マルチメディア通信と分散処理 66−17
（１９９４．７．８）

# 自律協調分散システム開発環境 Noah の通信機構

佐藤 文明　小塚 宏　宮崎 一哉　福岡 久雄

三菱電機 (株) 情報システム研究所

Noah(Network oriented applications harmony) は、場を介した通信機構上に協調プロトコル及び、アプリケーションとシステムの動作状態を監視／制御する機構を提供することによって、分散処理環境上で複数のアプリケーション間の協調動作を支援するためのソフトウェア・プラットフォームである。本稿では、Noah 環境のコンポーネントの一つである通信機構について論じる。Noah の通信機構は、階層的なタプルスペースに基づくアプリケーション間の通信を提供しており、アプリケーションの自律的で選択的な通信、同期／非同期通信、階層的なアプリケーション管理などのベースになっている。

# Communication Mechanism of Noah – An Environment for Autonomous and Cooperative Distributed Systems

Fumiaki SATO, Hiroshi KOZUKA, Kazuya MIYAZAKI, Hisao FUKUOKA

Computer & Information Systems Laboratory
Mitsubishi Electric Corporation
5-1-1, Ofuna, Kamakura, Kanagawa 247, Japan

Noah (Network oriented applications harmony) is a software platform to support distributed applications cooperation, and provides communication mechanisms based on cooperation field, cooperation protocols, agents and system management / control procedures. This paper describes communication mechanism which is one of the main compornents of the Noah environment. The mechanism provides communication among applications based on a hierarchical tuple space and this is the base of the autonomous communication of the applications, synchronous/asynchronous communication, and hierarchical management of the applications.

## 1 はじめに

コンピュータのハードウェアのコストパフォーマンスの向上によって、複数の高性能なコンピュータをネットワーク接続した分散環境が出現した。このような環境では、最適なサービスを高い信頼性で効率的に利用できる潜在的な能力が期待できる。しかし、これらの機能を利用するための分散型のソフトウェア開発は複雑さを極め、非常に専門的で難しくなっている。また、多くの場合、柔軟さに欠け、信頼性も乏しいシステムであることが多い。それは、分散環境において柔軟で信頼性の高いシステムを開発することが複雑で難しい作業であることに起因している。

Noah (Network oriented applications harmony) では、システムは自律性を持った複数のエージェントによって構築される。このエージェント自体は、システム全体から見れば非常に単純な作業を行なうモジュールであり、互いに協調を行なうことによって一つの機能を提供する。ユーザは、このエージェントの機能とエージェント間の協調手順を規定することでシステムを構築する。このようなシステム開発では、ソフトウェアのモジュール性が高まり、既存のソフトウェアを組み合わせたり、新たなモジュールを追加したりという、「ソフトウェア部品」の考え方を適用しやすくなることで、開発効率や信頼性を高めることができる。Noah では、これらのエージェントが互いに通信を行なうための通信機構、協調を行なうための協調機構、そしてエージェントの制御を行なうためのアプリケーション制御機構から構成される Noah 環境を提供する。

Noah 環境の特徴としては、エージェントが協調を行なう論理的な“場”を提供することによって、協調する相手を明示的に示すことを行なわない。それによって、エージェントの自律性を高めることが可能となる。また、それらの協調の場を階層的に定義することによって、階層的な問題解決に対応することができる。この階層は、協調の場に定義される協調手順の継承を可能とし、エージェント間の協調を柔軟に指定することを可能としている。

本報告では、第 2 章で Noah における基本概念を説明した後、 第 3 章において場を介した通信機構「Noah 通信機構」の機能について述べる。第 4 章では、通信機構の実現メカニズムについて述べる。第 5 章では、通信機構における基本性能の評価結果を述べる。第 6 章はまとめである。

## 2 Noah 概要

ここでは、 Noah の基本コンセプトについて述べる。 Noah は、分散環境におけるアプリケーション間の協調機構を提供するソフトウェアプラットフォームである。

### 2.1 コンセプトモデル：湖 / 運河モデル

Noah の計算アーキテクチャの基本モデルは、図 1に示す湖 / 運河モデルに基づいている [1]。

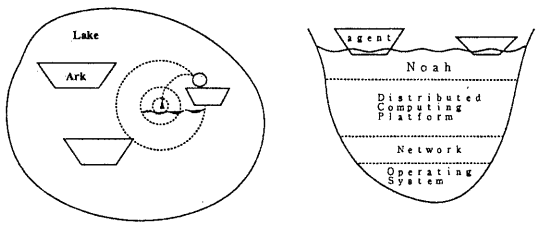


図 1: Noah 環境とその断面

湖 / 運河モデルは以下の抽象実体からなる。

1. 湖：湖はエージェント間での通信のための場である。同じアーキテクチャのコンピュータの集合は湖を構成する。

2. 運河：運河は 2 つの湖を接続するゲートウェイである。異なるアーキテクチャの湖を接続するためにデータやメッセージの変換を実行する。

3. 箱船：箱船はアプリケーションの機能単位であり、湖の上に存在するとみなされる。これは能動的なエージェントであり、湖の状態の変化によって処理を行ない、その結果を湖に返すことによって湖の環境を新しい状態に遷移させる。

### 2.2 Noah の全体アーキテクチャ

上記の特徴を実現するための、 Noah のアーキテクチャの全体構成は、図 2で示される。

ここで、以下の (1) ～ (3) が Noah 環境で実現する部分である。

(1) Noah 通信

アプリケーション間の同期、競合、データ共有、通信機能を提供する。

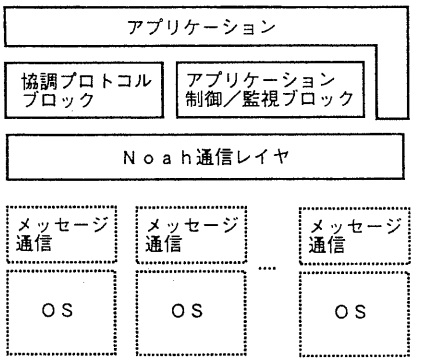


図 2: Noah の実現アーキテクチャ

**(2)** 協調プロトコル

アプリケーション間の様々な協調作業に必要な、共通の機能を提供する。

**(3)** アプリケーションの監視と制御

アプリケーションの協調処理のための制御を行なう。制御とは、アプリケーションの起動、停止、複製管理、属性値変更などである。

### 2.3 Noah 通信レイヤの条件

Noah では、エージェントの自律的な動作を実現するために、以下の項目を実現する必要がある。

- 個々のエージェントが判断基準を持つ
- 判断基準に基づき、周りの状況を判断する
- 状況判断に基づき、自己の動作を決定する
- 周りのエージェントとネゴシエーションを行なう
- 周りのエージェントと競合する

このようなエージェントが分散環境で信頼性を持って動作するために、Noah 通信機構に対して以下のような機能を持たせた。

- 位置透過に通信する機能
- 障害透過に通信する機能
- 同期的な通信と非同期的な通信の機能
- 自己の都合に応じてメッセージを選択する機能
- 同報通信の機能

## 3 Noah 通信機構

### 3.1 計算モデル

Noah では計算の実行主体 “Ark(箱船)” と通信媒体としてのタプルスペース [2] である “Lake(湖)” を組み合わせた “LAKE 計算モデル” を採用している（図 3参照）。アプリケーションは、複数の LAKE から構成される。LAKE は更に他の LAKE を生成したり、生成した LAKE を削除することができる。実行主体である Ark は、LAKE-ID を引数として、タプル（メッセージ）をタプルスペース（Lake: メッセージプール）に対して入出力する。LAKE 計算モデルの特徴は、タプルスペースを使ったバラエティに富んだ通信方式の実現である。タプルスペースを使った通信では、タプルスペースを黒板モデルとして利用した場合の同報通信、1 対 1 通信で利用した場合の同期通信、非同期通信が可能である。LAKE では、複数のタプルスペースが存在するため、アプリケーション毎にタプルスペースを割り当てたり、通信プロトコル毎にタプルスペースを割り当てることができる。これは、タプルの衝突を防ぐ意味で、有効である。また、タプルスペースを使った、アプリケーションのグルーピングの概念を導入することができる。これは、アプリケーションの階層的な管理をする際に、「あるタプルスペースをアクセスするグループ」ということで、アプリケーションを抽象的な単位にまとめることを意味する。

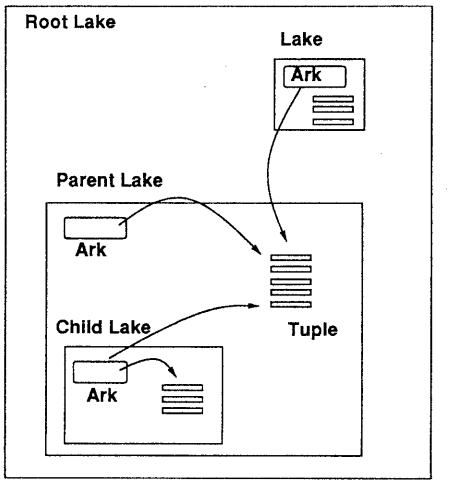


図 3: LAKE 計算モデルのコンセプト図

タプルスペース通信とは、共通のタプルスペース

に各アプリケーションが能動的にタプルを置き、あるいは取り出すことにより、アプリケーション間の通信を行なうものである。この通信方式では、RPCやオブジェクト指向のメッセージパッシングと異なり、通信相手を指定しない。そのため、通常の1対1通信以外に、ブロードキャスト、マルチキャスト、非同期通信、同期通信などを容易に実現できる。つまり、従来は通信相手を特定して、そこに処理を依頼するため障害や過負荷などで応答がないという事態が起こった。それに対してこの通信方式では、あるサービスを提供するタプルスペースを相手に通信することで、幾つかのサービス可能なサーバのうちの最適なサーバが応答するシステムを実現することが容易になる。

Noahのコンセプトであるアプリケーション間の協調においては、アプリケーションを陽に指定して協調を行なうのではなく、タプルにマッチングするアプリケーションが自律的に協調作業に参加することから、この通信方式が適している。

LAKE計算モデルのもう一つの特徴は、親Lakeが子Lakeを生成する場合、子Lakeは親Lakeとタプルスペースを共有することができる点である。この特徴は、親の通信が、子のLakeに対して暗黙のうちに影響を与えることができる。つまり、外部のアプリケーションが親のLakeを経由して、子のアプリケーションを制御したり情報を与えたりすることができることが特徴となる。これは、例えばある親Lakeを共通に持つ幾つかの子Lakeがあった場合、その子Lakeに対するデフォルトの動作や、共通に使われるマシン情報などが親Lakeを使って配布することが可能となる。

### 3.2 関連研究

(1) Cellula計算モデル [3]

Cellula計算モデルは、計算の実行主体であるプロセスとタプルスペースを組み合わせた“セル”をモジュールの単位としてアプリケーションを構築するものである。Cellulaでは、アプリケーション内の親子のCellは、独立なタプルスペースを持っている。そのため、アプリケーション間の密な結合を行なう場合、外部に対し親が子のCellを公開する必要がある。親が子のCellの公開を行なわない場合、外部のアプリケーションは、親のCellのみが通信の対象となり、子のCellに対して密な通信は不可能となる。

一方、LAKE計算モデルでは、子のLakeが親のLakeとタプルスペースを共有することから、陽に親が子のLakeを公開する必要はない。すなわち、外部のアプリケーションから親Lakeへの通信が、子のLakeにも暗黙のうちに影響を与えるからである。

(2) Kemari計算モデル [4]

Kemari計算モデルは、計算の実行主体であるプロセスと通信機構であるメッセージプールから構成されている。このプロセスには、内部状態、機能記述部、協調記述部が存在しており、協調の定義が行なえる。

この計算モデルでは、協調プロトコルが計算モデルの内部に入り込んでいる。従って、我々の協調プロトコルをアプリケーションの外部に置こうとする考え方と正反対であり、ユーザが複雑な協調手順をアプリケーション毎に書く必要が生じる。また、計算モデルに多くの内容が盛り込まれており、逆にシンプルさが損なわれている。

## 4 実現方式

### 4.1 分散環境でのLakeの実現

Noah通信機構は、タプルスペース管理を行なうタプルスペースマネージャと、タプルスペースマネージャをアプリケーション(タプルスペースクライアント、あるいはLake)からアクセスするためのアクセスライブラリから構成される。

タプルスペースアクセスライブラリのレベルからは、全てのLakeが位置透過である。

図4に、タプルスペースマネージャとアプリケーション（クライアント）の関係を示す。

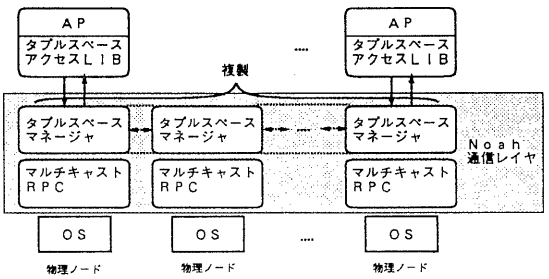


図4: タプルスペースマネージャとクライアント

タプルスペースマネージャは、各物理ノード上に1プロセスとして動作している。各タプルスペースマネージャは、LAKE計算モデルに基づいて、タプルスペースであるLakeとそれに属するアプリケーションプロセスの動作状況を管理している。全てのタプルスペースマネージャは、同じ情報を複製として持っており、どのノードが障害でダウンしても、残りのタプルスペースマネージャには影響がないようにしている。

各タプルスペースマネージャに対するアクセスは、マルチキャストＲＰＣを用いており[5]、複製間で一貫性のとれた情報更新が行なわれるようになっている。また、ノードのダウンや新たなノードの追加による複製の数の減少や増加についても、一貫性を保つことができるプロトコルを採用している（図5）。

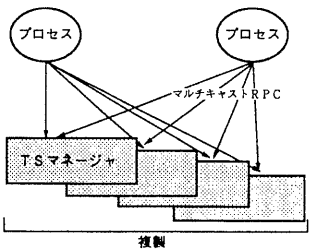


図 5: マルチキャストによるタプルスペースの一貫性管理

## 4.2 階層的なLakeの名前付け

Lakeは、タプルスペースマネージャのクライアントとして実現され、ある限定されたタプルスペース領域(タプルの管理テーブルの一部)を占有するプロセスとなる。Lakeはそれぞれtree形式の名前を持っており、起動されるとその名前をタプルスペースマネージャに登録する。タプルスペースマネージャは、登録された名前に応じて、別々のタプル管理テーブルを作成し、その管理テーブルに応じたIDを返答する（図6）。以後、このIDを引数として、クライアントからタプルの入出力要求が行なわれる。タプルスペースマネージャでは、このIDをkeyとして検索すべきタプル管理テーブルを識別し、タプルの入出力処理を行なう。

タプルの名前は、“/ABC/DE/FGH”というようなtree形式で登録される。あるLakeが“/example”という名前を持っており、別のLakeを自分の子Lakeとして作る場合、“/example/test1”のような名前を使う。また、デフォルトで“test1”を使えば、“/example/test1”となる。

タプルの入出力は、取り出し(IN)、読み込み(RD)については、指定されたLakeに対応するタプルスペース（タプル管理テーブル）から、より上位のタプルスペース（タプル管理テーブル）までを順番に検索する。書き込み(OUT)については、指定されたLakeに書き込む（図7）。

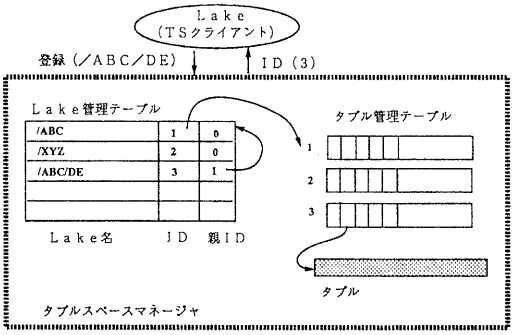


図 6: Lakeの管理機構

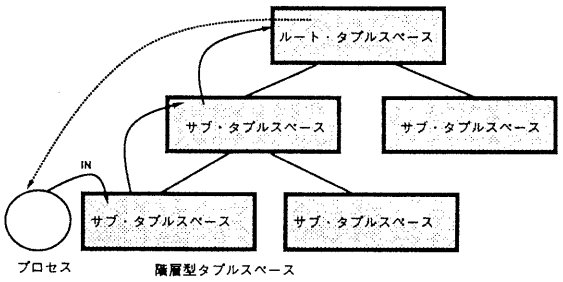


図 7: タプルスペースの階層

## 4.3 通信機構の各種インタフェース

通信機構では、Lakeを作成したり管理したり他のLakeと通信するために多くの操作インタフェースを提供する。

管理機構としては、Lakeの作成、削除、Lakeの名前からの検索、Lakeの作成状況やタプル及びブロックされている操作の状況をモニタする操作インタフェースが存在する。Lakeは、通常のタプルを扱うためのものと、ストリーム的な動作するLakeとに分けられており、その種類はLakeを作成する際に指定することができる。Lakeは、階層的な名

前付けが行なわれており、サービスの階層やアプリケーションのグルーピングに利用する。

また、タプル入出力では、タプルの投入、タプルの取り出し、タプルの読み出しの他、タプルの削除（応答なし）、タプルＩＤを指定した個別のタプル操作が行なえる。また、タプルの入力に関しては、ブロック型のアクセス、ノンブロック型のアクセス、時間指定型のアクセスが可能である。タプルの出力においては、タプルの有効期限を設定することで、タプルの有効期限が切れた場合は、タプルは自動的に消滅させることができる。

## 5 基本性能評価

図8に、Noah 通信機構の基本性能を測定した結果を示す。ベースとなる RPC のおよそ２～３分の１の性能になっている。これらの性能のネックとなっているのは、異なるタプルスペースにアクセスする場合でも、すべての処理がシーケンシャルに実行されていることと、アクセスライブラリとアプリケーション間でのデータのコピーによるオーバヘッドが考えられる。従って、通信性能の向上には、異なるタプルスペースアクセスを内部的に並行に実行できるようにするか、あるいはタプルスペースの複製を全ノードではなく、いくつかのノードに限定して置くことでタプルスペースマネージャの並行動作を可能にすることが考えられる。また、コピーについては、メッセージをポインタで渡すなどでコピーを減らすなどのチューニングが必要である。

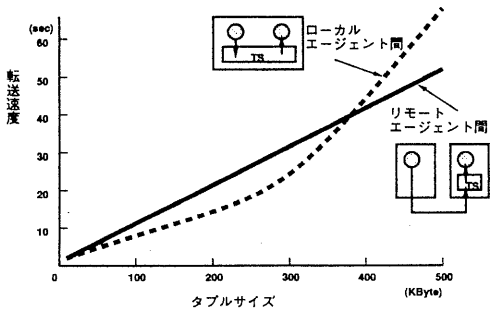


図 8: 通信機構の基本性能

## 6 おわりに

Noah は、分散環境においてアプリケーションを自律的に動作させ、それらを協調させて柔軟なサービスを提供するとともに、拡張性や信頼性を向上させる。本論文では、Noah のアプリケーション間通信を提供する Noah 通信機構について論じた。Noah のアプリケーションを自律的に動作させるためには、通信が能動的に行なわれることが必要である点から、分散環境にタプルスペース通信を実現している。また、サービスの階層的な管理を目的として、階層型のタプルスペースを実現している。

Noah 環境では、この通信機構のうえに協調プロトコル機構 [6]、アプリケーション監視 / 制御機構 [7] を備えており、これらを利用して、いくつかのサンプルアプリケーションを作成した。その結果、従来のアプリケーションを容易に連係させ、信頼性と拡張性の向上したシステムとすることが可能となった。

Noah 通信機構に関する今後の課題としては、通信性能の向上、障害回復の速度向上、また標準的な通信機構への移行などが残っている。

## 参考文献


[1] 小塚，佐藤，宮崎，福岡:“分散協調環境 Noah (Network oriented applications harmony) の提案”，情報処理学会研究報告 92-DPS-59-11 (1993).

[2] Sudhir Ahuja, Nicholas Carriero, and David Gelernter: “Linda and Friends,” IEEE COMPUTER, Vol.19, No.8, Aug.1986.

[3] 吉田, 楢崎:“場と一体化したプロセスの概念に基づく並列協調処理モデル Cellula”, 情報処理学会論文誌, Vol.31, No.7, pp.1071-1079 (1990).

[4] 矢野，武宮，布川，野口:“自律的な協調を行なう分権型計算モデル Kemari”，情報処理学会研究報告 90-SF-37-18 (1990).

[5] Kenneth Birman: “The Process Group Approach to Reliable Distributed Computing”, Cornell University TR91-1216(1991).

[6] 宮崎，小塚，佐藤，福岡:“自律協調分散システム Noah における協調機構”，情報処理学会第４８回全国大会６Ｆ－８ (1994).

[7] 小塚，佐藤，宮崎，福岡:“自律協調分散システム Noah におけるエージェントの制御機構”, 情報処理学会第４８回全国大会６Ｆ－６ (1994).