大域計算アーキテクチャ——広域環境での並行計算とマルチメディア処理の統合的実現

前川博俊† 斉藤隆之† 千葉哲央†

ネットワーク技術や半導体技術の発達に呼応して，マルチメディア情報処理や分散高性能計算など多様な分野でネットワーク計算の重要性が増している．それらのシステムをより高度で効率の良い形で実現するには，広域のネットワークを1つの統合的な計算環境としてとらえ，そのうえで大域的な計算を行えるような枠組みの提供が求められる．我々は，そのような大域計算を広域のネットワーク環境上で実現するための計算方式を提案する．その方式は，多様な通信メディアや実行環境の違いに適応可能な並行計算機構の実現を可能とする．アプリケーションシステムは広域の環境上にスケーラブルに配置し，その計算空間を柔軟に形成し管理することができる．また，マルチメディア処理に対応するため，連続メディアデータを含むデータ構造体を大域的に扱えるようにした．システムのオープン性を実現するため，ネットワーク上の未知のリソースを動的に探索して利用するための機能を導入した．現在，この方式に基づいた大域計算のフレームワークシステムが，Solaris 2.4 と WindowsNT 4.0 の環境上で稼働中である．フレームワークシステム上に実験アプリケーションを構築し，提案する計算方式の有効性を確かめた．

Global Computation Architecture: Synthesized Wide-area Concurrent Computation and Multimedia Processing

HIROTOSHI MAEGAWA,† TAKAYUKI SAITO† and TETSUHIRO CHIBA†

Network computing is increasing its needs in diverse areas, such as multimedia information processing and distributed high-performance computing, as technologies for networks and semiconductors become improved. In order to develop those systems more effectively and efficiently, a computation framework is needed that can manage wide-area networks as synthesized environment where a global computation can be developed. We propose a computation scheme capable of developing such a global computation over wide-area network environment. The scheme makes it possible to develop a concurrent computation feature adaptive to a variety of communication media and execution environment. The scheme can flexibly create and manage a computation space by scalably distributing an application system over wide-area environment. The scheme manipulates global data structures including continuous-media data with an aim for multimedia processing. The scheme also adopts a feature of searching and using unknown resources on a network in an attempt for achieving an open system. We have developed a global computation framework system based on the scheme on Solaris 2.4 and WindowsNT 4.0. We have confirmed the efficiency and effectiveness of the global computation scheme with an experimental application.

1. はじめに

本論文では，広域のネットワーク環境における大域的な計算を実現するための計算方式を提案する．また，その方式の検証システムとして開発している大域計算フレームワークについて報告する．その方式は，大域の計算を特徴づける本質的要件から構成されており，多様な計算環境やネットワーク環境に適応して広域の計算空間を形成し並行処理を実行することが可能である[1)~3)]．

通信技術や蓄積技術，半導体技術の発達を背景として，ネットワーク上の計算は，分散高性能計算や分散マルチメディア処理など多様な分野でその重要性を増している．WWW[4)]といった情報アクセス手段の普及も，広域で高度な情報交換システムが求められていることを示唆している．ネットワークとそこでの計算リソースを従来以上に有効に活用することにより期待されるのは，高性能計算のリソースを広域で活用したシステムや，素材や表現方法の異なる情報をネット

† 株式会社ディジタル・ビジョン・ラボラトリーズ
Digital Vision Laboratories Corporation

ワークワイドな多様な局面で柔軟に利用できるようなシステムの普及である．そのようなシステムをより容易に構築するためには，広域の環境での分散された処理を統合的な1つの計算として実現するための枠組みや，またさらに，連続メディアデータを含むデータ構造体をネットワークワイドに扱える機能の実現が求められる．

しかし，分散処理やネットワーク計算に関する従来の方式では，大域の計算を統合的に扱える機能や，大域空間とそのリソース管理の機能，フレキシブルでオープンなネットワーク接続の機能，連続メディアデータの分散処理機能への対応が十分ではない．CORBA[5]のような均一空間上でオブジェクトの所在を特定するパラダイムを持ったプラットフォームは，広域のネットワーク計算には不向きであり，分散処理の特質を隠蔽していることによる計算効率の低下をともなう．また，オブジェクト機能の呼び出しインタフェースを規定しなければならず，プログラミング上の柔軟性に欠ける．PVM[6]，MPI[7]といった分散システム構築機能は，より密に結合されたシステム用にデザインされたものである．分散オブジェクト指向言語や分散実行環境がかねてより開発されている[8)~13)]が，ネットワーク計算の種々の機構はさらにその上に構築する必要がある．クライアント・サーバ方式[14),15)]は，動的に機能や接続形態を変更したり複数のモジュールが相互対話的に動作する用途よりも，機能や仕様が確定した計算に向いている．URLなどリソース記述子の指定に基づいてネットワーク上のデータやプログラムをダウンロードし計算を進める方式[4),16)]は，実現される対話性やシステムの持つ機能の発展性に限界がある．文献17)，18)は広域の分散処理を実現するが，特定の言語に限定されており，また，計算実行のための空間が管理されない．一方ネットワーク構成の観点からは，ネットワークの物理的構成からネットワーク上での計算の概念形成に至る広い範囲で，その形態と運用に柔軟性を持たせる試みがなされている[19),20)]が，これらは計算のためのフレームワークと統合して活用したい．広域のデータは，一貫性を保った利用が実現されている[21),22)]が，構造の動的変更や処理の非精密性を許容した実現を考察したい．また，連続メディアストリームを扱ったシステムでは，リアルタイム性や処理の効率化を追求する試みはなされている[23)~25)]が，それらをさらにネットワークワイドな局面で大域的な構造体の一部として使用したい．我々が想定する高度で大域的なネットワーク計算を実現するには，このような機能やシステムの特長の発展とともに，大域的なアプリケーション空間の管理や，そこでの分散・協調計算のための機能，マルチメディアデータ構造体を大域的に処理できる機能の提供が求められる．

我々の研究目的は，広域ネットワーク環境を対象とした大域計算の本質的要件を明らかにし，その要件に基づいた大域計算機能を実現することである．また一方での研究の動機は，来たるべき高度なネットワーク社会におけるマルチメディア情報サービスシステムの実現を検証するためのネットワーク計算ミドルウェアを提供することである．我々は，映像や音声を含むマルチメディア情報を商品としてオープンなネットワーク上で自由に取引きできる機構の実現を目的として，システムミドルウェアの開発を行っている[26),27)]．本研究はその一環として，マルチメディア処理機能を備えネットワーク空間上でのシステム構成の柔軟性に対処した並行計算系を実現するものである．

本稿の以下においては，我々が提案する大域計算のパラダイムと，そこでの並行計算と空間管理，マルチメディア処理の方式について述べ，その方式の動作環境として我々が開発している大域計算フレームワークを通して機能実現方法に言及する．さらに実験アプリケーションによって，計算方式と実現機能の評価を行う．

2. 大域計算の方式

我々が想定する大域計算について述べ，それを実現するアーキテクチャとして大域計算の方式を提案し，その計算パラダイムについて論じる．

2.1 大域計算の基本パラダイム

大域計算（Global Computation）とは，広域のネットワーク上の計算であって，そこでの分散リソースを論理的に一体のものとしてとらえ，そしてそのうえで実行される計算であるとする．ここで，広域のネットワークとは，インターネットや，CATV網，衛星通信など多様な通信メディアが混在しているものであり，分散リソースとは，計算機ノードなどのハードウェアに加えて，データベースや高性能計算サーバ，WWWページ等の情報ソースなど，計算ポテンシャルを提供するものである．従来の広域（wide area）計算は，統合環境上での一体化された計算ではないと解釈し，大域計算とは区別する．**図1**に，我々が想定する大域計算の概念構成を示す．

ネットワーク計算を，単に分散モジュールの相互連携という形式によるものではなく，統合的環境の上で実現するためには，次の要件が求められる．

- 分散リソースとその非同期性に適合した計算のパ

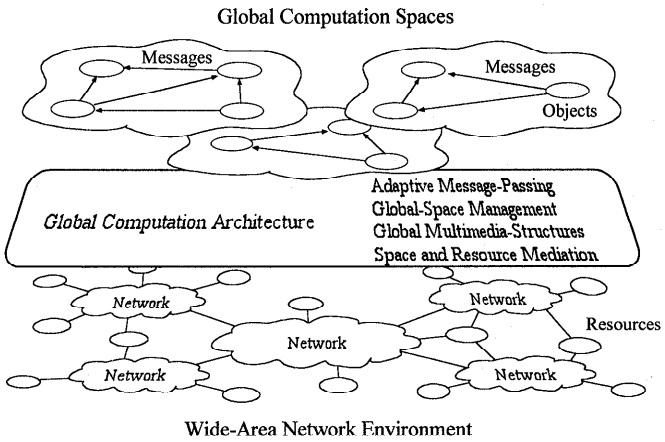


図1 大域計算と統合的広域ネットワーク環境
Fig. 1 Global computation over synthesized wide-area network environment.

ラダイムを持つこと
- 大域的なアプリケーション計算空間を柔軟に形成して管理できること
- 大域的な参照と名前の機能を持つこと
- 通信メディアや計算リソースの違いを吸収した大域環境を扱えること
- 連続メディアデータを含めたデータ構造体を大域的に扱えること
- 既存のリソースと処理系を活用した形で広域の計算を実行できること
- 計算空間やリソースの探索と利用の機能によってオープン性を実現すること
- 分散処理の見地からの信頼性とセキュリティに対処できること☆

これらの機能要件を満たしたうえで大域計算の機構を実現するため，我々は次の機能を備えた計算の方式を導入した．

- 大域並行計算
  基本となる計算の方式は，分散モジュールが非同期のメッセージ交換によって並行計算を実行し，大域的にはそのメッセージの連鎖によって計算が進行するものである．広域の分散・協調計算への適合性を考慮して，同期性の少ない計算の枠組みを提供する☆☆．この大域的並行計算は次の2つの機能により実現する．
- 適応型メッセージパッシング
  分散モジュール間のメッセージは，種々のプログラミング言語や，OS，ハードウェアといった実行環境の違いに適応した形で送信する．メッセージ上のデータは共通の論理形式で扱う．メッセージ送信先の特定は次の大域参照機能による．
- 大域計算空間管理
  アプリケーション空間を広域環境上に柔軟に形成して管理する．メッセージ送信先や空間内のリソースの参照は，大域参照によって特定する．計算実体の大域的特定により，分散モジュール間での効率の良いメッセージ送信を実現する．
  大域参照は，予約された大域リソースを表現するため，名前を付けて管理してよい．大域参照には，URL（Universal Resource Locator）など他のリソース参照も取り入れることができる．アプリケーションモジュールは，大域参照の処理によって，動的な再配置，追加，移動が可能である．
- 大域マルチメディア構造体
  大域参照を用いた分散データ構造体を，アプリケーション上で大域的に扱う．また，連続メディアデータをそのストリーム配送機構を含めてオブジェクト抽象化し，大域データ構造体の中で扱う．ストリーム配送機構は広域の環境上に分散して配置されてよい．ストリーム間同期などの機能も，このデータ構造体の中で処理する．
- 空間・リソースメディエーション
  広域の環境上でオープンな計算を実現するには，未知や所在不明，あるいは所在が更新されたリソースを探索して特定できる必要がある．このため，ネットワークメディエータという探索用のモジュールを存在させ，メディエータ間の連携によっ

☆ 信頼性とセキュリティの機能は，我々の計算方式がただちに実現するものではない．これらは高次のユーティリティ機能として位置付けている（4.4 節参照）．

☆☆ 我々は文献 28) で，非同期メッセージパッシングに基づいた分散処理の効率の良さを実証している．

てそれらのリソースの探索を行う．上記の大域参照あるいは大域データ構造体中のそれらに対しても，参照先が直接特定できない場合，このメディエーション機能を作用させることができる．

このような機能に基づいて実現される計算実行系は同時に，次の特長を備える．

- 分散モジュールの動的な配置・生成による柔軟なプログラム開発が可能である．
- 遠隔的に分散モジュールを配置できるので，広域のアプリケーションシステムの構築が容易である．
- 広域計算のための既存の通信メディアを活用できる．
- 広域環境での連続メディア処理が容易である．
- メッセージパッシングと大域参照の機構をコアとした移植性が良く既存の実行環境に適合しやすい実装が可能である．

以下の項において，提案する計算方式の詳細を述べる．

2.2 大域並行計算の方式

2.2.1 並行計算の方式

大域計算を実行する分散モジュールを，並行オブジェクトと呼ぶ．並行オブジェクトは大域アプリケーション空間上で管理する．並行オブジェクトによる次のパラダイムを持った並行計算を行う：

- 分散手続き（関数）呼び出し
- 分散オブジェクト指向計算
- 大域共有変数

並行オブジェクトは，アプリケーション空間上の計算の実体であり，リモート呼出可能な分散関数，分散オブジェクト，あるいは，大域共有変数である．並行オブジェクトは，互いのローカル記憶を共有しない．分散オブジェクトはその内部に，ローカルオブジェクトを所有できる．ローカルオブジェクトは，分散オブジェクト内で共有記憶を持つ．分散オブジェクトの生成は，そのための分散関数による．

上記の計算を行うための並行オブジェクト間の通信（分散関数や分散オブジェクトメソッドの呼び出し，大域共有変数へのアクセス）は，次の方法により行う．

- 非同期メッセージ送信（返値を持たない1方向の通信）
- 完全同期呼び出し（返値が来るまで呼出側がブロックする）
- 遅延評価型同期呼び出し（返値にアクセスしたときブロックする）

これらの通信を総称してメッセージと呼ぶ．メッセージ上では並行オブジェクト間で，後述のマルチメディアデータを含む任意のデータ構造体を送ることができる．

2.2.2 大域的計算の実行

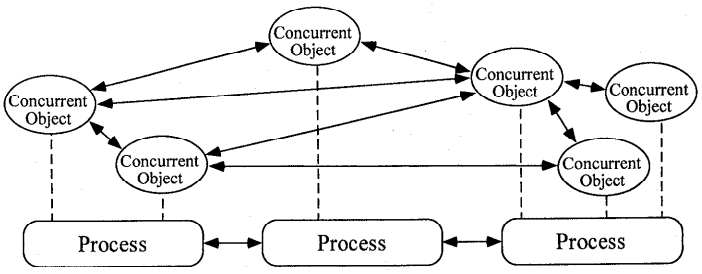
アプリケーション空間に配置する並行オブジェクトは，ネットワークノードにプロセスを生成して管理する．プロセスは，並行オブジェクトのための実行環境である．これはOSやハードウェアに対するシステムインタフェースを提供するとともに，大域空間上で共通の仮想的計算環境として機能する．プロセスは実行環境として，オブジェクト間メッセージ通信，大域オブジェクト参照，およびオブジェクト生成・配置の機能を持つ．アプリケーション空間を構成する並行オブジェクトとプロセスによる計算環境の関係と，並行オブジェクトの内部状態構成を，**図2**に示す．

メッセージの受信によって起動される並行オブジェクト上の計算は，関数やメソッドの存在するプロセスにおいて実行する．実際に稼働させるプログラミング言語によって，メソッドがオブジェクトに属するのであれば，その実行はそのオブジェクトで行う．メソッドが汎関数から派生するものであれば，実行はそのメソッドの所在で行う．計算の演算対象のスコープは，並行オブジェクトの内部である．リモートオブジェクトのオペランドを直接見ることはできない．

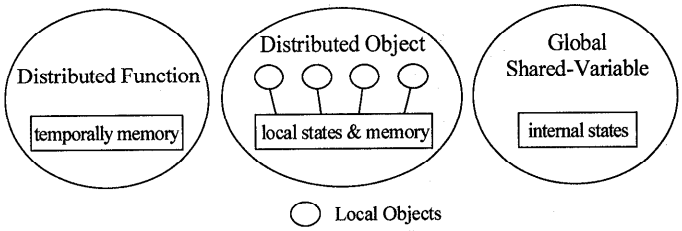
プロセスにおける受信メッセージの起動と実行は，長期と短期のスケジューリングによって行う．長期スケジューリングでは，メッセージに付帯した時間属性と優先度に基づいて，評価すべきメッセージを選択して実行可能状態とする．短期スケジューリングでは，可動状態にある複数のオブジェクトを並行して処理するための制御を行う．

メッセージの評価のうち分散オブジェクトメソッドの計算と大域共有変数アクセスは，そのオブジェクトにおいて排他的かつアトミックに実行する．同一のオブジェクト上で複数のメッセージを並行して評価したい場合や，メソッド内で遅延処理をしたい場合は，それらを非同期メッセージに分割した形で記述し処理する．

アプリケーションプログラミング上には，並行オブジェクトの基本機能と上述したメッセージ機能を組込ユーティリティとして用意する．それらの機能によって大域並行計算を記述することができる．また，メッセージ送信の際，使用するネットワークの種類を指定することが可能であり，通信するデータのメディアの種類と制御の内容に応じて異なる通信方法を選択できる．また，メッセージ通信と連続メディアの配送を中継プロキシを用いて多段的に扱うことにより，非均質



(a) Concurrent objects and processes



(b) Concurrent objects and their internal states

図2 並行計算環境
Fig. 2 Concurrent computation environment.

なネットワーク環境にも対応できる.

2.3 大域計算空間の管理

2.3.1 オブジェクトと参照の管理

並行オブジェクトと分散オブジェクトメソッドに,分散配置されたデータオブジェクトや利用可能な情報リソースを含めて,計算オブジェクトと呼ぶ.計算オブジェクトは,大域参照で管理する.大域参照は,オブジェクトの実行環境を与えるプロセスとプロセス内でのオブジェクトの識別子で表す.プロセスは,ネットワークノードとノード内のプロセスの識別子で表す.オブジェクトの識別子は,プロセスで決定する大域的に使用できる形式の識別子で,リモート識別子と呼ぶ.リモート識別子に対応しプロセス内で実際のオブジェクトを特定する実行環境依存の識別子を,ローカル識別子として区別する.オブジェクトが生成されたとき,これらの識別子を割り当て大域参照を生成する.

並行オブジェクトは,計算空間上をマイグレートすることが可能である.そのオブジェクトが持つ大域参照はそのまま保持すればよい.そのオブジェクトへの参照は,移動元のプロセスに間接参照を暫時存在させ,元の参照へのアクセスがあったときはそれを新たな参照に切り替える.間接参照は,最終的に大域ガーベジコレクションの対象となる.

計算空間は,アプリケーション空間の他にドメイン空間を形成して管理できる.ドメイン空間は,アプリケーション空間上の任意の並行オブジェクトやプロセスをまとめた空間であり,同様に大域参照で管理する.ドメインはまた,別のドメインを含んでもよい.ドメインはアプリケーション空間をまたがっていてもよく,個々のアプリケーション空間を超えた計算空間を提供することができる.

大域参照からその参照対象の情報への変換を,参照解決と呼ぶ.参照解決の方式と機能の構成を**図3**に示す.参照の解決は,オブジェクトの識別情報を特定できるまで再帰的に動作させる.大域参照には名前を付けて,そのオブジェクトを参照してよい.ただし,名前を扱うことによる応分のオーバヘッドをともなう.大域参照にはまた,URL など他のリソース特定手段も採り入れることができる.

計算オブジェクトは,大域ガーベジコレクションの対象である.大域ガーベジコレクションは,計算オブジェクトに対する大域参照の有無を判定することにより行う.大域参照と計算オブジェクトにはそれぞれ重みとカウンタをシステムタグとして付加し,重み付き参照カウント法[29]を基本とした即時型コレクションを行う.実行環境に依存したプロセス内のリソースは,ローカルコレクションあるいはアプリケーションプログラマによる処理の対象であると仮定する.大域参照の抹消は,ローカルコレクションから得る.大域的被参照のなくなったオブジェクトは,ローカルコレクションの対象とする.

2.3.2 大域アプリケーションとその空間

アプリケーション空間は,同一のネットワーク環境上に複数のものをそれぞれ個別に形成し管理するこ

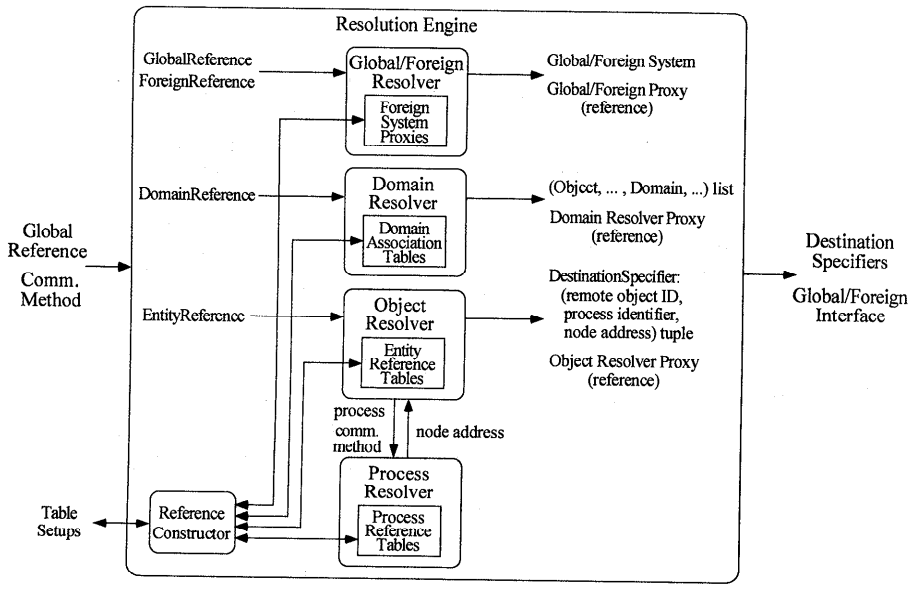


図3 参照解決の方式
Fig. 3 Reference resolution scheme.

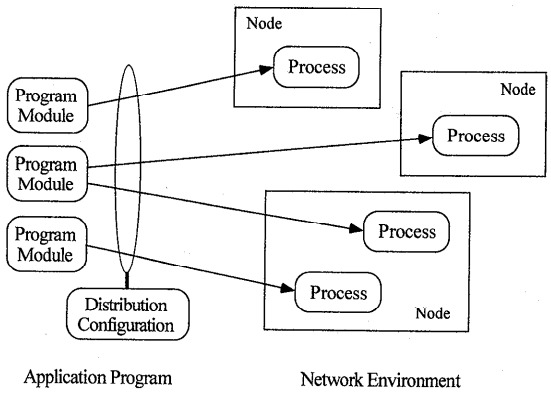


図4 アプリケーションプログラムの配置とプロセス
Fig. 4 Application program allocation and processes.

とができる．大域並行計算を実現するアプリケーションプログラムは，プロセスに相当する単位でプログラムモジュールとして分割し，ネットワーク上に分散配置する．その原理構成を図4に示す．プログラムモジュールは，共通機能の利用のため，複数のプログラムパッケージで構成する．プログラムパッケージは，複数のプログラムモジュールで共有してよい．分散配置は，アプリケーションプログラム上のコンフィグレーション設定によりシステムが行う．同一のプログラムモジュールから複数のプロセスを生成してもよい．

プロセスの生成時，そのプログラムモジュールが外部のリソースを参照している場合は，コンフィグレーションで指定された対応するプロセスに問い合わせてそのリソースの大域参照を取得する．プロセス生成時の相互参照の生成は動的に行うことが可能であり，プログラムモジュールを動的に追加して配置することができる．

アプリケーションプログラムは，プログラムモジュールのネットワークノードへの配置替え，あるいは，モジュール内の並行オブジェクトのクラスタ構成の変更により，アプリケーション構成のトポロジーを保存したままで，ネットワーク上へのスケーラブルな再配置が可能である．また，物理的なネットワーク構成に依存しない柔軟性を持った空間設定が可能である．

2.4 大域マルチメディア構造体の処理

大域データ構造体は，大域データを表す計算オブジェクトの相互参照された集合，あるいは，並行オブジェクト上でその集合を指し示すローカルオブジェクトである．ローカルオブジェクトは，並行オブジェクト間でメッセージによって伝達できる．大域データ構造体は，動的に変化してよい．また，複数の並行オブジェクトによって，その一部を共有してもよい☆．

連続メディアデータは，そのデータとストリーム配送機構を含めてオブジェクト抽象化し，ストリームオブジェクトとして表現し処理する．その機能構成を図5に示す．ストリームオブジェクトは，ストリームのソースとシンクの記述子と，その指定に基づいて

☆ 大域的な分散構造体に対する我々のさらなる試みは，その参照の動的解決と，非精密的処理を許容することである（2.5, 4.4節参照）．

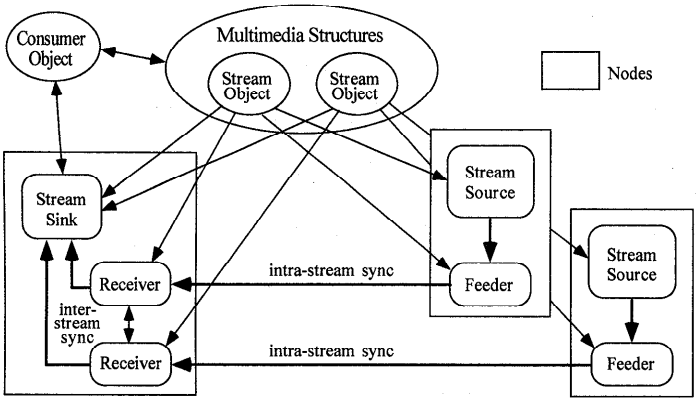


図5 連続メディア処理とマルチメディア構造体
Fig. 5 Continuous-media processing and multimedia structures.

ネットワークにわたるストリーム配送機構を生成し制御する機能とを備える．連続メディアデータは時系列であるが，その配送機構と制御をカプセル化することにより，プログラミング上他のデータと統一的に扱うことが可能であり，ストリームのバックトラック制御なども容易となる．また，複数のストリームオブジェクトを大域データ構造体の中で扱うことにより，構造体内部にストリーム間同期などの処理を組み込むことが可能である．

ストリームオブジェクトは，大域データオブジェクトであってもローカルオブジェクトであってもよい．大域データオブジェクトの場合は，ネットワーク上の任意の場所で機能させることができる．ローカルオブジェクトであれば，ネットワーク上を転送可能である．いずれの場合も，ストリーム配送機構は必要に応じて生成することが可能であり，また，複数のストリームオブジェクトによって同一の配送機構を制御してもよい．

上記の大域データ構造体に連続メディアストリームオブジェクトを含めて，大域マルチメディア構造体と呼ぶ．HTML等従来の方式によるマルチメディアデータもまた，1つのデータの形式としてここでの構造体の中で扱うことができる．我々が想定する大域的なマルチメディア構造体の例を，**図6** に示す．この例は，マルチメディアデータのシーケンスを，そのシーケンス記述と内容属性などの情報による構造体で表したものである（4.2節参照）．

2.5 空間とリソース認識のメディエーション

ネットワークメディエータは，近隣領域の特徴的なリソースを管理し，他のメディエータと連携してリソースの探索やリソース機能の呼び出しを行う☆．ネットワークメディエータもまた分散オブジェクトであり，システムのメタ機能として一般の並行オブジェクトから呼び出すことを可能とする．大域参照の機能によってその参照先が特定できなかったときや，大域参照によらず論理的属性に基づいて参照先を得たいとき，あるいは，論理的属性を指定して分散機能を実行したいとき，最寄りのネットワークメディエータを呼び出して，その探索や機能実行を行う．大域参照には，その参照先が特定できない場合，自動的にネットワークメディエータを呼び出すオプション機能を備える．

ネットワークメディエータは，**図7** に示すようなトークンをメッセージとして送信して，その機能を動作させる．トークンは，探索のための情報と，探索先での機能実行の指定，探索経路を示す情報からなる．メディエータの機能構成を，**図8** に示す．メディエータは，トークンを受信すると，自分が認識する領域のリソース情報とトークン情報とのヒューリスティックマッチング（たとえば文献31)，28)の方法）を行う．所望のリソースが得られない場合は，近傍の別のメディエータにさらにトークンを送信し，それらの連携により探索を進める．探索途中のトークンの処理は，トークンヒープ上に状態フレームを生成しその実行を管理する．探索結果は複数得られることがあるが，それらは必要に応じて利用する．また，領域を指定してそこでの全解探索を行うことが可能である．

3. 機能実現の方法

ここでは，提案する計算方式の動作環境として開発している大域計算フレームワークの機能実現方法について述べる．

☆ これは，文献30)で述べられているような分散系でのオープン性の本質を実現しようというものである．

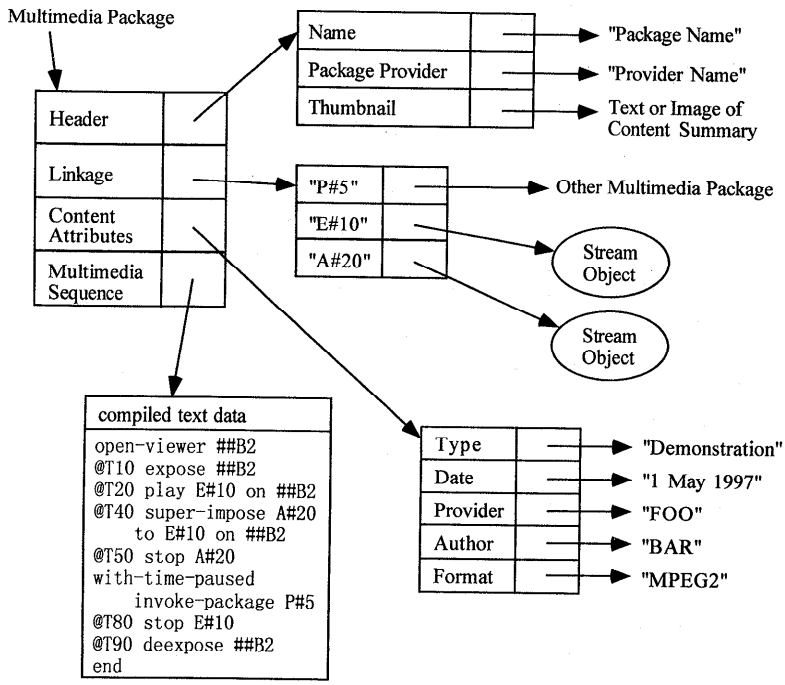


図6 マルチメディア構造体の典型例
Fig. 6 Exemplary multimedia structures.

```
Token:
(mediate
     search
     ((name "FOO VoD Server")
      (reference 'domain local node reference')
      (object-name "FOO server manager")
      (object-reference 'domain local object reference')
      (network-domain :network-provider)
      (problem-domain "multimeia-network-service")
      (application-domain "video-on-demand")
      (communication-medium internet))
     (destination-object method arguments)
     (return-value return-status)
     token-ID
     origin-node
     origin-mediator
     predecessor
     (successor successor ... successor)
     )
```

図7 メディエーショントークンの例
Fig. 7 Mediation token example.

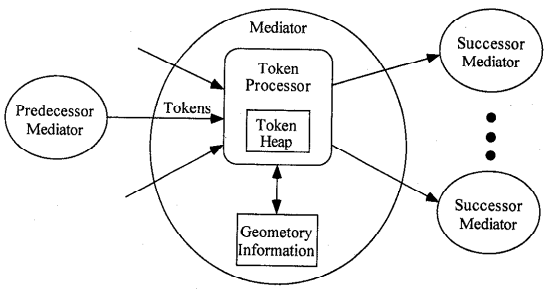


図8 ネットワークメディエーションの方式
Fig. 8 Network mediation scheme.

大域計算フレームワークのメッセージ通信機構を，**図9** に示す．この機構は，プロセスにおける並行オブジェクトの実行環境となる．メッセージの送受信は，メッセージ通信部（Message Passing Operator）で機能する（3.2 節）．送受信にともなう計算実体は，参照表と参照解決機能によって特定する（3.1 節）．受信したメッセージは，スケジューラでその実行を制御する（3.3 節）．アプリケーションプログラミング機能は，メッセージ送信と連続メディア処理に関連した API（Application Programming Interface）を提供する（3.4～3.6 節）．実行環境に対する SPI（System Programming Interface）では，OS，ハードウェアについての独立性に留意した構成とした．

以下では，すでに実装した機能を中心にその実現方法を述べる．

3.1 大域参照の管理

大域参照は，プロセス内のリモート参照表とローカル参照表で管理する．リモート参照表はメッセージ送信時やリソース特定時に，大域参照からネットワークノード，プロセス，リモート識別子を得るための管理表である．ローカル参照表は，リモート識別子とローカル識別子の対応表で，プロセス内で計算実体を特定するためのものである．メッセージ送信時のオブジェ

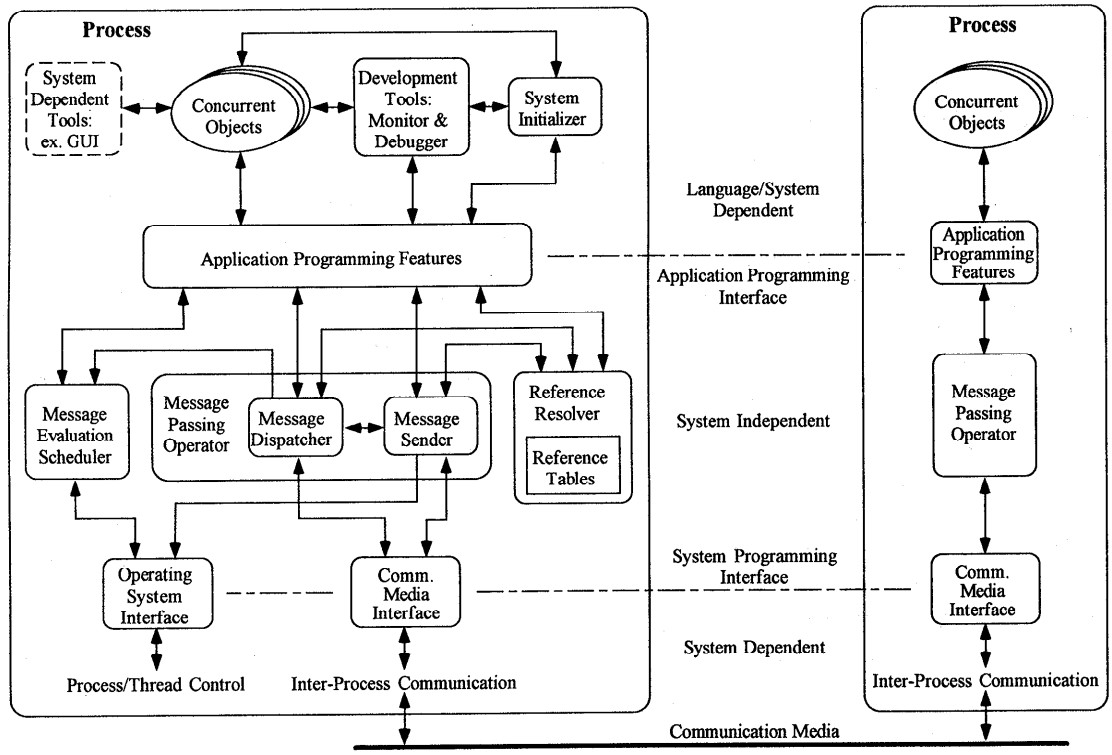


図 9 大域メッセージ通信機構
Fig. 9 Global message-passing feature.

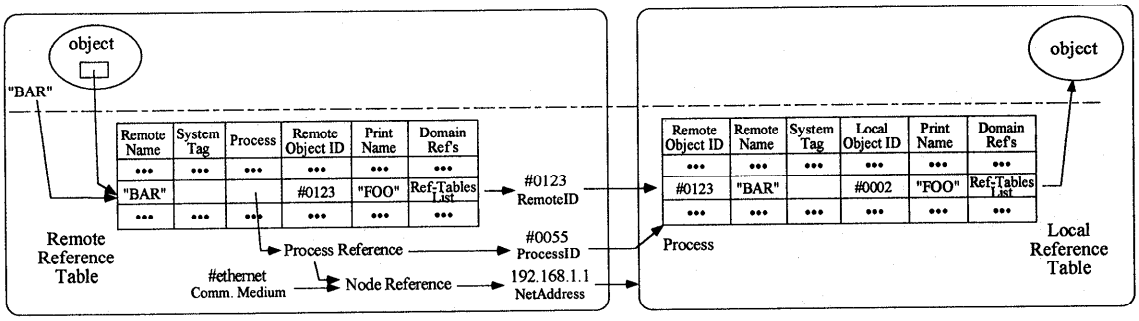


図 10 大域参照と並行オブジェクト特定の方式
Fig. 10 Global reference and concurrent-object identification scheme.

クト参照のスキームを例として，それら参照表の対応を図 10 に示す．大域参照にはまた，図に示すように論理名をつけて扱うことが可能である．多様な通信メディアに対応するためのメッセージ送信のパスは，通信メディアの指定によってメッセージ送信時に決定する．

参照表はプロセスの生成時に，プログラムモジュール間の参照関係に基づいて初期設定する（2.3.1 項，4.1 節参照）．また，分散オブジェクトが生成されたとき，大域参照をメッセージによって得たとき，および，新たなプログラムモジュールが配置されたときに，新たな参照を登録する．

大域ガーベジコレクションにおける参照カウントの処理や，オブジェクトマイグレーションにともなうメッセージ転送の処理は，参照表を作業領域として実行する．

3.2 メッセージの送信と受信

メッセージ送信では，API 上のメッセージ送信関数（Send, SyncCall, Call）の呼び出しによって，引数の構造体を認識して送信可能な論理形態に変換し（3.5

節参照)，送信先オブジェクトの参照解決を行い，送信の処理をする．送信先の所在によって，ネットワーク上のRPC，ノード内IPC，あるいは，プロセス内通信を選択して送信する．さらに広域性に対応するため，IIOP (Internet Inter-ORB Protocol)[5]を用いた送信機能を実装組込中である．送信先がドメインやオブジェクトクラスタになっていれば，マルチキャストを行う．マルチキャストは送信先の所在に基づいてルーティングを行う仕様であるが，現在単一メッセージの繰返しで実現している．

メッセージを受信するとメッセージディスパッチャは，受け取ったリモート識別子から受信オブジェクトを特定し，その処理を行うための関数記述子を生成する．関数記述子は，分散関数とオブジェクトメソッドの呼び出しあるいは共有変数のアクセスを起動するための情報と，メッセージの時間属性と優先度を保持したものである．関数記述子は，スケジューラに渡され処理される．メッセージ送信がSyncCallとCallによる場合は，スケジューラによる実行で得た評価値を，同じメッセージ送信機能により呼び出し側に送信する．

3.3 メッセージ評価のスケジューリング

スケジューラは，関数記述子を受け取り，スケジューラキューに格納する．分散オブジェクトでのメッセージ評価の排他性の処理は，スケジューラ前段の関数起動部で行う．

長期スケジューリングでは，評価を起動すべき関数記述子を，その時間属性と優先度に基づいて選択する．選択した関数記述子はスケジューラキューからスケジューラプールに移し，その実行スレッドを生成する．メッセージの時間属性としては，送信タイムや，評価のデッドラインタイム，計算の非精密性の許容度，メッセージの同期性と周期性といったものがあるが，現時点では送信タイムのみを扱っている．

短期スケジューリングでは，スケジューラプール上で可動状態となっている関数記述子の評価を，その属性に基づいて実行制御する．この制御は，OSのプロセスとスレッドの機能を用いて行っている．短期スケジューリングをOS機能によって行うのは，メッセージ評価以外のタスクのリアルタイム処理の観点からも妥当である．ただし，後述の実装システムではSolaris 2.4とWindows NT 4.0が提供する以上のリアルタイム化の処理はしていない．

3.4 プログラミングインタフェースと多言語対応

プログラミングインタフェースとして提供するのは，並行オブジェクトの基本定義，メッセージ送信関数，プログラム配置定義，プロセス間での大域参照のエクスポートとインポートの宣言の機能である．

現在我々はアプリケーション言語として，C++[32]をサポートしている．また，Java[33] APIとCommon Lisp[34] APIを構築中である．これらの言語のいずれも，計算は，並行オブジェクト間は並列に，並行オブジェクト内では逐次的に行う．大域共有変数は分散オブジェクトで実現し，変数アクセスのためのメソッドを提供している．ただし，分散処理の特性からtest-and-set機能はあるがロック機能は備えていない．変数アクセスにおいては，分散オブジェクトでのメソッド処理の排他性によりクリティカルセクションを実現している．

Common Lispオブジェクト（CLOSオブジェクト）を複数引数に持つメソッドが呼ばれたとき，前述のように計算スコープは局所的であり，直接アクセスできる分散オブジェクトはたかだか1つで，残りは大域参照である．

C++ APIとJava APIにおいては，それらに固有の機能に拘束されない柔軟な形で分散処理を記述するためのプログラムトランスレータを実現した．トランスレータは，並行オブジェクトやメッセージ通信関数を用いたユーザプログラムから，分散オブジェクトの生成手続き，メッセージ送信手続き，メソッド/関数呼び出し手続き，データ構造体の構造変換手続きなどを自動生成する．また，プログラム配置情報に基づいて，大域参照のエクスポート，インポート手続きの生成を行う．

3.5 メッセージ上のデータ構造体の処理

並行オブジェクト間のメッセージ上でデータ構造体を扱うには，データ構造を論理形式で処理しなければならない．したがって，上述したメッセージ通信では，送信する引数のデータ構造をトレースし，その構造体要素を論理形式にフォーマットする機能と，論理形式から元のデータと同じ構造体を生成する機能を持つことで，データ構造体の送受信を実現する．ただし，現在は循環構造は扱えない．部分構造を共有している場合は，受信側でその共有箇所が複製される☆．

Common Lispではデータの動的型チェックが可能であるので，データ構造体の変換機能はフレームワークシステムが提供する．C++とJavaではトランスレータを利用して，この機能を提供している．ただし，C++ APIでは，配列は，実行時の構造認識ができな

☆ 送受信側双方にわたって言語処理系が均質であれば，ポインタの補正を行うことで物理イメージを送信することが可能である(たとえば文献35))．しかしその効果に比して，そのための均質性の同定を非同期系で行うことは得策ではない．

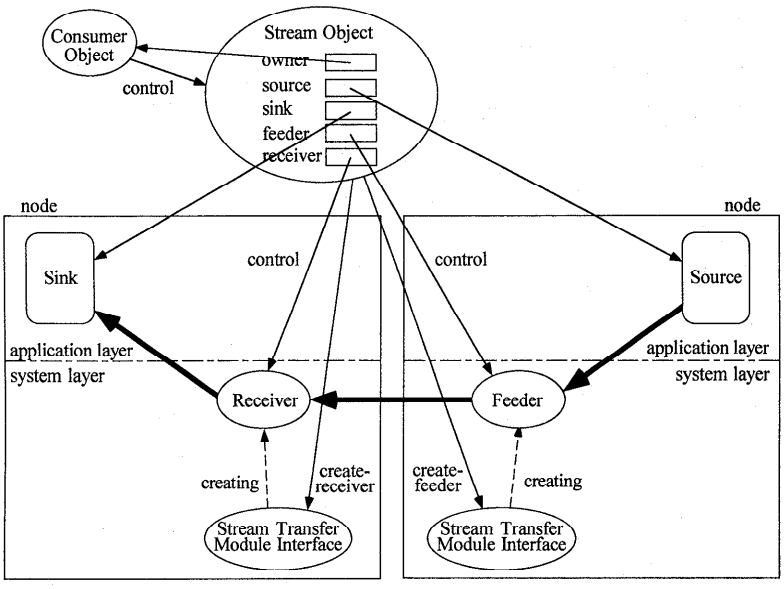


図11 ストリームオブジェクトと連続メディア処理
Fig. 11 Stream object and continuous-media processing.

いため，structの中に埋め込んで使用しなければならない．

3.6 連続メディア処理の方式

ストリームオブジェクトとして実現した連続メディア処理の機能構成を，**図11** に示す．API上ではそのオブジェクトメソッドとして，ストリーム送信の開始，終了や一時停止，再開といった制御が可能である．その機能は，オブジェクト内で配送機構の各モジュールを適切に制御して実現される．ストリーム配送機構は，我々が開発しているネットワークスケーラビリティを考慮したデータ配送の機能[36]を使用している．複数のストリームを同期させて使用するときは，それらのストリームオブジェクトの相互連携で同期処理を実現する☆．

4. 実験システムと評価

機能実現方法について上述した大域計算フレームワークを，実際の計算機ネットワーク上に実装した．そのフレームワークシステム上にさらに，Video-on-Demand（VoD）型の情報提供サービスアプリケーションを，実験問題として試験構築した．ここでは，試作フレームワークシステムと実験アプリケーションに基づいて，提案する方式と機能の評価を行う．

☆ これはアプリケーション制御での同期によって行うが，ストリーム受信機能の相互作用による同期が実現されれば，より高性能なストリーム間同期が可能となる．

4.1 大域計算フレームワークシステム

フレームワークシステムは，Solaris 2.4/SparcStationとWindowsNT 4.0/IBM PC/AT互換機，ONC（Open Network Computing）RPC[37]の環境上で，C++をAPI言語として稼働している．API以下のシステム依存部は，基本実行性能の良さなどの観点からC++で実装した．SPIではPOSIX[38]への対応を考慮した．通信メディアは現時点では（狭義の）インターネットにのみ対応している．

計算機ノード上には，プロセス生成機能を持った管理プロセスを常駐させ，そのプロセスによって新たなアプリケーション空間を生成できる構成とした．図4に原理を示したプログラム分散配置は，そのための配置プロセスを用意して実現した．配置プロセスは，プログラムファイルとノードアドレスからなる空間構成情報を基に，対象ノード上の管理プロセスに指示して，アプリケーションプロセスを生成し，さらにプロセス間での大域参照の交換の制御を行う．配置プロセス上にはGUIを構築して，空間構成情報の編集と分散配置の操作のためのユーザフロントエンドを用意した．

開発支援機能として，プログラム中に埋め込んで所望の情報を発信するプローブ機能と，プローブからのメッセージを受信するコレクタオブジェクトを実装した．この機能は，デバッグやチューニング，統計などに活かすことができる．プローブ・コレクタを用いた組込機能としてまた，アプリケーション実行の振舞いを観測するためのメッセージトレースGUIを実現し

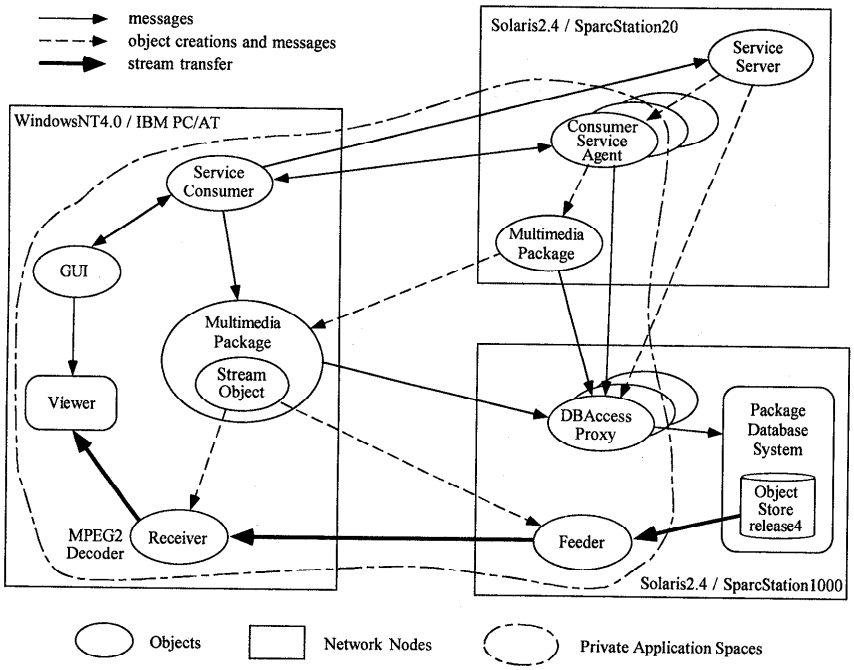


図12 実験アプリケーションシステムの構成
Fig. 12 Experimental application system configuration.

た．このGUIは，プロセス間のメッセージ交換の様子のリアルタイム表示と，並行オブジェクト間のメッセージ送信の時系列表示の機能を備える．

4.2 実験アプリケーション

試験実装するアプリケーションは，提案する計算方式と機能に対する実験問題として次の性質を持つことが求められる．

- アプリケーション空間はネットワークシステム上に動的に生成され機能すること
- オブジェクトは動的に生成され空間内のオブジェクト構成は動的に変化すること
- 計算オブジェクトは非同期メッセージを送信しながら並行に動作すること
- 空間上に大域データを生成しオブジェクト間で共有すること
- ネットワークにわたる連続メディア処理を行うこと

これらを満たした形で構築した実験アプリケーションの構成を，**図12**に示す．図でプロセスの表示は省略した．ServiceServerは，VoD型サービスを管理している永続的オブジェクトで，そのアプリケーション空間とプロセス，オブジェクト名を指定してそれへのメッセージの送信ができる．ServiceConsumerは動的に生成され，その求めに応じてServiceServerはConsumerServiceAgentとDBAccessProxyを生成する．ServiceConsumerとConsumerServiceAgentはその関連するオブジェクトとあわせて，1つのアプリケーション空間を形成し，その空間上で協調してマルチメディア情報提供の制御を行う．提供情報は，データベースからの取得後大域データ構造体として扱い，空間内のオブジェクトが連携して操作する．そのデータ構造体は，図6の形式（マルチメディアパッケージと呼ぶ）で表現した．ただし，実験ではコンテント属性は省略し，マルチメディアシーケンスは単一のMPEG2ストリームのみを扱った．

マルチメディアパッケージは連続メディアデータを含み，ServiceConsumerあるいはConsumerServiceAgentの制御によりそのストリームの転送を行う．マルチメディアパッケージオブジェクトは，ServiceConsumerとConsumerServiceAgentの機能作用の形態に応じて処理しやすいノードに生成させた．データベースには，マルチメディアパッケージに対応する単位で，その内容を構造タグとその値によるSGML[39]形式で記述して格納した．データベースは，タグの指定による構造単位のアクセスが可能である☆．

4.3 方式と機能の評価

大域計算フレームワークシステムの実装と実験アプ

☆ ServiceConsumerなどのサービスオブジェクトの実装は，当研究所フレームワークグループの助力による．ストリーム転送機能とデータベースの構造アクセスの機能はそれぞれ，同プロトコルグループとデータベースグループの提供による．

リケーション構築による知見に基づいて，提案する計算方式と機能実現方法を応用性の観点から評価する．

(1) 計算空間のスケーラビリティ

我々の大域計算方式では，プログラム配置とメッセージ通信機構の性質から，計算空間の構成は，実行環境の形態には拘束されない．実験アプリケーションは，その開発段階と試験形態に応じて，プロセスとオブジェクトの配置と構成を柔軟に変更しながら構築していくことができた．実現した参照管理方式では，オブジェクトやメソッドといった計算実体は大域参照から直接特定できるため，分散メモリマルチプロセッサのような密なシステム上でも効率の良いメッセージ送信が実現できる．計算空間のスケーラビリティは，広域のネットワークから局所的なシステムに及ぶ．

大域参照は，それを有するプロセスに分散して管理している．参照は正しく手繰れれば機能として十分であるので，計算空間の全体はそれで適切に管理される．大規模なアプリケーションシステムでも，参照空間管理の組合せ爆発的な破綻を招くことがない．

(2) マルチメディア適応性

API での連続メディアデータの扱いは，時間による変容性の異なるデータを同じ演算系で処理できる手段を提供している．マルチメディアパッケージという大域データ構造体は，異なる種類のデータによる構造体の抽象化を実現している．実験アプリケーションでの連続メディアストリームの配送制御は，マルチメディアパッケージの処理の過程の中で，ストリームオブジェクトの演算の形で実現できた．マルチメディアパッケージはまた，複数の並行オブジェクトの協調動作のもとで処理することができた．連続メディア処理と大域マルチメディア構造体の機能は，広域のマルチメディア処理のための１つの枠組みを実現している．

大域参照は，空間構成に基づいた多段的な構成とすることができる．メッセージ送信では，使用する通信メディアを選択できる．これらの組合せにより大域計算フレームワークは，多様なネットワーク形態にも対応することが可能である．マルチメディア処理の広域性とあわせて，提案する計算方式は，多様なメディア形態を統合化させたシステムの構築を支援するものであるといえる．

(3) プログラミングの柔軟性と規範性

プログラミングにおいては，プロセスとオブジェクトの配置と構成の柔軟性に加えて，メッセージ機能に柔軟性を持たせた．オブジェクト機能の呼び出しでは，個々の呼び出しインタフェースやプロトコルを規定する必要はない．メッセージ引数においては，データ構造体の使用を制限しない．システム開発を容易とするこのような柔軟性は，実験アプリケーションの構築でも有効であった．

実験アプリケーションでは，各オブジェクトの機能はメッセージによって駆動し，システム全体は ServiceConsumer と ConsumerServiceAgent やマルチメディアパッケージオブジェクトに対する非同期のメッセージ作用の相互連携によって協調動作させた．大域計算フレームワークが提供する分散処理の方式は，遅延制約が少なく並行性の高いシステムの構築を促すパラダイムとなっている☆．

(4) システムの均質性と拡張性

大域計算フレームワークは，メッセージ通信とプロセス管理の機構をベースとして，その上に構築されるシステム機能とアプリケーションに対し，その機構によるユニフォームで拡張性のある系を提供する．フレームワークシステムで実現した管理プロセス，配置プロセスとそこでのオブジェクトも，フレームワーク API 機能の中で作成した．系の均質性と同時に，効果的な参照管理による効率の良いメッセージ通信を実現しており，廉価で機能性能の高いフレームワークシステムの実装ができているといえる．また，大域計算フレームワークのコア機能は C++ライブラリとして提供可能であり，高い移植性を持つ．

フレームワークにおけるアプリケーション空間は，他の空間やそこでのオブジェクトを参照する形で階層的に形成することができる．実験アプリケーションでは，ServiceServer を有する基幹システムに追加する形で，ServiceConsumer による個別のアプリケーション空間を構成することができた．計算空間もまた，均質で拡張性のある構成が可能であり，大規模でオープンなアプリケーションの構築を支援するものとなっている．

4.4 課題と今後の予定

試作したシステムによる実験では，広域性の検証が十分ではない．現在実装中の IIOP によるメッセージ通信機能を用いて，より広域での実証を行う予定である．また応用面から，広域環境での計算手段の１つである HTTP や大域的データの表現手段である HTML など既存の方式との連携を試行することを検討している．通信メディアについてはインターネットに加えて，放送をシミュレートした通信系を実装することで，非均質で非対称なネットワーク上での実験を行う予定である．

☆ 非同期性に関するより高度な実験例は文献 28) を参照．

非同期分散処理には，メッセージの追越しやオブジェクト状態の複雑化など種々の問題をともなうが，文献 28) で述べたような非同期処理のための組込みユーティリティを実現する予定である．今後のマルチメディア応用ではリアルタイム処理の機能は不可欠である．デッドライン制御などのより高機能なメッセージ評価スケジューリングと，連続メディア処理との連携も含めたリアルタイム処理の実験を行う予定である．広域で実用規模のシステムを稼働させるには，ネットワークにわたる計算リソースの再生利用手段が必要である．大域ガーベジコレクションの実現は，今後の実装課題である．

フレームワークシステムの試作を基に，提案する計算方式と機能実現方法の有効性を論じたが，さらに今後の実験と検証を通じたシステムの性能評価が求められる．また，大域計算の要件の 1 つとしてあげたネットワークメディエーション機能は，まだ実装されていない．広域の応用実証の中で，その機能の実装と検証を行う予定である．

広域のネットワーク計算では一方，計算の信頼性や，リソースに対するセキュリティの問題を解決しなければならない．広域の分散処理という性質から計算の実行は必ずしも保証されないため，大域的な非精密的計算とデータ処理の枠組みが求められる．また，プログラム配置やリソース参照において，その要求属性を基にセキュリティチェックをかける機能が望まれる．これらは現在我々の計算方式が備えているものではないが，付帯機能として組み込んでいくことが可能である．

5. まとめ

本論文では，広域のネットワーク環境上で実現する大域計算の方式を提案し，それに基づいて我々が開発している大域計算フレームワークについて報告した．

この大域計算の方式は，多様な通信メディアや実行環境の違いに適応し，広域での分散処理とマルチメディア処理の特性に基づいた計算実行が可能な性質を備えている．そこでは，大域計算空間を広域の環境上に柔軟に形成し管理することができる．計算リソースは大域参照で管理し，計算オブジェクト間では大域参照を用いた効率の良いメッセージパッシングが可能である．また，連続メディアデータを含む大域データ構造体の処理を採り入れている．ネットワーク上のリソースは，動的に探索して利用することが可能である．

この方式に基づいて開発しているフレームワークシステムは現在，Solaris 2.4 と WindowsNT 4.0 の環境上で，C++をアプリケーションプログラミング言語とするプロトタイプが稼働中である．プロトタイプ上に実験アプリケーションを構築し，提案する計算方式とその機能実現方法の有効性を確かめた．今後，さらに広域での検証を進めるとともに，リアルタイム性能の向上と，大域ガーベジコレクションの実現，および，ネットワークメディエーションによるオープン性への対応を行う予定である．この大域計算フレームワークは，実行環境への依存が少ないミドルウェアコアとして実装しており，種々の環境上へそこでの機能を活用したうえでの移植が容易である．

ネットワーク技術や半導体技術の発展と呼応して，広域のネットワーク上での高度情報処理が今後ますますその重要性を増すと考えられる．我々の大域計算方式では，広域の多様な実行環境にわたってシステムを展開し計算を実行することが容易である．また，今後のマルチメディア情報処理の重要性を考慮して，連続メディアデータを含む情報構造体を大域的に扱えるようにした．大域計算フレームワークの開発を通じて，その優位性を検証していく予定である．

謝辞 フレームワークシステムのプログラム配置機能と開発フロントエンド GUI の作成は，当研究所小早川雄一研究員の助力による．研究の方向付けに関して，同五十嵐達治部長から有益な助言を得た．

参考文献

1) 前川博俊ほか：ネットワーク環境におけるマルチメディア並行計算プラットフォームの実現，マルチメディア通信と分散処理ワークショップ論文集，pp.417–424, 情報処理学会 (1996).
2) 前川博俊：HD マルチメディア情報サービスプラットフォーム—非同期ネットワーク計算と連続メディア処理，第 52 回情報処理学会全国大会論文集，Vol.3, pp.225–226 (1996).
3) 小早川雄一ほか：ネットワークスケーラブルな分散オブジェクト空間管理方式，情報処理学会研究報告，97-DPS-80，pp.211–216 (1997).
4) Berners-Lee, T., et al.: The World-Wide Web, *Comm. ACM*, Vol.37, No.8, pp.76–81 (1994).
5) *The Common Object Request Broker: Architecture and Specification*, Wiley, New York (1991).
6) Beguelin, A., et al.: A Users' Guide to PVM (Parallel Virtual Machine), ORNL/TM-11826, Oak Ridge National Laboratory, Oak Ridge (1991).
7) Snir, M., et al.: *MPI: The Complete Reference*, MIT Press, Cambridge (1996).
8) Grimshaw, A.S.: Easy-to-Use Object-Oriented Parallel Processing with Mentat, *IEEE Com-*

puter, Vol.26, No.5, pp.39–51 (1993).
9) 小中浩喜ほか：超並列オブジェクトベース言語Ocoreの並列計算機上での実装，情報処理学会論文誌，Vol.36, No.7, pp.1520–1528 (1995).
10) 塚本享治ほか：クラスの共有と配送に基づくオブジェクト指向分散システムの設計と実現，情報処理学会論文誌，Vol.37, No.5, pp.853–864 (1996).
11) Achauer, B.: The DOWL Distributed Object-Oriented Language, *Comm.ACM*, Vol.36, No.9, pp.48–55 (1993).
12) Chandra R., Gupta, A., and Hennessy, J.L.: COOL: An Object-Based Language for Parallel Programming, *IEEE Computer*, Vol.27, No.8, pp.13–26 (1994).
13) Yonezawa, A., et al.: Implcmcnting Concurrent Object-Oriented Languages on Multicomputers, *IEEE Parallel and Distributed Technology*, Vol.1, No.2, pp.49–61 (1993).
14) Adler, R.M.: Distributed Coordination Models for Client/Server Computing, *IEEE Computer*, Vol.28, No.4, pp.14–22 (1995).
15) Lewis, T.G.: Where Is Client/Server Software Headed? *IEEE Computer*, Vol.28, No.4, pp.49–55 (1995).
16) Kramer, D.: *The Java Platform: A White Paper*, Sun Microsystems, Mountain View (1995).
17) Wollrath, A., Riggs, R. and Waldo, J.: A Distributed Object Model for the Java System, *Proc. 2nd Conf. Object-Oriented Technologies and Systems*, pp.219–231, USENIX, Berkeley (1996).
18) 平野 聡，塚本享治：分散Java実行のためのポータブルなORBの構成法，情報処理学会研究報告，OS73-10, pp.55–60 (1996).
19) 白鳥則郎，菅原研次：やわらかいネットワークの開発に向けて—知識型設計方法論，情報処理学会研究報告，DPS62-11, pp.79–86 (1993).
20) 富永英義：フレキシブルネットワーク—総論，電子情報通信学会誌，Vol.77, No.4, pp.350–354 (1994).
21) Awerbuch, B. and Schulman, L.J.: The Maintenance of Common Data in a Distributed System, *J. ACM*, Vol.44, No.1, pp.86–103 (1997).
22) 宮西洋太郎ほか：分散システムにおけるデータの複製管理方式，情報処理学会論文誌，Vol.37, No.5, pp.830–839 (1996).
23) *IMA Draft Recommended Practice: Multimedia System Service*, First Edition, Interactive Multimedia Association, Annapolis (1995).
24) 徳田英幸：Real-Time Mach：実時間マイクロカーネル，情報処理，Vol.37, No.12, pp.1117–1124 (1996).
25) 西尾信彦，徳田英幸：連続メディア処理のためのミドルウェアの性能評価，情報処理学会研究報告，OS69-10, pp.55–60 (1995).
26) 萩原四郎，西村 康，大竹和雄：HDマルチメディア情報サービスプラットフォーム—システムコンセプト，第52回情報処理学会全国大会論文集，Vol.3, pp.223–224 (1996).
27) 萩原四郎ほか：HDマルチメディア情報サービスプラットフォーム，画像電子学会技術研究予稿，MT-6-S1-2 (1996).
28) Maegawa, H.: ConClass: A Framework for Real-Time Distributed Knowledge-Based Processing, *IEEE Trans. Knowledge and Data Engineering*, Vol.6, No.6, pp.909–919 (1994).
29) Bevan, D.I.: An Efficient Reference Counting Solution to the Distributed Garbage Collection Problem, *Parallel Computing*, Vol.9, pp.179–192 (1989).
30) Hewitt, C. and de Jong, P.: Open Systems, *On Conceptual Modeling*, Brodie, M.L., et al. (Eds), pp.147–164, Springer-Verlag, New York (1984).
31) Clancy, W.: Heuristic Classification, *Artif. Intell.*, Vol.27, pp.289–350 (1985).
32) Stroustrup, B.: An Overview of C++, *ACM SIGPLAN Notices*, Vol.21, No.10, pp.7–18 (1986).
33) Arnold, K. and Gosling, J.: *The Java Programming Language*, Addison-Wesley, Reading (1996).
34) Steel Jr., G.L.: *Common Lisp: The Language*, Second Edition, Digital, Bedford (1990).
35) Hamazaki, Y., et al.: The Object Communication Mechanisms of OZ++: An Object-Oriented Distributed Programming Environment, *Proc. 9th Int. Conf. Information Networking*, pp.425–430 (1994).
36) 谷 英明，谷口幸治：HDマルチメディア配送システムの開発—リンク接続によるEnd-to-End QoS管理機構，マルチメディア通信と分散処理ワークショップ論文集，pp.395–400, 情報処理学会 (1996).
37) Bloomer, J.: *Power Programming with RPC*, O'Reilley & Associates, Sebastopol (1991).
38) *Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface API [C Language]*, IEEE, New York (1996).
39) Goldfarb, C.F. and Rubinsky, Y.: *The SGML Handbook*, Clarendon, Oxford (1990).

(平成9年 5 月19日受付)
(平成9年11月 5 日採録)

前川 博俊（正会員）

1979年大阪大学工学部応用物理学科卒業．1982年同大学大学院工学研究科修士課程修了．同年ソニー（株）入社．1989年より2年間Stanford大学Knowledge Systems Laboratory客員研究員．この間，並列Lispマシン，設計シミュレーション支援システム，分散協調・問題解決システム等の製作研究に従事．1995年より（株）ディジタル・ビジョン・ラボラトリーズに出向し，実時間分散マルチメディア処理の研究に従事．並列処理と人工知能のためのアーキテクチャに興味を持つ．ACM，IEEE，JSAI，AAAI各会員．昭和58年度本学会論文賞受賞．

斉藤 隆之（正会員）

1983年慶応義塾大学工学部応用化学科卒業．1985年同大学大学院工学研究科応用化学専攻前期博士課程修了．1988年同大学大学院理工学研究科物理学専攻後期博士課程単位取得退学．同年（株）リクルート入社．並列計算機を中心に高速数値計算技術の研究に従事．1990年ソニー（株）入社．物性研究のためのコンピュータ支援環境の構築，第一原理分子動力学法の研究に従事．1995年11月より（株）ディジタル・ビジョン・ラボラトリーズに出向中．

千葉 哲央

1990年静岡大学工学部電子工学科卒業．1992年同大学大学院工学研究科電子工学専攻修士課程修了．同年（株）富士通静岡エンジニアリング入社．1995年より（株）ディジタル・ビジョン・ラボラトリーズに出向．分散オブジェクト処理の研究に従事．