マルチメディア通信と分散処理 80－36
グループウェア 21－36
（1997. 1. 31）

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

小早川 雄一，斉藤 隆之，前川 博俊
（株）ディジタル・ビジョン・ラボラトリーズ

ネットワーク技術の発達とそこで実現されるアプリケーションの多様化に伴い，広域のネットワーク環境上に柔軟で動的にシステムを構築できるようなアプリケーション実行・管理の機構が必要となっている．我々は，ネットワーク環境における分散アプリケーション空間の形成とその空間上の計算オブジェクトの生成・管理を柔軟且つ動的な形で実現することが可能な空間管理方式を提案する．本稿ではまた，我々が開発中の並行計算プラットフォームとそこでの本方式の実装について述べる．

The Management Scheme of Distributed Object-Space For Network Environment

Yuichi Kobayakawa, Takayuki Saito, Hirotoshi Maegawa
Digital Vision Laboratories Corporation

We propose a management scheme for distributed application space and its computation objects. The scheme is capable of flexibly and dynamaically organizing application systems and managing objects creation and allocation over network environment. We also describe an implementation of the management scheme on a concurrent computation platform that we are developing.

1.はじめに

ネットワーク技術の発達に伴い，ネットワークワイドな多様な局面に柔軟に対応して動作するアプリケーションシステムの必要性が高まっている．

従来の分散処理やネットワーク計算に関するシステムではしかし，ネットワーク構成に柔軟に対応してアプリケーション個別の自由な空間を生成して管理する機能が十分ではない．CORBA Name Service[1]のような集中型のオブジェクト管理機構は，広域な並行計算には向いていない．また，分散プログラミング言語や実行環境[1]に対しては，ネットワーク計算のための機構をさらに構築する必要がある．WWW をベースとしてデータやプログラムをダウンロードしながら計算を進める方式[7,8]は，その実行効率と対話性に限界がある．ネットワーク構成の観点からは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[9,10]が，さらにその上で動作するアプリケーション実行のための機構の実現が求められる．

本研究ではアプリケーションの実行空間を柔軟且つ動的に生成・管理できる空間管理方式を提案する．本論文ではまた，広域のネットワーク環境上でのマルチメディア情報サービスの実現[11,12]を目的として我々が開発している並行計算プラットフォーム[13,14]と，そこでの本管理方式の実現について述べる

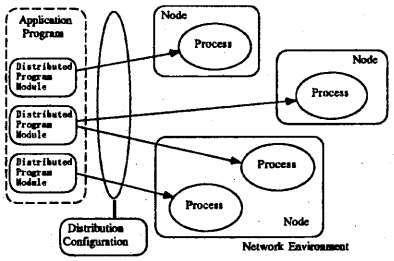
2.アプリケーション空間の管理

2.1.アプリケーション空間とオブジェクトの構成

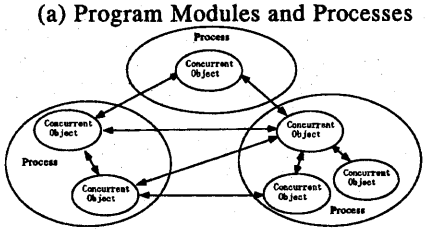
分散アプリケーションは,ネットワークワイドに分散した計算実体である並行オブジェクト[2]により構成する．ひとつの分散アプリケーションを構

[1] 例えば，[2-6].

[2] 並行オブジェクトは，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[14].



(a) Program Modules and Processes



(b) Concurrent Objects and Processes

図 1. Application Space and Process Distribuion

成する並行オブジェクトの集合を，アプリケーション空間と呼ぶ．異なるアプリケーションは，それぞれ個別のアプリケーション空間を形成する．

アプリケーションプログラムは，プログラムパッケージ[3]により構成する．アプリケーションの分散配置は，プログラムパッケージの集合であるプログラムモジュールを単位として行う．1つのプログラムパッケージは複数のプログラムモジュールに属していてもよい．分散アプリケーションの起動は，プログラムモジュールをネットワーク上に分散配置することにより行う．各プログラムモジュールは，実行状態においてプロセスとして管理する．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(図 1.a)．プロセスは，アプリケーション実行時における並行オブジェクトの管理単位であり，並行オブジェクトを保持する[4]．並行オブジェクト及びプロセスの所在は，参照により管理する．参照の生成は，アプリケーション実行時に動的に行う．参照にはさらに，名前をつけることができる．参照および名前の空間は，アプリケーションごと個別に管理する．

分散アプリケーションの実行は，並行オブジェクトが相互にメッセージを交換しあい計算を進めることで実現する(図 1.b)．メッセージ送信先の並行オブジェクトの特定は，参照または名前により行う．このアプリケーション空間上で有効な参照/名前は，各プロセス毎に存在させる名前・参照解決器により，メッセージ通信を行うネットワークシステム上で有効な形式に変換する．

2.2. 並行オブジェクトの参照と管理

並行オブジェクトはその属するプロセスにおいて，外部からの参照を表現するためのリモート識別子とプロセス内で実際の所在を特定するためのローカル識別子で管理する．並行オブジェクトへの参照は，それを保持するプロセスへの参照とそのプロセス内で割り当てられるリモート識別子の組で表現する．

プロセスへの参照は，ネットワークノードへの参照とノード内のプロセス識別子の組で表現する．ノードへの参照は，ネットワークシステム上の物理的識別子(ネットワークアドレス)あるいは名前で表現する．

任意の並行オブジェクトやプロセスをまとめてドメイン空間を構成することができる．ドメインは，別のドメインを含むことができる．ドメインは，アプリケーション空間を跨がって形成することが可能であり，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．

また，参照には名前をつけることができる．参照と名前はプロセス毎に存在する参照表により管理する．

並行オブジェクトへのメッセージ送信は，リモート参照表からそのオブジェクトを保持するプロセスとそのプロセス内のリモート識別子を特定し，さらにプロセス参照表からネットワークノードとプロセス識別子を特定することにより行

[3] プログラムパッケージは，関数，メソッド，クラスを包括したプログラムの単位．

[4] プロセスは，アプリケーションの実行制御とオブジェクト間通信機能を提供する．

う. 受信プロセスでは，ローカル参照表を用いリモート識別子をローカル識別子に変換し，宛先並行オブジェクトへメッセージを伝達する. (図 2)

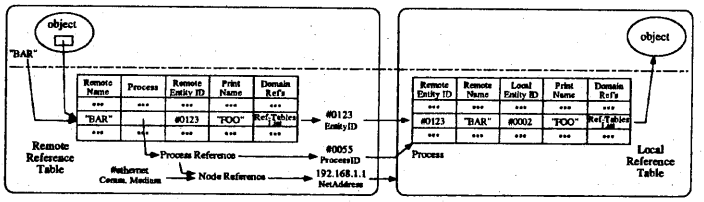


図 2. Remote Object Identification Scheme

2.3.参照と名前の解決

名前・参照解決器は，参照/名前をネットワークシステム上で有効な形式に変換する. 解決結果に新たな参照/名前が含まれる場合は，再度解決器に入力し解決を行う. この作業は，すべての参照/名前がシステム上で有効な解決結果となるまで再帰的に行う.

名前・参照解決器の論理構成を図 3 に示す. Entity Resolver は，参照/名前をプロセス参照/名前とリモート識別子の組に解決する. Process Resolver は，プロセス参照/名前を，通信メディア[5]の指定に従いネットワークノード情報とプロセス識別子の組に解決する. 参照/名前が WWW 上の URL 等の外部システム上のものであれば，Global/Foreign Resolver により解決する. Domain Resolver はドメイン参照/名前をそのドメインを構成するサブドメイン，並行オブジェクト，プロセスの参照のリストに解決する.

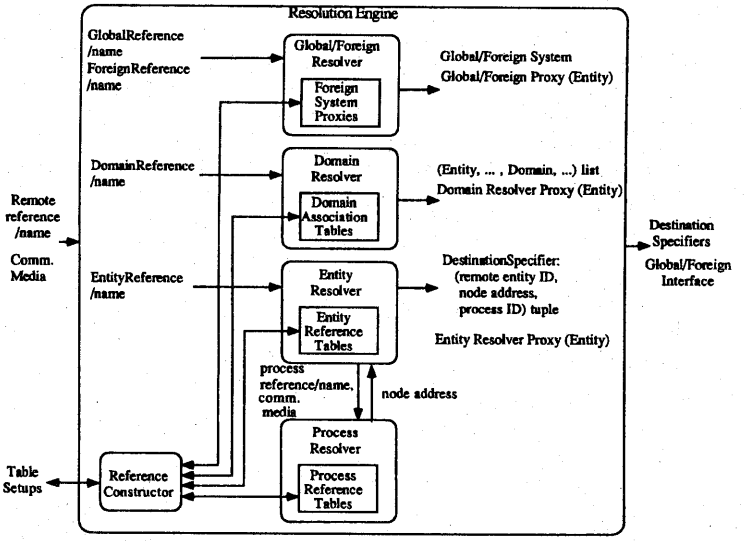


図 3. Reference Resolution and Resolution Engine

2.4.アプリケーション空間の生成

アプリケーション空間の生成は，その空間構成情報に基づいてプログラムモジュールを対象となるネットワークノードに分散配置し，配置後そのアプリケーションを構成するオブジェクトの相互参照を確立することにより行う.

空間構成は，プログラムモジュールの所在と配置先ネットワークノードの対応，プログラムモジュールとプログラムパッケージの対応，ネットワークノードとそのネットワークアドレス及び通信メディアとの対応からなる.

相互参照の確立は，参照の輸入手続きと輸出手続きとによって行う. 輸出手続きは，アプリケーションプログラムの構成から並行オブジェクトの参照/名前を予め判断し，そのオブジェクトのローカル識別子と各プロセスで生成するリモート識別子をローカル参照表に登録して行う. 輸入手続きは，参照するオブジェクトを保持するプロセスへの参照を空間管理機構から取得し，さらにそのプロセスにそのオブジェクトのリモート識別子を問い合わせてそれらをプロセス参照表とリモート参照表に登録して行う.

参照確立の手続きはアプリケーション実行時に，新たなプログラムモジュールに対して行うことが可能であり，アプリケーション機能の動的な

[5] (狭義の)インターネットやケーブル等.

ダウンロードを可能にする.

3. 並行計算プラットフォームと空間管理方式の実装

我々はまた, 並行計算プラットフォームを開発中であり, その上に提案する空間管理方式を実現している. このプラットフォームは, 分散アプリケーションの実行環境を提供するもので, 並行オブジェクト間のメッセージ送受信機能及び名前・参照の管理・解決機能を備える[14].

3.1. プラットフォームの並行計算の方式

並行計算プラットフォームは, 次のパラダイムに基づく並行計算機能を提供する:

- 分散手続き(関数)呼出
- 分散オブジェクト指向計算
- ネットワーク大域共有変数

並行オブジェクト間のメッセージ通信は, 次の方式に基づく:

- 非同期メッセージ送信(返値を持たない一方向の通信)
- 完全同期呼出(返値が来るまで呼出側がブロックする)
- 遅延評価型同期呼出(返値にアクセスしたときブロックする)

メッセージ上では, 任意のデータ構造体を送ることができる. メッセージによって起動される計算は, 並行オブジェクトの存在するプロセスにおいて実行する.

メッセージ送信では, 送信先並行オブジェクトの参照/名前を, 名前・参照解決器(送信部)により解決し, 並行オブジェクトを保持するプロセスと, 並行オブジェクトのリモート識別子を特定し, 処理を行なう. 送信先がドメインの場合は, マルチキャストを行なう. 送信先の所在によって, ノード間 IPC, ノード内 IPC, あるいは, プロセス内通信のうち最適な通信手段を選択して送信する. メッセージ受信では, 名前・参照解決器(受信部)により, リモート識別子をローカル識別子に変換し, 宛先並行オブジェクトへメッセージを伝播する.

3.2.参照表

名前・参照解決器が使用する参照表の構成を図 4～図 7 に示す.

ドメインに属するオブジェクトやプロセスは, それらとドメインとの相互関係をそれぞれの参照表で管理する.

リモート識別子からのローカル識別子の特定, 名前からの参照の特定は, ハッシュテーブルを用いて行う. オブジェクトの印刷名も併せて参照表で管理する. これは, 主にデバッグ等で利

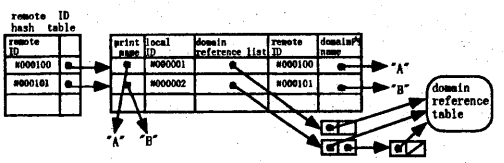


図 4. Local Reference Table

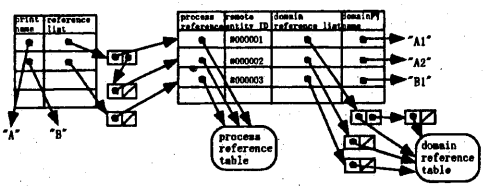


図 5. Remote Reference Table

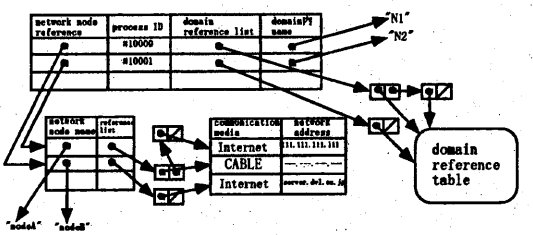


図 6. Process Reference Table

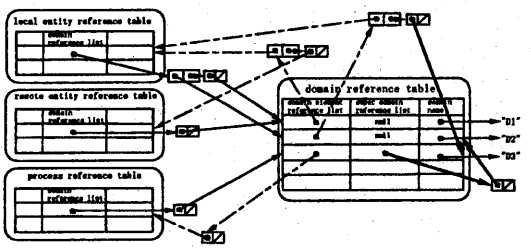


図 7. Domain Reference Table

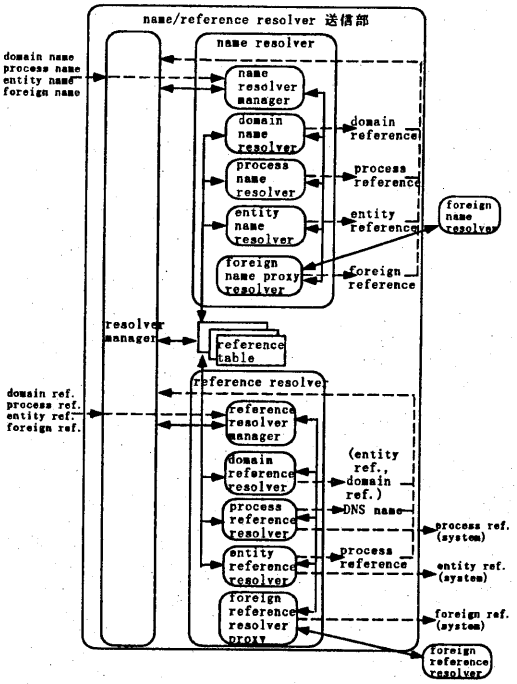
用する.

3.3.参照と名前の解決

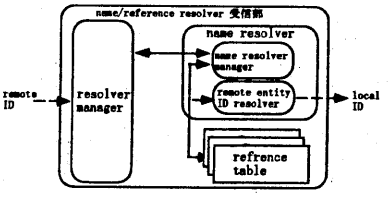
名前・参照解決器は，メッセージ送信時に送信先オブジェクトの所在を特定するために呼び出される送信部と，メッセージ受信時に受信オブジェクトを特定するために呼び出される受信部からなる(図 8).

送信部では，resolver manager,name resolver manager,refrence resolver manager が連携して，参照/名前の種類に応じ適切な resolver を選択し解決する．解決結果にさらに解決の必要がある要素が含まれる場合は，再び resolver manager に解決要求を出す.

受信部は，受信したリモート識別子をローカル識別子に変換し，プロセス内の並行オブジェクトを特定する.



(a)送信部



(b)受信部

図 8.名前・参照解決器

3.4.配置と初期化の方式

プログラムモジュールの分散配置は，プラットフォームシステムが持つ Allocator により行う. Allocator は，並行オブジェクトとして実装されシステムプロセス内に存在する.

Allocator は，アプリケーション実行時に与えられる空間構成情報に従い，各ネットワークノード上に予め配備した管理プロセスと連携して，アプリケーションを配置し起動する. (図 9) 各管理プロセスは，Allocator の指示に従いプログラムモジュールの転送とプロセスの生成を行う. プロセス識別子は，各管理プロセスがそれぞれのネットワークノード上で一意に割り当る.

Allocator は，プロセス生成とそこでの参照の輸出手続きの終了後，各プロセスに参照の輸入手続きを要求する. すべての相互参照が確立した時点で Allocator は，アプリケーションの起動を行うべきプロセスに対して計算の開始を指示する.

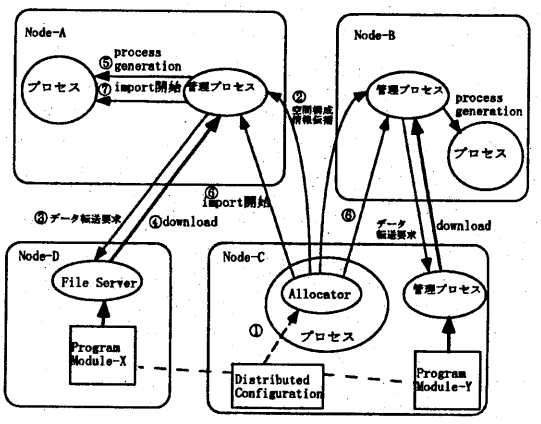


図 9. Allocator と管理プロセス

4.まとめ

本稿では，ネットワーク環境上で動作する分散アプリケーションとその計算オブジェクトを動的且つ柔軟な形で管理できる空間管理方式について述べた．アプリケーション空間は，動作時のネットワーク環境上に対応させて，柔軟に生成・配置することが可能であり，オブジェクト間の相互参照は効率の良い解決が可能な形式で表現し管理している．従ってより実行効率の良いオブジェクト間通信が実現される．

本方式ではまた，アプリケーションのそれぞれを個別の空間を形成して管理しており，同一のネットワーク環境に複数のアプリケーションを展開して運用することが可能である．

本管理方式は，我々が開発している並行計算プラットフォームにおいて実装中であり，今後その優位性を検証する予定である．我々はまた，ネットワークメディエーションと呼ぶ空間探索と接続のため機能を提案しており[14]，これを用いてさらに，異なるアプリケーションに亙った計算実行のための機能を実現していく予定である．

謝辞

本研究の遂行にあたって多大なる助言を頂いた五十嵐達治部長を始め当研究所諸氏に，感謝いたします．

参考文献

[1] Jon Siegel, "Common Object Services Spcification", vol.1, OMG-94-1-1, Mar, 1994.

[2] 小中 他, "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," 情処論文誌, vol. 36, no. 7, pp. 1520-1528, Jul. 1995.

[3] 塚本 他, "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," 情処論文誌, vol. 37, no. 5, pp. 853-864, May 1996.

[4] B. Achauer, "The DOWL Distributed Object-Oriented Language," Comm. ACM, vol. 36, no. 9, pp. 48-55, Sep. 1993.

[5] R. Chandra, et. al., "COOL: An Object-Based Language for Parallel Programming," IEEE Computer, vol. 27, no. 8, pp. 13-26, Aug. 1994.

[6] A. Yonezawa, et. al., "Implementing Concurrent Object-Oriented Languages on Multicomputers," IEEE Parallel and Distributed Technology, vol. 1, no. 2, pp. 49-61, May 1993.

[7] E. Yourdon, "Java, the Web, and Software Development," IEEE Computer, vol. 29, no. 8, pp. 25-30, Aug. 1996.

[8] Kramer, The Java Platform: A White Paper, Mountain View, USA: Sun Microsystems, 1995.

[9] 白鳥 他, "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—," 情処研究会報告 DPS62-11, pp. 79-86, Sep. 1993.

[10] 富永, "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, Apr. 1994.

[11] 萩原 他, "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—," 情処第 52 回全国大会講演論文集, vol. 3, pp. 223-224, Mar. 1996.

[12] 萩原 他, "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 第 6 回メディア統合技術研究会 MT-6-S1-2,1996.

[13] 前川, "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—," 情処第 52 回全国大会講演論文集, vol. 3, pp. 225-226, Mar. 1996.

[14] 前川 他,"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,Oct.1996