開発環境 [5][6] などが研究されている．また，分散システムに関する標準化作業が現在ＩＳＯ（International Organization for Standardization）において行われている [7]．また，ＯＭＧ（Object Management Group）は，オブジェクト指向システムの標準化を推進しており，分散したオブジェクト間の通信インタフェースであるＯＲＢ（Object Request Broker）の仕様を規定した [8]．

しかし，これらの環境を使って，効率的なオブジェクト指向分散アプリケーションを動作させるには，オブジェクトの状態（クラスーインスタンス関係などの静的な情報以外に，ノードの負荷などの動的な情報を含む）をどのように管理するかが問題になる．上記の分散システムにおいて，オブジェクトの分散方式についての検討はなされているが，オブジェクトの状態を管理する機構に関する検討はなされていない．

一方，ネットワーク管理などの，システムの資源をオブジェクトとしてとらえて管理するシステムの研究開発や標準化が行なわれている [9][10]．最適な負荷の分散など，資源の利用率を向上させるには，集中した資源の管理が必要である．しかし，資源が集中管理されることは，管理の要求が集中することになり，逆に効率が悪くなることがある．

これらの環境やソフトウェアの問題は，処理を分散させるモデルと資源を管理するモデルが統合されていないために生じている．ここでは，処理の分散と資源の管理を協調させた分散環境としてリソース指向分散環境を提案する．この環境の特徴は，分散環境上で動作するオブジェクトに２つのインタフェースを持たせていることである．一つは，サービスインタフェースであり，従来の分散環境で実現されているオブジェクトが備えているインタフェースである．もう一つが管理インタフェースである．管理インタフェースは，環境とオブジェクトが情報交換をするためのインタフェースであり，オブジェクトを活性化させたり，状態を試験するためのインタフェースである．この環境では，オブジェクトは，あるサービスを提供するサーバであり，またある資源（オブジェクト）を管理する管理機構となる．

リソース指向分散環境は，オブジェクト指向に基づく環境であるため，オブジェクト間の通信はメッセージパッシングにより統一的に行なわれる．また，オブジェクトのネーミング機構には，オブジェクトＩＤだけでなく，オブジェクトが属するクラス名による指定，クラスが提供するサービス名による指定が可能である．

本論文では，我々が試作したリソース指向分散環境のアーキテクチャと基本設計の概要，およびその評価について論じる．

以下，２章において分散処理における処理の分散に関する議論と資源の管理に関する議論を行なって，その問題点を提示する．３章で，ここで提案する分散処理環境本システムについて，そのコンセプトとアーキテクチャについて述べる．４章では，本システムの実現方式について，各マネージャの機能，オブジェクトの内部構成を中心に論じる．５章では，本システム環境の性能評価と考察を行なう．６章は，本論文のまとめである．

2 分散処理と資源の管理

分散処理の目的の一つは，分散環境に含まれている資源の共有である．分散した資源を使うことで，ユーザはジョブをより負荷の少ない高速なマシンで実行させることが可能となった．また，異なるマシン上で実行している間に，別の作業を行なうことでジョブの並列性が増し，効率的に処理することが可能となった．資源の共有は，高速プロセッサや特殊グラフィック表示装置，高速プリンタや専用ハードウェアなどへのアクセスを可能とし，更にデータベースなどの情報の共有も可能となった．

分散処理のもう一つの目的は，従来単一のプロセッサで行なっていた作業を複数のプロセッサで分担することにより，パフォーマンスを向上させることにある．大きなアプリケーションを，サーバ機能とクライアント機能の二つに分割し，それぞれ別のマシンで処理を行なうことにより，より高速な応答速度を得ることを目的としたクライアント・サーバ型アーキテクチャがその代表的なものである．例えば，データベースを使ったアプリケーションでは，実際に共有されているデータをアクセスするデータベース管理部 (サーバ) と，グラフィックを使ってデータの入力・表示を行なうユーザインタフェース部 (クライアント) とに分けられることが多い．分散処理のパフォーマンスを向上させるには，並行に処理できる機能単位をアプリケーションから抽出し，それを多くの分散資源に最適に配布することが必要である．

また，ネットワーク管理やファイル管理，システム管理，入出力管理などの資源管理アプリケーションは，その情報を集中的に管理し，最適な資源の管理を行なう必要がある．そのため，多くの資源管理アプリケーションは，一つの集中データを持ち一つのノード上から多くの他のノードを監視する形態で処理を行なうようになる．この場合，その管

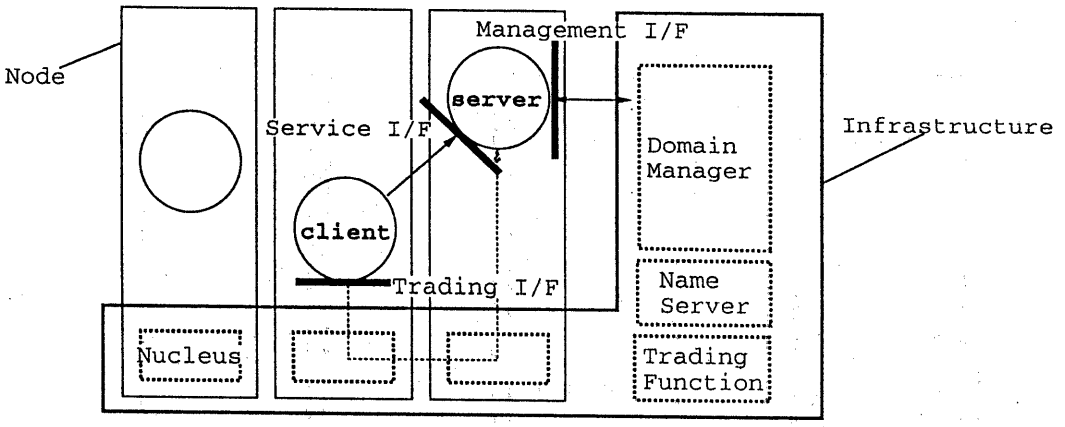

図 1: 実現モデル

理アプリケーションに集中することになる負荷は，分散の度合が大きければ大きいほど高くなり，処理効率は逆に悪くなる．

これは，即ち処理の効率を向上させるためには，分散の度合が大きい必要があり，管理の効率を向上させるためには，分散の度合が問題となる．従来，管理アプリケーションの設計では，その処理方式の効率化について考慮されることは少なかった．また，処理の分散において，その資源の管理の側面を考慮している例も少ない．即ち，管理アプリケーションは，分散されて高速に処理されるべき通常のアプリケーションの一種でもあり，更にそのシステム資源を管理するという側面を持つ．また，通常アプリケーションは，処理が分散されることによって高速に処理を進めていくことが可能だが，その処理の中には，システム資源の状況を把握することによって高速化を行なうなど，管理の機能が含まれてくる．

これは，広くアプリケーション自体がただの計算や入出力処理から資源管理を含んだものへと変化していくことを意味しており，この新しいアプリケーションに対応するためには，従来の処理の分散と管理を分離した分散環境では困難である．

ここでは，このような問題を解決するための環境として，リソース指向分散環境 [11] を提案する．本システムの目的は，処理の分散と資源の管理を含む新しいアプリケーションを効率良く開発し動作させることである．

3 リソース指向分散環境

本システムの提供する機能は，位置透過なオブジェクト管理機構と，最適なオブジェクト選定機能，階層的なオブジェクト管理機構である．また，本システム環境上で動作するオブジェクトには，他のオブジェクトからの要求に応じてサービスを提供するためのサービスインタフェースと，本環境からの要求に応じて管理情報を提供したり，本システムの環境へ自律的に管理メッセージを送ったりするための管理インタフェースを持っている．

本システムのアーキテクチャ上のコンセプトとしては，次の５つの機能を環境の主要要素として設計を行なっていった．

1. 各ノードにおけるサーバオブジェクトと環境のインタラクション処理，
2. 環境全体に渡るオブジェクトの動作状況の把握，
3. オブジェクトの抽象的な階層構造の管理，
4. クライアントオブジェクトの要求インタフェース処理，
5. 最適オブジェクトの選択処理．

それぞれ，ドメインマネージャ（Domain Manager）ネームサーバ（Name Server），ニュークリアス（Nucleus），トレーディングファンクション（Trading Function）として設計した．本システムのアーキテクチャを図 1 に示す．なお，ドメインマネージャの静的な情報を管理する部分としてタイプマネージャ（Type Manager），ニュークリアスのノードにおけるオブジェクトを操作する機能としてノードマネージャ（Node Manager）が存在する．ドメインマネージャとタイプマネージャは，論理的に環境に一つ存在している．

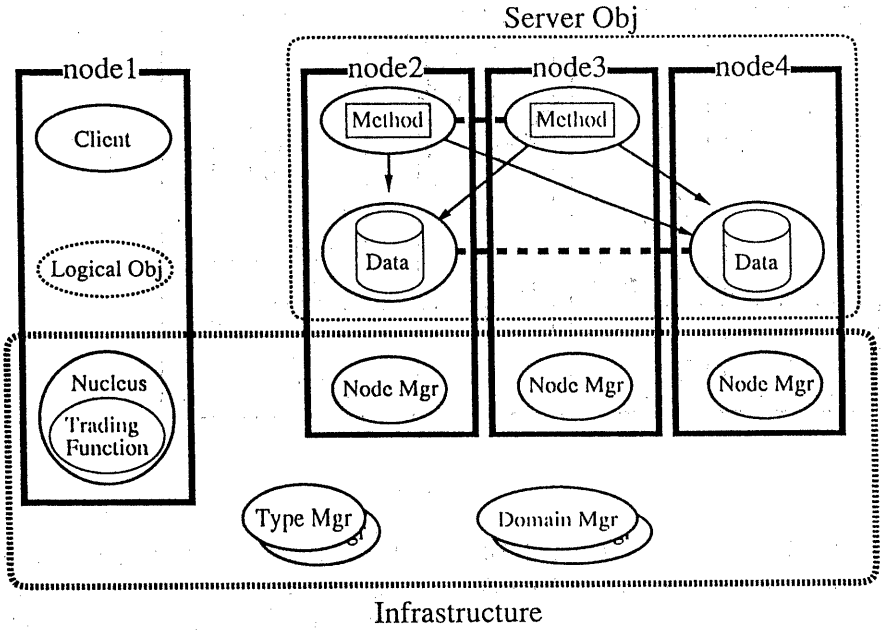

図 2: 手続きとデータが分散したオブジェクトとその管理機構

4 本システムの実現方式

本システムの実現は，UNIX[1]上にC言語を使って行なわれている．各マネージャ間は，共通のメッセージフォーマットを使って，メッセージ通信を行なっている．マネージャの複製機能により，フォールトトレラント性についても考慮している．

本システムのプログラマインタフェースは，オブジェクト指向のインタフェースとなっている．オブジェクトの生成，削除，メッセージ送信などの基本プリミティブがライブラリとして提供されているほか，オブジェクト指向C言語 [12] のインタフェースとしても提供されている．

マルチベンダの分散システム環境もいくつかの提案されているが，これらの環境は多くの機能要素が複雑に関係しており，結果として非常に負荷の重いシステムになっている [5]．本システムは，オブジェクトの管理に重点を置いており，シンプルな構造を持つことによって負荷の軽いシステムを目指している．また，クライアントのみのシステムや，サーバのみのシステムなど，自由な構成をとることができる．

[1]UNIX オペレーティングシステムは UNIX System Laboratories,Inc. が開発し，ライセンスしています．

4.1 オブジェクトの構造

本システムでのオブジェクトは，利用者（クライアント）には通常のオブジェクト指向におけるオブジェクトと同じく，データ，手続き，手続きを処理する実行者が一つにカプセル化した論理的に一つのオブジェクトとして見える．しかし，実際のオブジェクトの内部はメソッド部 (Method) とデータ部 (Data) に分離され，物理的に分散配置されている．そして，それぞれが複製を持つことも可能である [13]．このようなオブジェクトとそれを管理する環境を図2に図示する．

1. データ部

   オブジェクトのデータ部分が蓄積されている．複製を持つことにより，効率および信頼性を向上させることができる．また，データ全体が膨大な量で一つのノードでは保持しきれない場合には，複数のデータ部に分割することができる．

2. メソッド部

   クライアントがデータを利用する場合は，必ずメソッドを介してアクセスしなければならない．メソッド部もデータ部と同様複数のノードに分散している．このため，ノードの負荷やデータ部との関係などに応じて

メソッドを動的に選択することにより，クライアントに最適なサービスを提供することができる．

各オブジェクトのメソッド部およびデータ部は，複製も含め，互いの情報を保持しており，関係を自律的に管理している．

このようにメソッド部とデータ部に分離することにより，例えば，大容量の記憶領域を持つノードにデータ部を高速なＣＰＵを持つノードにメソッド部を配置したり，頻繁にアクセスを行なう利用者の近くにメソッドとデータの一部を配置するなど，利用形態に応じて最適なオブジェクト配置が可能となる．

4.2 オブジェクトの管理

複数のノードに分散配置されているオブジェクトのメソッド部およびデータ部は，インフラストラクチャによって管理されている．インフラストラクチャは，以下の構成要素からなる．

1. ニュークリアス (Nucleus：ＮＵ)

   各物理ノードに常駐し，クライアントにオブジェクトの分散透過性を提供する．クライアントからの要求に基づきトレーディング ファンクションを用いて最適なサーバ オブジェクトを選択し，そのサーバに対してメッセージを転送する．

2. トレーディング機能 (Trading Function：ＴＦ)

   ニュークリアス内の機能の一つである．クライアントからの要求とタイプ マネージャ，ドメイン マネージャからの情報をもとに最適なサーバを決定する．負荷分散アルゴリズムの実現も，この構成要素で実現されている [14].

3. タイプマネージャ (Type Manager：ＴＭ)

   ドメイン内のオブジェクトやクラスに関する名前とオブジェクトとの間の関係，および，クラス名とそのクラス情報を持つノードの関係を管理している．

   タイプ マネージャが管理している名前には，クラスおよびオブジェクトを一意に識別するプリミティブ (primitive) 名と性質を表すディスクリプティブ (descriptive) 名がある．

   タイプ マネージャは，概念的には，ドメイン内に一つの存在であるが，実際には複製を持つことにより分散されている．

4. ドメイン マネージャ (Domain Manager：ＤＭ)

   ドメイン内の各オブジェクトに関する管理情報を保持している．各ノード マネージャからの情報に基づいて，オブジェクトが動作しているノード，各ノードの負荷情報などを一元管理している．

   ただし，タイプ マネージャと同様に，ドメイン マネージャも複製を持つことができる．

5. ノードマネージャ (Node Manager：ＮＭ)

   各物理ノードに常駐し，オブジェクトの動作状況や負荷情報をドメイン マネージャに報告する．また，クラス情報も管理しており，オブジェクトの生成削除も行なう．

以上の構成要素が協調動作することにより，複数ノードに分散しているオブジェクトを利用者には，一つの論理的なオブジェクトとして透過的に見せている．

4.3 動作例

ここでは，今まで述べてきたインフラストラクチャ内の各マネージャ群とオブジェクトが実際にどのように協調動作しているかを，オブジェクトの生成，削除，メッソッド呼び出しを例として示す．

1. オブジェクトの生成
   - ＮＵは，ＴＭにクラスが存在するノードを問い合わせる．
   - ＮＵは，生成するノードのＮＭに生成要求を送る．
   - ＮＭは，fork(), exec() により，指定されたクラスのオブジェクト (メソッド) を生成する．
   - メソッドは，ＮＭに自分のポートをＮＭに伝える．
   - ＮＭは，ポート番号を登録し，ＤＭに生成を報告する．
   - ＮＵは，ＴＭにオブジェクトの生成を報告する．
2. オブジェクトの削除
   - ＮＵは，ＤＭに削除したいオブジェクトの動作ノードを問い合わせる．
   - ＮＵは，ＮＭに対して，オブジェクト (メソッド) の削除要求を送る．
   - ＮＭは，ＤＭにメソッドを終了させたことを報告する．
   - ＤＭは，オブジェクトが動作していた全てのノードからメソッドの終了報告が来たら (そのオブジェクトが動作しているノードがなくなったら)，ＴＭにオブジェクトの削除を要求する．

3. メソッド呼出し

- NUは，クラス名，クラスタイプ名，オブジェクトタイプ名などを指定して，TMに対してオブジェクトＩＤを問い合わせる．
- NUは，DMにオブジェクトの負荷情報を問い合わせる．
- NUは，最適なサーバオブジェクトを選択し，そこに要求を送信する．
- メソッドは，要求を受けとり処理を開始するときに，自分がアクティブになったことをNMに伝える．
- メソッドは，処理を終了してアイドルになったときに，自分がアイドルになったことをNMに伝える．
- NMは，ノード内でのアクティブなメソッド数をインクリメントし，その値が閾値を越えた場合は，負荷が「High」になったことをDMに報告する．その値が閾値を下回った場合は，負荷が「Low」になったことをDMに報告する．

この他に，クラスの登録，サーバオブジェクトの登録，グループ化，名前付け，オブジェクトの検索などの手続きが提供される．

表 1: 構成要素の配置

	ノード1				ノード2			
	Svr	NM	TM	DM	Svr	NM	TM	DM
a	○	○	○	○				
b	○	○					○	○
c	○	○	○					○
d	○	○		○			○	
e			○	○	○	○		
f					○	○	○	○
g			○		○	○		○
h				○	○	○	○	

5 評価

本システムの機能面での評価と，性能面での評価を行なうために，簡単な分散アプリケーションの記述を行なった．この分散アプリケーションは，データベースという共有資源を階層的に管理するアプリケーションであり，信頼性と速度が要求されている．

これに対して，本システムではオブジェクトの階層的な管理機構を用いて，このシステムを自然に設計して実現できる．また，複製を用いることにより，このシステムの信頼性と性能の向上を期待することができる．

5.1 測定項目

本システムの性能を評価するために，次のような測定を行なった．

a．オブジェクトの生成及び消滅に要する時間

b．オブジェクトを呼び出すために要する時間

表 2: クライアント／サーバの個数

番号	クライアント	サーバ
X	1	1
Y	2	1
Z	2	2

表 3: 各コンフィグレーション毎の測定結果

測定項目	a	b	c	d	e	f	g	h
生成・消滅	1	0.97	0.97	0.97	0.95	0.92	0.93	0.93
呼び出し	1	0.99	1	1	0.81	0.79	0.80	0.81

5.2 測定環境と構成

- CPU

  モトローラ 68030

- OS

  UNIX4.3bad

- 測定用アプリケーション開発言語

  superC[12]

- 通信インタフェース

  socket インタフェース

- 通信プロトコル

  UDP/IP(データグラム) プロトコル

測定項目 a と b における各コンポーネント (クライアント (Clt), サーバ (Svr), 及び各マネージャ (DM, TM, NM, NU)) の配置方法を表 1 に示す. また, 測定項目 c において, 測定に用いたクライアントとサーバの冗長度の組合せを表 2 に示す. 本測定では, クライアントおよびサーバ共に, その最大値を 2 とした.

5.3 測定結果

表 3 に, 測定項目 a と b の結果を示す. これは、コンフィグレーション a の値を基準とした時に、各コンフィグレーションと相対比較した結果である。

表 3 において, オブジェクトの生成・消滅時間をみると, TM と DM の配置位置を変えて測定した結果は, ほぼ同じ値となっている. すなわち, マネージャの位置が, 生成／消滅の処理時間にほとんど影響を与えない (a).

クライアントからのサーバ呼び出し時間では, クライアントとサーバが別ノードに存在する方が処理速度が速い (b).

5.4 考 察

測定結果 (a) の原因は, データを伴わない測定のため, 通信機構自体が, それほどローカルとリモートの通信に差がなかったためと考えられる. また, プロセスの生成時間が大きいため, 通信時間の影響が少なかったと考えられる.

測定結果 (b) の原因としては, クライアントとサーバの位置関係の違いにより, 2 つの最もアクティブなマネージャである NU と NM の位置が分散され, 間接的に速度が向上したものと考えられる.

以上の評価結果から, 資源の管理を一部に含むような新しい形態のアプリケーションの効率的な開発に十分有効な環境である. 性能的には通常 RPC の 3 倍程度の速度でメッセージ指向通信が実現され, オブジェクトの階層構造の管理機能, 複製管理, 負荷分散機能を考慮すれば十分評価できる性能であると考えられる.

6 おわりに

本論文では, 分散環境上の新しいアプリケーションの形態 (処理の分散と資源管理) に対応する新しい分散環境として, リソース指向分散環境を提案した. 本システムは, 処理の分散と資源の管理という異なる特徴を含む新しいアプリケーションに対応するため, サービスインタフェースと, 管理インタフェースを持たせることにより, 容易に実現できるようにした.

本システムを実現し, その有効性と性能を評価するために, 分散したデータベースを共有するアプリケーションを記述した. その結果, 階層化された本システムのオブジェクト管理機構を使って自然に設計することが可能であったことと, 性能面においても, 十分実用的であることが明らかとなった.

今後の課題としては, 広域のオブジェクトを管理するために, 標準的な情報管理機構であるディレクトリシステムとの連係などを考える必要がある. また, 負荷分散アルゴリズムや負荷情報の抽出機構の改良を行なう必要がある.

参考文献

[1]B.Walker, G.Popek, R.English, C.Kline and G.Thiel:"The LOCUS Distributed Operating System," Proceedings of ACM SIGOPS Conference, pp.49-70(1983).

[2]D.R.Cheriton:"The V Distributed System,"

Communications of the ACM, Vol.31, No.3, pp.314-333(1988).
[3]Y.Yokote and M.Tokoro:" Experience and Evolution of Concurrent Smalltalk," ACM OOPSLA'87 Proceedings (1987).
[4]F.Sato, I.Imai, K.Katsuyama and T.Mizuno:"Development of the Concurrent Object-Oriented Language superC2," Proceedings of IEEE 11th IPCCC'92,pp.547-554(1992).
[5]OSF:Distributed Computing Environment - An Overview, Open Software Foundation (1992).
[6]T.Yoshinaga and T.Baba:"A Parallel Object-Oriented Language A-NETL and Its Programming Environment," Proceedings of IEEE COMPSAC'91, pp.459-464(1991).
[7]ISO:Basic Reference Model of Open Distributed Processing(part 1-5), ISO/IEC JTC1/SC21/WG7(1991).
[8]OMG: Common Object Request Broker: Architecture and Specification, Revision 1.1(1991).
[9]Y.C.Shim and C.V.Ramamoorthy:"Management of Distributed Systems," Proceedings of IEEE 9th IPCCC'90, pp.689-696(1990).
[10]ISO: Information Processing Systems - OSI - System Management, ISO10165(1991).
[11] 中川路，谷林，水野：" 処理分散と資源管理を協調させた分散処理システム" ，情処学会研究報告 91-DPS-52-17(1991).
[12] 勝山，佐藤，中川路，水野：通信ソフトウェア向けオブジェクト指向言語 superC，情報処理学会論文誌，Vol.30, No.2, pp.234-241 (1989).
[13] 谷林，佐藤，中川路，水野：" 分散処理環境におけるオブジェクト実現方式" ，情処学会第４４回全国大会 (1992).
[14] 今井，佐藤，中川路，水野：" 同一機種分散システムにおける負荷分散方式" ，情処学会第４４回全国大会 (1992).