3E-7　分散型通信ノードシステムソフトウェアにおける透明性実現手法

村上龍郎　　荒木伸夫

ＮＴＴ交換システム研究所

1．まえがき

通信ノードのシステム構成は分散化される傾向にあり，この状況は制御ソフトウェアの開発，維持・管理に対してオーバヘッドの要因となっている．この問題克服のため「システムの分散構成によらない制御ソフトウェア」の確立を目指して検討をはじめた．

本稿では，上記課題実現のキーとなる，装置の分散構成をアプリケーションプログラムから隠す手法（ここでは透明性と呼ぶ）について検討結果を述べる．

2．要求条件

本検討は一般情報処理分野における『分散型ＯＳ[1]』の技術に相当するが，通信ノードシステムのアプリケーションプログラム（以降ＡＰＬと略す）を対象としていることから，一般ユーザを対象としている分散型ＯＳと比較して以下の条件が異なる．

(a) ＡＰＬの構造，書き方に制限を設けられる．

(b) 超多重処理が行われる．

(c) 実時間性に対する要求が厳しい．

(d) 信頼性に対する要求が厳しい．

3．ソフトウェアモデル

本検討の基本方針は，次の通りである．

(1) 条件(b),(c),(d) 満足の実績を持つ通信ノード用のソフトウェア技術を基盤とする．（図１参照）

・機能単位のモジュール化とこれをメッセージ結合型並行プロセスとして実現するＡＰＬ構成技術[2],[3]

・軽量プロセスの実時間実行制御技術[4]

(2) ソフトウェアの分散の単位はプロセス単位とする．

(3) 分散型ＯＳで検討されている透明性実現の技術を上記基盤に盛り込む．

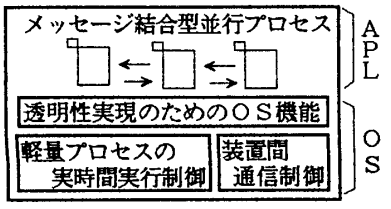

図１．ソフトウェアモデルの概要

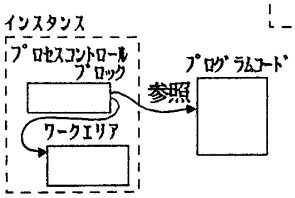

図２．軽量プロセスのイメージ

4．プロセスの分散配置に伴う透明性実現技術

本検討で扱う軽量プロセスは図２で示す構成になっている．ワークエリアを持ったプロセスコントロールブロック（これを総称して“インスタンス”と呼ぶ）はプロセス生成に伴い動的に割り当てられる．プログラムコードは制御装置上に静的に割り当てられ各プロセスによって共有される．従って，プロセスの分散配置に伴う透明性を実現するには，インスタンスレベルの透明性とプログラムコードレベルの透明性を考慮する必要がある．

4．1　インスタンスレベルの透明性

（1）概要　メッセージ転送する相手のプロセス（一般に，インスタンスのことをプロセスと呼ぶ）が違うサイト（以降，装置の分散単位をサイトと呼ぶ）にあっても意識することなく同一サイトのプロセスと同様にアクセスできることである．本検討での実現手法を図３に示す．サイト内／外の判別には分散型ＯＳでよく使われているプロセスに付与されるグローバルな名前を用いる[5]．

（2）名前通知方法　各プロセスへの名前の通知は，「プロセス生成時の名前の引き渡し方式」と「必要時（名前サーバ等への）問い合わせ方式」があるが，本検討ではこれらを混合して適用している．（**混合適用**）

超多重性の影響を考慮するためプロセスをコールプロセスとシステムプロセスに分類する．

コールプロセス	超多重の原因となるコール（呼）の発生に伴って生成されるプロセスで多いときには数千個に達する
システムプロセス	コールプロセスの生成やリソースの管理データの管理を行うプロセスでコールの発生に関係なく存在する

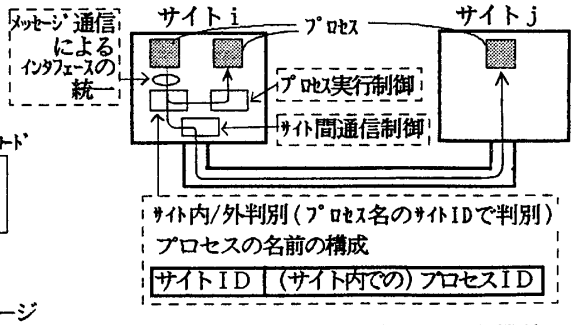

図３．動的レベル透明性実現機能

Implementation of transparent and location independent software for distributed communications node systems

Tatsurou MURAKAMI　Nobuo ARAKI

NTT Communication Switching Laboratories

〔コールプロセス〕数が多いこと，生成，消滅が頻繁なこと，１つのコールプロセスにアクセスするプロセス数が少ないことから問い合わせ方式はメモリ量，処理量の点で不利なため，名前の引き渡し方式をとる．

〔システムプロセス〕数は多くないが，全てのコールプロセスから参照される可能性があり，全てのコールプロセスが引き渡し方式でこの名前を持ち回るとメモリ量が冗長に使われるので，名前の問い合わせ方式をとる．

更に，システムプロセスは故障等に備えたサイトの独立性，データ参照の処理時間短縮を考慮して各サイトで多重置きされるため，各コールプロセスには通知する名前を限定することによって一重置きに見せている．この名前の限定には①被アクセスプロセスのサイトによりユニークに決まるものと②処理実行中に決まるものがあり，①はＯＳ内に，②はＡＰＬ内に名前変換テーブルを持たせている．

また，①に関しては名前問い合わせを該プロセスへのメッセージ転送と兼ねる，②に関しては通常の処理手順に名前管理とその問い合わせを埋め込むという**縮退形**で実現して，名前問い合わせだけのためのメッセージを排除した．縮退形の①に関する例を図４に示す．

名前通知方式を混合適用したこと，その中の問い合わせ方式を縮退形で実現したことの効果例を図５に示した．

４．２　プログラムコードレベルの透明性

通信ノードは蓄積プログラム制御方式であり，一般にインスタンスが参照するプログラムコードは同一サイトに存在する．しかし，複数サイトでのプログラムコードの共用化や特定プログラムコードに関する開発，管理の専用サイト化などの観点からインスタンスとプログラムコードを別サイト置く分散方式が考えられている．

このレベルの透明性は，プログラムコードがインスタンスと別サイトにあっても意識することなく同一サイトにあるプログラムコードと同様に実行できることである．

実現方式の概略を図６に示す．本検討では，ハードウェア制御時の実時間性を重視してプロセスコードをプロセス生成時のみインスタンスが生成されるサイトに転送，コピーして参照する方式をとっている．

透明性は以下のことより実現されている．

・プロセスの生成命令に対して，その論理名（これはプログラムコードに対応）からプログラムコードのサイト内／外を判別する．

・他プロセスからのアクセスは（インスタンスに対応する）名前をキーとするのでプログラムコードのコピー先の絶対アドレスに依存しない．

５．むすび

本検討では，分散配置されたインスタンス間の透明性を名前による手法で実現した．特に，ＡＰＬの性格を考慮した名前通知の混合適用，縮退形による実現方式が特徴である．更に，プログラムコードとインスタンスの分散配置に対する透明性もサポートして多様なシステム構成に対する柔軟性を考慮した．

〔参考文献〕(1)A.S.Tanenbaum et.al.:ACM Computing Surveys Vol.17,4,pp.419,1985 (2)近藤他：研実報Vol.33,11,pp.2599,1984 (3)丸山：信学技報SE86-20,1986 (4)久保田：信学論B,Vol.J70-B,4,pp.422,1987 (5)D.R.Cheriton:IEEE Software Vol.1,Apr.,pp.19,1984

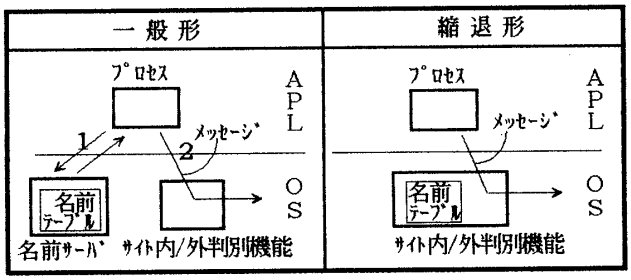

図４．名前問い合わせ方式の縮退形

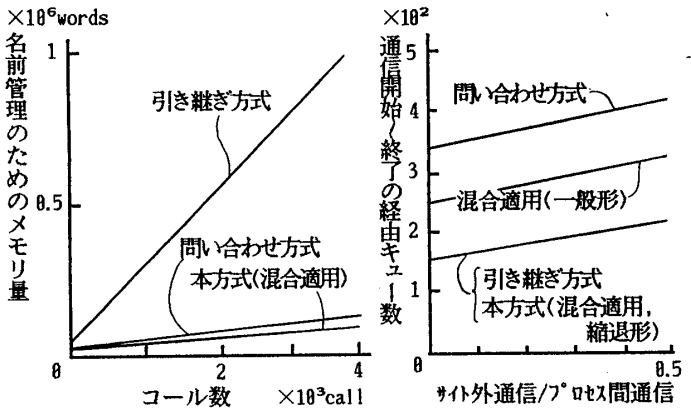

図５．名前通知方式の混合適用、縮退形使用の効果例

［プログラムコードレベルの透明性］

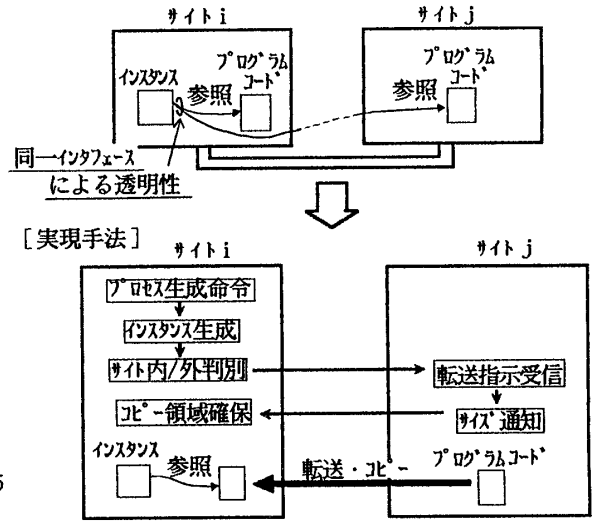

図６．プログラムコードレベルの透明性実現手法