マルチメディア通信と分散処理　69－1
（１９９５．　３．　２）

# 自律分散指向のスケジュール管理システムの開発

佐々木健一　明星秀一　田中幹夫　後藤浩一　松原　広
鉄道総合技術研究所

データの分散保有に注目した簡単な自律分散の仕組みを考案し，その実現に必要なシステムの構成及び動作フローを定義した．さらに実際にこの仕組みの上で動作するアプリケーションとして，スケジュール管理システムを考え，そのモジュール構成，プロトコル，シーケンス等を確定し作成を行った．最後に，そのスケジュール管理アプリケーションの性能を評価し，問題点についての改善案を講じ，データ分散保有をベースとした自律分散システムにおける今後の課題について論じた．

# A DEVELOPMENT OF AUTONOMAS DISTRIBUTED ORIENTED SCHDULE MANAGEMENT SYSTEM

Kenichi SASAKI Shuichi MYOJO Mikio TANAKA Koichi GOTO Hiroshi MATSUBARA
Railway Technical Research Institute
2-8-38, Hikari-cho, Kokubunji-si, Tokyo 185, Japan

We think a simple autonomas distibuted oriented system which pay attention to distributed data structure, and defined its system structure and application's action flow. In addition, we thought of schdule management system as an application which based on this system, and confirmed its module structure, protocols and flow sequences, after we have made it. In conclution, we measured performance of the schdule management system, considered improvement of the system, and discussed remaind problems of autonomas distibuted oriented system basing on distributed data structure.

# 1 はじめに

近年パソコンやワークステーションなどの価格の低下により、オフィスにもこれらのコンピュータ機器が進出して来ており，その運用も，旧来の汎用機と端末の関係から，巨大な資源を有したサーバマシンと，パソコンやワークステーションに代表されるクライアントマシンで構成されたクライアント・サーバシステムが一般的になりつつある．そういった中今後の動向として，それがさらに押し進められ，システムの自律分散環境が実現することが考えられる．

本稿では，上記の状況を踏まえた上で，一般的なパソコンのみを用いて簡単な自律分散環境を実現したシステムのプロトタイプを作成し，その評価をするものである．

# 2 データの分散保有による自律分散システム

## 2.1 クライアント・サーバ方式の問題点

クライアント・サーバ方式で強調されたのは負荷の分散である．しかし現行のクライアント・サーバシステムは，従来の大型機を同等の性能を有するサーバに，従来の端末機をパソコンに置き換えただけのシステムが多い．

更にデータの検索やデータベースの更新などのタスクは依然として高価なサーバマシンが担当しており，サーバマシンがシステムダウンを起こせば，システム全体がダウンしてしまう事態は改善されていない．またサーバが妨害者などから侵入を受けた場合などの，セキュリティ上の問題も解決されない．

## 2.2 分散保有時の特性

分散環境を実現する簡単な方法として，サーバに蓄積されたデータを各端末に分散して保有し，必要があればその都度ネットワークを介してデータをやり取りする方法がある．この方法には，

1. サーバマシンの変調等によるシステム異常がなく，システムの安定性が高い．
2. ネットワークの孤立化 (isolation)，局地化 (localization) 等により，他のマシンとの通信手段を切断されても，ある程度の業務遂行が可能である．
3. データが分散しているので，妨害者にシステムへ侵入された場合の明確な標的が確定し難い．よってデータの盗難に遭遇した場合でも，重大な損失を招く可能性はクライアント・サーバ方式と比較し低く押えられる．

といった長所があり，上記の問題はかなり改善される．しかし一方で，

1. データの分散配置法により，システムの性能が変化する．
2. いわゆる統計的なデータがすぐには取り出しにくい．
3. 統一的なデータの整合性が必ずしも保証されない．

といったさまざまな欠点が派生する．

上記諸欠点について順番にみてゆく．まず分散データベースにおいて，最適なデータ配置を得るための研究は，主としてファイルの最適な分散配置を求める問題に置き換えられた上で，以前から研究されてきている [1]．従前よりのそういった研究ではシステムの効率が重視されているのだが，本稿では先述したように障害発生時の状況を考慮し，そうした数学的手法を用いて提示された配置ではなく，システムの可用性や単純性を重視した分散配置を考えるものとする．

次に統計的データに関しては，業務における通信を私信（一対一），広報（一対多），会議（多対多）に分類する [2] と，会議型の業務では統計的データは即応を必要とする場合がないことが多く，しかも参加者以外に情報を知らせる必要も特にないので，データを分散しても影響が出ないだろうことが推測される．

なおデータの整合性については後述する．

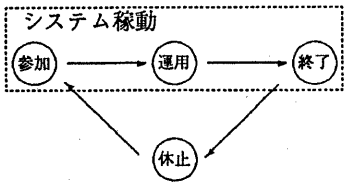


図 1: フェーズの状態遷移

### 2.3 システム動作

データの分散保有による自律分散システムは，システムの中核とも言うべきサーバを所有しないシステム構成を採っているので，システム起動直後にデータをネットワークより収集する必要がある．

またシステム稼動中はデータの不整合がないように，他のシステム稼動中のノードと連携を取る必要がある．

最後に，システムを終了する際には，他のシステム稼動中のノードにその旨を連絡する義務を負う．

こういった意味で，データの分散保有による自律分散システムは，いわゆるクライアント・サーバ方式とは異なったシステムの動作管理が必要であり，システム稼動中の過程は，大きく三つのフェーズに分けられる．

**参加フェーズ** 任意のノードが，システムを起動するのに必要な手続きを行うフェーズ．

**運用フェーズ** ユーザの意志に従ってデータの交換及び編集を行うフェーズ．

**終了フェーズ** 任意のノードが，システムを終了するのに必要な手続きを行うフェーズ．

さらに，各フェーズ間の関係を図 1 へ示す．

### 2.4 システム構成

データの分散保有を基本とする自律分散システムの基本構成を図 2 へ示す．以下に，各モジュールの働きと，ファイルの役割について解説する．

**IF マネージャ** データの入出力など，ユーザとのインターフェイス部分を管理する．

**エージェントコントローラ** 各マネージャを管理し，必要があればイベントを発生させる．

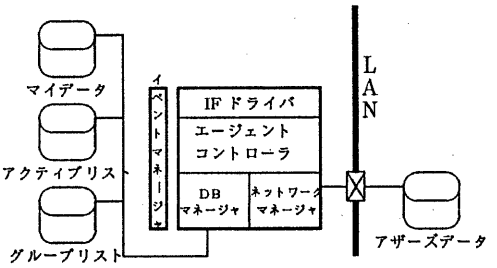


図 2: モジュール構成図

**ネットワークマネージャ** イベントマネージャと連携しながら，ネットワークの管理する．

**イベントマネージャ** ネットワークマネージャから受けたメッセージをイベントに翻訳し，該当モジュールへメッセージにして通知する．

**DB マネージャ** 各種データベースの管理業務を行う．データベースには，システムが稼動するのに必要とされる情報が格納されており，以下の各種がある．

- **グループリスト** ユーザが必要とするノードを登録した静的なデータベース．
- **アクティブリスト** グループリストに記載されたノードのうち，現在稼動中のノードを登録した動的なデータベース．
- **マイデータ** 自ノードで管理するデータの各項目を登録したデータベース．
- **アザーズデータ** 他ユーザのデータの各項目を登録したデータベース．各ノードで管理すると同時に，システム稼動中は，他システムよりの干渉により動的に変更される．

## 3 スケジュール管理アプリケーション

### 3.1 予定表

本節で述べるスケジュール管理アプリケーションは特殊なものではなく，図 6 に示すような，一般には予定表，あるいはホワイトボードなどと呼ばれているもので，その内容は，例えて言えば「誰々

さんは，何日の何時から何処へ出張」といった事柄を書き留めておくスプレッドシート風の表である．

### 3.2 システム概観

本稿では，このスケジュール管理アプリケーションをデータの分散保有をベースとした自律分散型のシステムとした．このシステムは複数台の計算機 (PC) 上で稼動し，各計算機上で稼動するシステムは，ネットワークで接続された他の計算機上で稼動する，他のシステムと協調しながら，統合的に一つのシステム，すなわちここではスケジュール管理アプリケーションを形成する．

### 3.3 データ配置

現在頻繁に論議されるデータの分散配置法の一つに回覧版方式 [3] がある．しかし一般的なクライアント・サーバシステムにおいて，システムが使用不能になる状況を考慮すると，

1. サーバマシンにかかる負荷が過大になり，タスク処理が極端に遅くなった，又は停止した．
2. 孤立化・局地化等の理由によりネットワークが正常に機能しなくなった．

の 2 要因が主なものとして考えられる．回覧版方式では，1 の場合障害の分散化は可能である．しかし，ネットワークは常に正常に動作するという前提の基にデータの分散が図られているので，2 の場合ユーザの要求を満たすことはできない．そこで，

1. 各計算機に「所有者」を設定する．
2. 「所有者」に関するデータは，その「所有者」の所有する計算機に格納する．このデータを「オリジナルデータ」と呼ぶ．
3. 所有者とは関係のないデータであっても，ある所有者がそのデータを必要とする場合，そのデータはそのデータの所有者の所有する計算機と，そのデータを必要とする所有者の計算機の両方へ格納される．この場合オリジナルデータから複写され別の計算機に格納されたデータをコピーデータと呼ぶ．
4. 上記の場合，でかつ両データの内容に違いがある場合は，オリジナルデータへ記された内容が優先される．
5. データの修正は，オリジナルデータのみが，「所有者」の所有する計算機上で可能である．コピーデータはあくまで参照できるだけであり，修正はできない．

といった原則を設定することにより，ユーザからの要求に対応することとした．

### 3.4 データ構造とその整合

データの分散所有を前提とするとき，問題となるのは先述したがデータの整合性である．スケジュール管理アプリケーションではデータの整合性を確実なものにするため，分散したデータには ID と最終更新時刻をタイムスタンプとして付加し，同じ ID のデータを受信した場合はタイムスタンプの新しいものを優先するという原則を設定した．この場合，タイムスタンプはオリジナルデータの「所有者」の計算機が常に付加するので，システム全体としてのタイムスタンプの同一性は，原則として必要ない．

また，システムは起動後，同じシステムの稼動している全ての計算機へ自ノードのオリジナルデータを送付し，さらに自ノードの所有するコピーデータのタイムスタンプを表にして，相手ノードのそれと交換することにより，データの整合性を保つ．

### 3.5 セッション

計算機上で稼動する各システム間の協調はネットワークを介してメッセージで行われ，メッセージの通知側システムと被通知側システムの間には，メッセージの内容に基づき，データの交換などのセッションが行われる．

スケジュール管理アプリケーションで行われるセッションには，参加通知，更新通知，転送要求，終了通知の 4 種が存在する．

### 3.5.1 参加通知

参加通知セッションでは，データ転送を，基本的に以下の手順で行うものとする．

1. 通知側ノードから被通知側ノードへ，参加通知メッセージを発信（メッセージ中にノード名を含める）．
2. 通知側ノードから被通知側ノードへ，オリジナルデータを順次転送．
3. 通知側ノードから被通知側ノードへ，コピーデータリストの転送を要求．
4. 被通知側ノードより通知側ノードへ，データ転送メッセージにてコピーデータリストを転送．

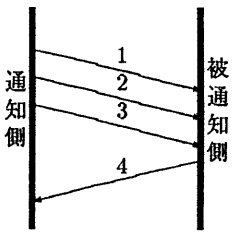


図 3: メッセージ交換の手順（参加通知）

上記の通信手順を一括すると，通知側から被通知側への往信と被通知側から通知側への返信によって処理可能であるとわかる．手順簡略化のため，実際の通信は一往復で行うものとする．

### 3.5.2 更新通知

通知側ノードから被通知側ノードへ変更データ内容などを記したメッセージを通知する．

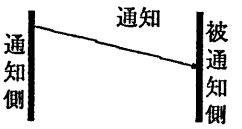


図 4: メッセージ交換の手順（更新・終了通知）

### 3.5.3 転送要求

要求を記した通知側ノードから被通知側ノードへの往信と，データを記した被通知側ノードから通知側ノードへの返信から構成される．

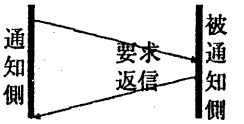


図 5: メッセージ交換の手順（転送要求）

### 3.5.4 終了通知

セッションを終了する旨を意味するコードを記したメッセージを被通知側へ送信する．

## 4 性能評価および考察

### 4.1 プロトタイプの評価

上記の仕様に従い，自律分散型スケジュール管理アプリケーションのプロトタイプを試作した．使用したハードウェア及びソフトウェアの環境を表 1 へ示す．

表 1: プロトタイプ開発に使用した環境

| | |
|---|---|
| ハードウェア | NEC PC-9801 及び PC-H98 シリーズ機 |
| OS | Microsoft Windows 3.1 |
| ネットワークプロトコル | TCP/IP |
| 開発環境 | Microsoft C Ver 7.0 Microsoft Windows SDK |
| ネットワーク開発環境 | BSD ソケット |

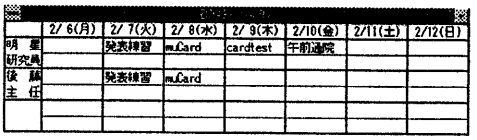


図 6: スケジュール管理アプリケーション表示画面

さらに，作成した自律分散型スケジュール管理アプリケーションの表示画面を図 6 へ示す．

### 4.2 現在の問題点とその改善案

上記のプロトタイプを試験運用をし，判明した問題点を以下に列挙する．

#### 4.2.1 コピーデータ転送の問題

このシステムでは，システム立上り時に各マシンがお互いにコピーデータリストを交換しあって，その後そのリストに基づき最新のデータをお互いに交換しあうが，データ本体の交換は立上り時ではなく，その後システムがユーザからの要求を受けてない時間を見計らって行われる．

しかし相手側の計算機がデータを渡せる状態かどうかは感知できないので，要求側は何度かリトライを行うが，最終的にデータを渡せるかどうかは保証されていない．

もっともこれは非プリエンプティブなＯＳを使用したために起こる現象であるから，マルチタスク可能なＯＳを用いることにより，速やかに解決するものと思われる．

#### 4.2.2 システム立上り時間の問題

このシステムでは，システム起動時の参加通知メッセージを送信するときに，グループリストに属する他の全てマシンとコネクションを張る．従って，

| | | | |
|---|---|---|---|
| $T_{out}$ | コネクションタイムアウト時間 | $T$ | 立上り時間 |
| | | $n$ | 非運用者数 |
| $\bar{t}$ | コネクション成立に必要な時間の平均値 | $N$ | 全ユーザ数 |

とすると，システムの立上り時間は，

$$T = (N-n)\bar{t} + nT_{out} \qquad (1)$$

のように定式化される．ここで $\bar{t} \ll T_{out}$ から，$(N-n)\bar{t}$ の項は無視でき，

$$T = nT_{out} \qquad (2)$$

となり，立上り時間は非運用者数に比例し，システムのパフォーマンスはユーザ数に依存することがわかる．また，それゆえに大規模システムには向かない．

簡単な解決策として参加通知セッションをコネクションレス型とし，ブロードキャストを用いるなどの方法があるが，データ転送の信頼性を優先したため，今回は実現していない．

#### 4.2.3 運用上の問題

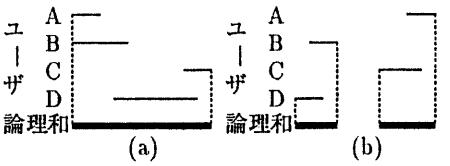


図 7: 運用上の問題の一例

このシステムは，全ユーザの計算機のうち，常に最低一台は稼動していることを期待している．ここで図 7 に A から D までの 4 人のユーザの使用時間を並べ，その論理和を横軸に取ったとき，図 7-(a) のような場合は問題ない．しかし図 7-(b) の場合，ユーザ A と C の最新データは B と D に伝わることはなく，またその逆もない．現実の運用において，なるべく (b) のような事態が発生しないよう，グループを考慮するのが望ましい．

### 4.3 今後の課題

現在システム構成を簡便にするため，データ所有者のマシン上において所有者のデータのみ編集を可能としており，他人のデータは参照できるだけで編集は不可能である．同様に，他人のマシン上ではそのマシンの所有者のデータのみで，ユーザが変わってもその人のデータは参照はできても編集はできない．これは不便なので，認証プロトコル等を設定して解決する方法を検討している．

## 参考文献


[1] 重田和弘・高野　誠・斎藤　勲「通信システムにおける分散データ配置方式」，マルチメディアと分散処理 45-13, (1990.5.25)

[2] 小塚　宏・辻順一郎・坂下善彦「分散環境における業務情報交換の試み」，マルチメディアと分散処理 47-10, (1990.9.20)

[3] 北村泰彦・小川　均・北橋忠宏「分散型問題解決における問題割当てのための一通信方式」，電子情報通信学会論文誌D J71-D-2, No.2, 439-447, (1988.2)