リソース指向分散環境 RODS の提案と実現

曽我正和[†] 谷林陽一[†] 長田純[†]
今井功[†] 佐藤文明[†] 中川路哲男[†]
水野忠則[†]

分散環境における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．ここでは，処理の分散と資源の管理を協調させたオブジェクト指向分散環境として“リソース指向分散環境 (RODS)”を提案する．RODS の特徴は，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，RODS 内のオブジェクト管理機構からオブジェクトを制御したり，オブジェクト管理機構に対してオブジェクトから情報を通知するためのインタフェースである．この二つのインタフェースを持たせることによって，オブジェクトはあるサービスを提供するサーバであり，かつある資源を管理する管理機構となる．この結果，資源の管理を含むアプリケーションの開発を容易に行うことができる．また，RODS のオブジェクト管理機構は，オブジェクトに関する管理情報に基づいて，クライアントからの要求メッセージを最適なサーバオブジェクトに配送する機能を持っている．この結果，効率の良い分散処理が可能となる．RODS の試作を行い，その機能および性能的な評価を行った結果，処理が分散しており資源の管理が重要なアプリケーションにおいて十分実用的な結果が得られた．

An Implementation of Resource Oriented Distributed Euvironment RODS

Masakazu Soga,[†] Youichi Tanibayashi,[†] Jun Osada,[†] Isao Imai,[†]
Fumiaki Sato,[†] Tetsuo Nakakawaji[†] and Tadanori Mizuno[†]

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. It is called RODS (Resource Oriented Distributed System). 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. On the other hand, RODS has object aggregation mechanism to manage server objects. It makes easy to identify the object providing appropriate service. We develope a prototype system of RODS and evaluate it. From the result of the evaluation, the load balancing mechanism in RODS is effective and communication cost is reasonable.

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，性能や信頼性の向上が可能となる．分散処理システムの研究としては，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)]などがある．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，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

† 三菱電機(株)情報電子研究所
Computer and Information Systems Laboratory, Mitsubishi Electric Corporation

ス-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．

従来の分散環境の問題は，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 (RODS) を提案する．RODS は，オブジェクト指向に基づく分散環境であり，オブジェクト管理機構を含んでいる．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われる．RODS の特徴は，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，オブジェクト管理機構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．RODS 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，ある資源を管理する管理機構ともなる．また，RODS のオブジェクト管理機構は，オブジェクトに関する管理情報に基づいて，クライアントからの要求メッセージを最適なサーバオブジェクトに配送する機能を持っている．

本論文では，RODS のアーキテクチャと基本設計の概要，および分散制御システムへの適用結果と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い，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 RODS について，その概念とアーキテクチャについて述べる．4章では，RODS 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，RODS の性能評価と考察を行う．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．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する．

分散処理のもう一つの目的は，処理を複数のプロセッサで分担することにより，性能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，それぞれ別の計算機で処理を行うことにより，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．

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うようになる．この場合，その管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，すなわ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．すなわち，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あり，さら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 RODS (Resource-Oriented Distributed System) を提案する．RODS 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

3.1 RODS の計算モデル[11]

RODS は，位置透過なオブジェクト管理機構，最適なオブジェクト選定機能，オブジェクト間でのメッセージ配送機構を含むインフラストラクチャと，インフラストラクチャ上で動作するオブジェクト自身の管理および動作機構から構成されている．また，RODS 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，インフラストラクチャからの要求に応じて管理情報を提供したり，インフラストラクチャ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（**図1**）．

RODS では，オブジェクトへの要求は通常の計算に関する要求のほかに，オブジェクトの管理要求が含まれる．例えば，オブジェクトの生成，オブジェクトの削除，オブジェクトの属性変更などが要求として送られる．これに対して RODS の環境を利用することにより，オブジェクトの位置を意識することなく，これらの管理業務を可能とすることができる．

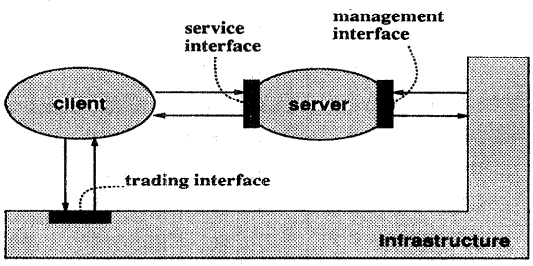


図 1 計算モデル
Fig. 1 Computational model.

RODSが提供する管理機能としては，(1)オブジェクトの生成，(2)オブジェクトの削除，(3)オブジェクトのタイプ設定，(4)オブジェクトのタイプ変更，(5)オブジェクトの複製の数の設定，(6)オブジェクトの生成可能ノードの管理，(7)オブジェクトの動作状況の管理，(8)オブジェクトが動作するノードの負荷状況の管理，があるが，これらを組み合わせることでオブジェクトの管理を行い，管理機能が含まれるアプリケーションの構築を容易にすることができる．

3.2 RODS の実現モデル

RODS のアーキテクチャ上のコンセプトとしては，次の五つの機能を環境の主要要素として設計を行っていった．

(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，ノードマネージャ (Node Manager)，ドメインマネージャ (Domain Manager)，タイプマネージャ (Type Manager)，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．RODS のアーキテクチャを**図2**に示す．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．

RODS には，複製を管理する機能があり，複製が存在することはユーザには隠蔽される（複製透過），

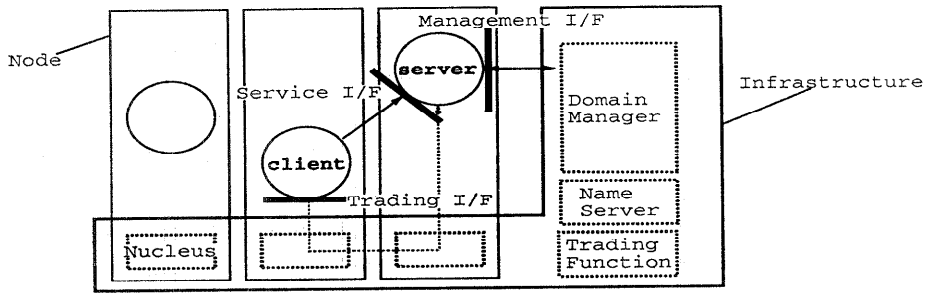


図 2 実現モデル
Fig. 2 Engineering model.

幾つかの複製が存在する場合，そのオブジェクトへのメッセージはトレーディングファンクションによって最適なオブジェクトが選択されて送信されることになる．また，オブジェクトの存在状況は，ドメインマネージャによって監視されており，複製に異常があれば，クライアントからのメッセージは異常なオブジェクトを避けて正常なオブジェクトに送信されることになる．

オブジェクトをクラス名とは別のカテゴリ（性質など）で分類し，ディスクリプティブ名で管理することによりオブジェクトの抽象度を上げ，アプリケーションをより自然に記述することが可能となった．

最適なオブジェクトの選択機能により，負荷の低いノードでの処理を可能とすることで負荷の分散が実現できる．また，このことから，RODS では従来の分散 OS で問題となる管理機能の集中化を回避することが可能となり，資源の階層的な管理機構を自然に実現することができ，負荷を分散することで管理機構の応答性を高めることができる．

4. RODS の実現方式

RODS の実現は，UNIX 上にC言語を使って行われている．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．

RODS 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．

また，RODS は，オブジェクトの管理に重点を置いており，単純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．

4.1 オブジェクトの構造

RODS でのオブジェクトは，利用者（クライアント）には，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存在として見える．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 3** に図示する．

（1） 論理オブジェクト

論理オブジェクトは，オブジェクトの実体ではなく，クライアントに対して，物理的な位置や内部が分散されていることを隠し，論理的に一つのオブジェクトとして見せるものである．各論理オブジェクトはオブジェクト ID に対応している．

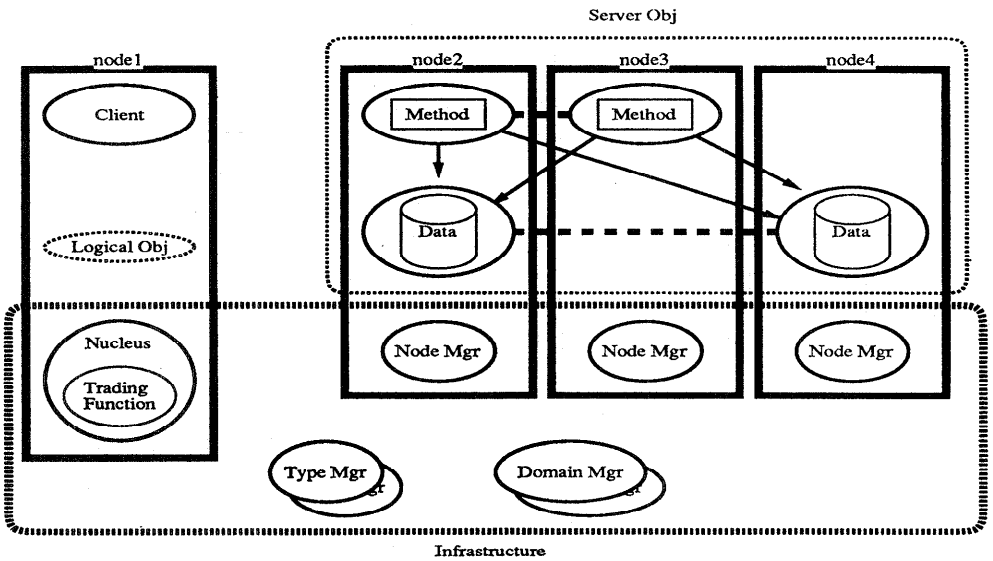


図 3 手続きとデータが分散したオブジェクトとその管理機構
Fig. 3 Distribution and management of data and procedure of objects.

（2） 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．

（3） 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，それを管理する機能を持っている．

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

RODS におけるオブジェクトは，インフラストラクチャにより管理されている．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） トレーダ（Trader : Tr）

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得られるオブジェクトやノードに関する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]．ただし，本実装では，トレーダは独立したプロセスとしてではなく，ニュークリアス内の一つの機能（トレーディングファンクション：TF）として実現されている．

（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）名の他に，性質などを表すディスクリプティブ（descriptive）名があり，より抽象度を上げたオブジェクトの管理が可能となっている．

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，物理的には分散していてもよい．

（4） ドメインマネージャ（Domain Manager : DM）

ドメ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，物理的には分散していてもよい．

（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，ノードごとの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ク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，以下，オブジェクトの生成，削除，メソッド呼び出しを例として示す．

（1） オブジェクトの生成（**図 4**）

- NU は，TM にクラスが存在するノードを問い合わせる．これに対して TM は，クラスが存在するすべてのノードとそのクラスに設定されている冗長度（複製の数）を返す．

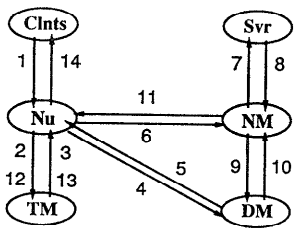


図 4 オブジェクトの生成
Fig. 4 Creating an object.

- NU は，生成するノードの NM に生成要求を送る．複製を生成する場合は，生成するすべてのノードに対して生成要求を送る．
- NM は，指定されたクラスのオブジェクトを fork()，exec() により Unix のプロセスとして生成する．
  このときオブジェクトのメソッドおよびデータは，それぞれプロセスとして起動される．
- メソッドは，Unix におけるトランスポート端点であるポートを確保し，このポート番号を 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すべ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） メソッド呼び出し

- 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，処理が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．ここで，閾値は各ノードの管理者がそのノードの性能などを考慮して決定した値で，初期設定ファイルに指定されている．

このほかに，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．

5. 評　　価

RODS の機能面での評価と，性能面での評価を行うために，簡単な分散アプリケーションの記述を行った．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，RODS 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 分散アプリケーションの概要

本節では，工場での生産システムを分散制御するアプリケーションについて述べる．本アプリケーションは，RODS が持つ，1)拡張性，2)位置透過性，3)負荷分散，4)フォールトトレラントといった特徴を検証するものである．以下に，その概要を述べる．

本アプリケーションは，データ管理部，機械管理部，ユーザ管理部から構成される．

本アプリケーションは，複数のノード上に存在する加工機械オブジェクトと搬送オブジェクトを，一つのノードから制御するためのアプリケーションである．制御オブジェクトのほかに，DB（データベース）オブジェクト，ディスプレイオブジェクト，コマンドインタプリタオブジェクトが存在する構成とする．制御アプリケーションの動作は，プログラムのダウンロード，スケジュールのダウンロード，加工結果のアップロード，機械の起動，停止，モニタリング等である．

5.2 分散アプリケーションのオブジェクト構成

分散アプリケーションのオブジェクト構成は，**図5** のようになる．

各構成要素の動作内容を以下に示す．

- メイン・オブジェクト

コマンドインタプリタ，加工機械，搬送機械，ディスプレイ，プログラム DB，コマンド実行の各オブジェクトの起動および初期化を行う．

- コマンドインタプリタ・オブジェクト

ユーザから，プログラムのダウンロード，起動，停

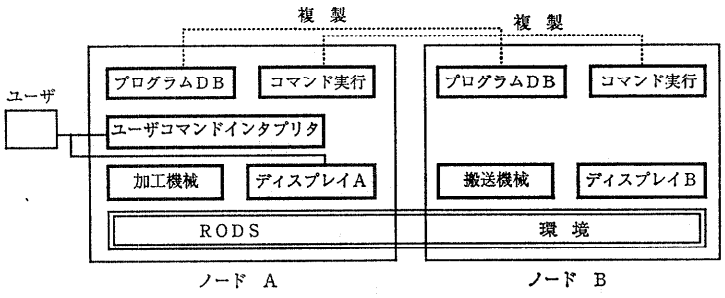


図 5 分散アプリケーションの構成図
Fig. 5 Structure for the distributed application.

止のコマンドを受け付け，コマンド実行オブジェクトへの依頼を行う．

●コマンド実行オブジェクト

コマンドインタプリタからの要求に応じて，プログラム DB，加工機械，ディスプレイとメッセージをやり取りして加工作業を実施する．

●プログラム DB オブジェクト

コマンド実行オブジェクトからの，プログラム検索メッセージにより動作して，指定されたプログラムデータを返す．また，この DB はオブジェクト指向に基づいて実現されており，プログラムだけでなく部品情報，治工具，CAD 情報などの複雑な構成を持つ情報も管理できる．

●加工機械オブジェクト

コマンド実行オブジェクトからの，プログラムロードメッセージ，起動メッセージ，停止メッセージにより動作して，その応答として，現在の状態をモニタ情報として返す．

●搬送機械オブジェクト

加工機械オブジェクトと同様にコマンド実行オブジェクトからの，起動メッセージ，停止メッセージにより動作して，その応答として，現在の状態をモニタ情報として返す．

●ディスプレイオブジェクト

コマンド実行オブジェクトからの，モニタ結果表示メッセージにより起動して，モニタ結果を表示する．

ここで，本分散アプリケーションの全体的な動作概要を説明すると，まずコマンドインタプリタ・オブジェクトが，ユーザ・コマンドを受けつける．そしてユーザコマンドは，コマンド実行オブジェクトへ送られる．コマンド実行オブジェクトは，プログラム DB 加工機械，搬送機械オブジェクトとメッセージ交信を行って，機械加工を実行すると同時に，ディスプレイ・オブジェクトにその過程を表示する．

5.3 分散アプリケーションによる RODS の検証

前節で述べたアプリケーションによって，以下のことを検証することができる．

コマンドインタプリタ・オブジェクトからコマンド実行オブジェクトへのメッセージ，あるいはコマンド実行オブジェクトから，各加工機械へのメッセージを位置透過に送ることができる．

ノードAの負荷が高くなってきた場合に，コマンドインタプリタからのメッセージが自動的にノードBのコマンド実行オブジェクトへ送られるようになり，負荷を分散できる．また，ノードAのオブジェクトに障害が起こった場合，ノードBのオブジェクトにメッセージを送り直すことによって，ノードBで処理を代替し，耐故障性を高めることができる．

搬送機械も含めて，すべてノードAで動作する環境から，ソースコードを変更することなく，搬送機械をノードBで動作させたり，コマンド実行やプログラム DB の複製を動作させたりする環境へ容易に拡張できる．

5.4 性能評価

（1） 測定項目

これに対して，RODS の性能を評価するために，次のような測定を行った．

ⅰ．オブジェクトの生成および消滅に要する時間
ⅱ．オブジェクトを呼び出すために要する時間
ⅲ．複製の並行処理による応答速度

（2） 測定環境と構成

●CPU
モトローラ 68030

●OS
UNIX 4.3 BSD

●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ⅰとⅱの測定においては，各コンポーネン

表 1 構成要素の配置
Table 1 Position of components.

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

表 2 クライアント/サーバの個数
Table 2 Number of servers and clients.

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ごとの測定結果
Table 3 Performance of each system configuration.

測定項目	a	b	c	d	e	f	g	h
生成・消滅	524	507	506	508	498	487	490	488
呼び出し	87	87	86	87	71	69	70	70

(単位：ミリ秒)

ト（クライアント (Clt)，サーバ (Svr)，および各マネージャ (DM, TM, NM, NU)）の配置方法が性能に与える影響を調べるために，二つのノードで考えられるすべての配置方法に関して測定を行った．**表1**で a, …, h は，配置方法を識別する記号である．また，測定項目 iii 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み合わせを**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．

（3） 測 定 結 果

表3に，測定項目 i と ii の結果を示す．この表において，a, …, h は，表1で示した各配置方法に対応している．また**図6**に測定項目 iii の結果（単位：ミリ秒）を示す．横軸はサーバ内での処理量（単位：整数加算回数×10^3）を，縦軸はクライアントから呼び出した全体の応答時間（単位：ミリ秒）を各々表している．

表3において，オブジェクトの生成・消滅時間をみると，a～d と e～h はそれぞれ近い値となっている．すなわち，クライアントとサーバの位置を固定し，TM と DM の配置位置を変えて，生成/消滅の処理時間にほとんど影響を与えないことを意味している (A)．

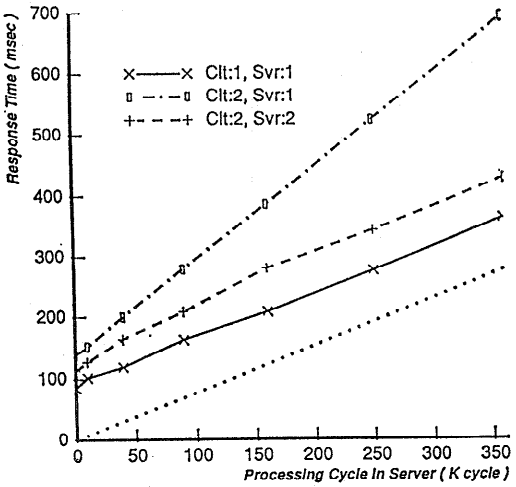


図 6 クライアントとサーバの処理時間
Fig. 6 Performance of clients and servers.

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同一ノードに存在する場合より（例えば，f の方が b より）応答時間が短い (B)．

図6から分かるように，ある程度の通信のオーバヘッドが伴うが，サーバでの処理時間の増加に応じて，クライアントでの応答時間が，直線的に増加している．クライアントが二つになると，その応答時間はほぼ2倍になり，サーバも二つにすると，応答速度は1：1 の場合の結果に近づいている (C)．

5.5 考 察

測定結果(A)の原因は，データを伴わない測定のため，通信機構自体の，ローカルとリモートの処理時間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，二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．

測定結果(C)のクライアントとサーバの数が，2：1 から 2：2 になった際に，応答時間が約半分になったのは，RODS の動的な負荷分散機構が有効に働いたためと考えられる．ただし，クライアントとサーバが

2：2 の場合，1：1 の場合よりも多少の応答時間の増加が見られる．これは，RODS の持つ動的な負荷分散機構において，負荷情報通知の遅れによって，負荷の分散が適切に行われない場合があり，平均応答時間の増加に継っていると考えられる．

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倍程度の速度でオブジェクト間のメッセージ通信が実現され，オブジェクトの管理機構な負荷分散機能が重要なアプリケーションにおいては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（RODS）を提案した．RODS 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．

RODS 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 RODS 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，標準的なディレクトリシステムとの連携による広域分散環境への対応，並行性制御，複製管理の実現，負荷分散機構の改良がある．

謝辞 本システムの開発に御協力いただいた(株)三菱総合研究所 石川雅基氏に感謝いたします．また，貴重な御助言をいただいた三菱電機(株)名古屋製作所 中野宣政氏，山下昭裕氏ほか関係各位，ならびに同情報電子研究所の関係各位に感謝いたします．

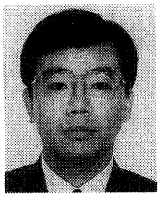
参考文献

1) Walker, B., Popek, G., English, R., Kline, C. and Thiel, G.: The LOCUS Distributed Operating System, *Proceedings of ACM SIGOPS Conference*, pp. 49-70 (1983).
2) Cheriton, D. R.: The V Distributed System, *Communications of the ACM*, Vol. 31, No. 3, pp. 314-333 (1988).
3) Yokote, Y. and Tokoro, M.: Experience and Evolution of Concurrent Smalltalk, *ACM OOPSLA '87 Proceedings* (1987).
4) Sato, F., Imai, I., Katsuyama, K. and Mizuno, T.: 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) Yoshinaga, T. and Baba, T.: 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 JTC 1/SC 21/WG 7 (1991).
8) OMG: Common Object Request Broker: Acchitecture and Specification, Revision 1.1 (1991).
9) Shim, Y. C. and Ramamoorthy, C. V.: Management of Distributed Systems, *Proceedings of IEEE 9th IACCC '90*, pp. 689-696 (1990).
10) ISO: Information Processing Systems—OSI—System Management, ISO 10165 (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，情報処理学会研究報告 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，第 44 回情報処理学会全国大会論文集，pp. 1-229 (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，第 44 回情報処理学会全国大会論文集，pp. 1-235 (1992).

（平成4月11月5日受付）
（平成5年 4 月8日採録）

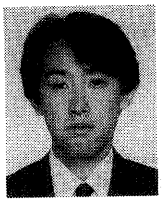
曽我 正和

昭和 11 年2月生．昭和 33 年京都大学電子工学科卒業．昭和 35 年同修士課程卒業．同年三菱電機(株)入社．国鉄郡山操車場自動化システム，リアルタイムコンピュータ MELCOM 350，汎用コンピュータ MELCOM-COSMO，等の CPU およびハードウェアシステムの開発主任を務めた．その後も主に各種開発プロジェクトに従事．平成2年12月より情報電子研究所所長．電子情報通信学会会員．



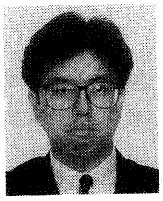
谷林 陽一（正会員）

1964 年生．1988 年慶応義塾大学理工学部電気工学科卒業．1990 年同大学院修士課程修了．同年三菱電機株式会社入社．現在同社情報電子研究所システム技術開発部に勤務．情報通信システムおよび分散処理システムに関する研究・開発に従事．



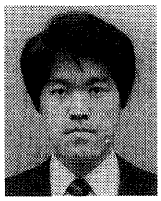
長田 純（正会員）

昭和 42 年生．平成 3 年法政大学電機工学科電気電子専攻卒業．同年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部に勤務．情報通信システムおよび分散処理システムに関する研究・開発に従事．



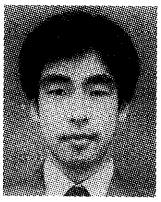
今井 功（正会員）

昭和 40 年生．昭和 63 年東海大学工学部電子工学科卒業．平成 2 年同大学院工学研究科電気工学専攻修士課程修了．同年三菱電機(株)に入社．現在，同社情報電子研究所に勤務．情報通信ソフトウェアおよび並列分散処理システムの研究・開発に従事．



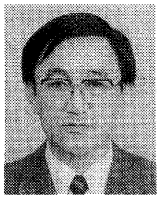
佐藤 文明（正会員）

昭和 37 年生．昭和 59 年岩手大学工学部電気工学科卒業．昭和 61 年東北大学大学院修士課程修了．同年三菱電機(株)入社．現在，同社情報電子研究所にて通信ソフトウェア開発環境，形式記述技法，並列分散処理システムの研究開発に従事．工学博士．電子情報通信学会，IEEE 各会員．



中川路哲男（正会員）

昭和 33 年生．昭和 56 年 3 月東京大学工学部電気工学科卒業．昭和 58年 3 月同大学大学院工学研究科電気工学専攻修士課程修了．同年三菱電機(株)入社．現在同社情報電子研究所システム技術開発部に勤務．工学博士．OSI 通信ソフトウェアを中心とする分散処理システムの構築およびソフトウェア工学に関する研究・開発に従事．昭和 63 年 9 月情報処理学会全国大会において学術奨励賞受賞．電子情報通信学会各会員．



水野 忠則（正会員）

昭和 20 年生．昭和 43 年名古屋工業大学経営工学科卒業．同年三菱電機(株)入社．平成 5 年静岡大学工学部情報知識工学科教授．工学博士．情報通信システムおよび分散処理システムに関する研究に従事．著書としては，「マイコンローカルネットワーク」（産報出版），「MAP/TOP と生産システム」（オーム社），「分散システム入門」（共著，近代科学社），「分散システム—コンセプトとデザイン」（共訳，電気書院）などがある．電子情報通信学会，オフィスオートメーション学会，日本経営工学会，IEEE 各会員．