論　文

分散型画像処理環境 VIOS

正員　松尾　啓志[†]　　准員　和田　錦一[†]
正員　岩田　彰[†]　　正員　鈴村　宣夫[†]

A Distributed Image Processing System VIOS

Hiroshi MATSUO [†], *Member*, Kinichi WADA [†], *Associate Member*, Akira IWATA [†] *and* Nobuo SUZUMURA [†], *Members*

あらまし　本論文では，ローカルエリアネットワークで結合された複数の計算機を用いて，画像処理を高速に行う分散型画像処理環境 VIOS を提案する．VIOS は，ネットワーク上にユーザとのインタフェースである VPE，画像オブジェクトの処理を行う IPU，ネットワーク上に分散されたオブジェクトの管理，および処理のスケジューリングを行う OM の三つのプロセスを配置した分散型画像処理システムである．本論文では，VIOS の設計方針，計算機への実装方法について示す．また分散処理を行う際に問題となるスケジューリング方法について検討し，計算機の負荷を予測する新しいスケジューリング方法を提案する．更に VIOS を計算機上に実装することにより，提案したスケジューリング方法と，従来用いられている種々のスケジューリング方法との比較，複数の計算機を用いた場合の処理速度の向上などについて検討した結果について示す．

キーワード　画像処理，分散処理，オブジェクト指向，ネットワークコンピューティング

1. まえがき

画像処理は一般に逐次的，思考錯誤的な作業が多く，その処理も画像間の加算，減算，マスク処理などの簡単な処理の組合せから，複雑な処理まで，多種多様な処理の組合せで実現される場合が多い．更に，現在，数多くの画像処理方法およびそのアルゴリズムが研究されている．

しかし複雑なアルゴリズムの記述は，画像処理の専門家以外では困難であることが多い．そのため，種々の画像処理ライブラリーが整備されている[(1)~(3)]．しかしそれらのライブラリーは，使用言語として FORTRAN を用い，主に大型中型計算機環境での使用を前提としており，近年急速に発展してきたカラーの濃淡画像が表示できるビットマップディスプレイを有し，大型計算機とは比較にならないマン・マシンインタフェースを備えたワークステーションが高速なネットワーク上で複数台結合された環境を想定したものではない．

† 名古屋工業大学電気情報工学科，名古屋市
Department of Electrical and Computer Engineering, Nagoya Institute of Technology, Nagoya-shi, 466 Japan

現在，新しい画像処理環境として View-Station が提案されている[(4)]．View-Station は画像処理プロセッサと汎用ワークステーションの結合を目指したシステムである．ソフトウェア環境面ではオブジェクト指向の考え方を導入した先進的なシステムである．

ところがワークステーションは，ウィンドウシステムに代表されるようなユーザインタフェースには優れているものの，画像処理を行う上での演算処理能力は必ずしも満足できるものではない．また非常に大きな計算力を必要とする画像処理手法も近年数多く開発されつつある．そのため，View-Station では画像処理専用ハードウェアとの密結合システムを想定している．また岡田らは，膨大な計算時間がかかる 3 次元画像の表示において，画像表示部分と計算部分を独立させて，計算部分は大型計算機で行うことにより，表示速度の向上を図る分散システムとして会話型画像表示環境 VISUAL を提案している[(5)]．

本論文では，ネットワークで結合された多数のワークステーションを用いる分散処理を導入することにより，処理の高速化を目指した分散型画像処理環境 VIOS の設計方針およびその実現について示す．更に，

分散処理において問題となるスケジューリング方法についての検討を行った結果について示す．システム名 VIOS はネットワーク上に分散された三つのプロセス名 (Visual programming editor, Image processing unit, Object manager) に由来する．なおシステムの開発言語としては C＋＋を使用した．

2. 分散画像処理システム VIOS の基本構成

2.1 システムの特徴

本システムは次の特徴を有する．

・分散処理

画像処理は処理の並列性が比較的高く，並列実行が可能な処理が多い．そこで大規模な画像処理演算を，高速なネットワークで結合された複数のワークステーションで構成されたワークステーション群で分散して実行することにより，処理速度の向上を図る．

・オブジェクト指向の導入

画像は縦横の画素数，1ピクセルのビット長および型などが種々多様であり，従来の画像処理プログラミングでは，ユーザはこれらの属性に注意する必要があった．本システムでは画像を画像データと手続きをもった画像オブジェクトとしてとらえることにより，画像の属性はすべて隠ぺいし，すべての処理を画像オブジェクトへメッセージを送ることによって実現する．その結果，本システムの使用者は画像の属性を考慮することなく，本来の画像処理プログラミングに専念できる．

2.2 画像データの取扱い

オブジェクト指向を導入することにより，すべての画像データは処理手続きと共にオブジェクトとしてカプセル化することが可能である．つまり画像に対するすべての処理は，オブジェクトにメッセージを送ることによって実現する．このことよりユーザは

・個々の画像の属性の管理

・画像データの加工に伴う，もとのデータの安全性の確保

を行う必要がなく，アルゴリズムの記述に専念することができる．

VIOS では画像オブジェクトを C＋＋のクラスによって実現した．VIOS でのクラス構造は，図1に示すように，基本クラス Image を頂点とした階層構造を有し，派生クラスとして，画像の種類により GrayImage (濃淡画像)，BinaryImage (2値画像)，FullcolorImage (カラー画像) の3種類のクラスに分類した．なお個々の画像データはオブジェクトとして三つの派生クラスのいずれかのクラスに属する．またそれぞれのクラスには図1に示した画像情報を有している．基本クラス Image では，個々のクラスに共通な情報 (画像の画素数，画素の型，名前，型変換時の画像の基準点となる画像原点) を定義し，派生クラスでは，個々のクラスに固有な情報や表示に必要な情報を定義した．濃淡画像 GrayImage では，濃淡画像データのほかに表示時に必要なウインドウパラメータなども保持している．カラー画像 FullColorImage では RGB 独立した画像データのほかに，ルックアップテーブルなどを保持している．

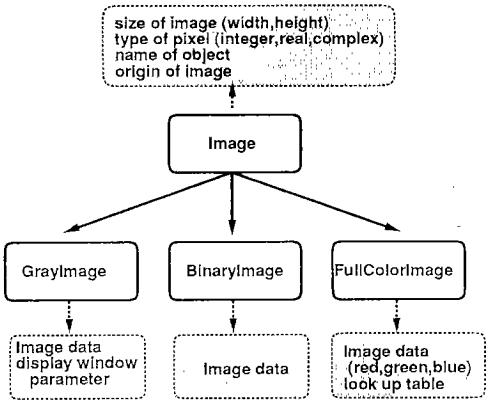

図1 VIOS における画像データ型
Fig. 1 Image data types by VIOS.

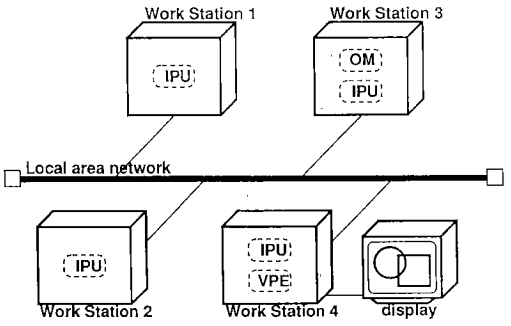

図2 ネットワーク上の3種類のプロセス
Fig. 2 Three kinds of processes in local area network.

2.3 システムの構成

VIOS では，ネットワーク上に図2で示すように3種類のプロセス (IPU，VPE，OM) を配置する．各プロセスはネットワーク上のワークステーション上に分散して配置，実行される．3種類のプロセスを以下に示す．

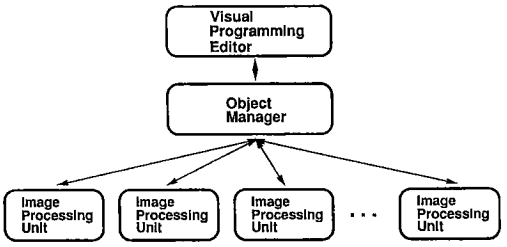

図3　3種類のプロセスの関係
Fig. 3　Relation of three kinds of processes.

・IPU (Image Processing Unit)

OMから送られてくる処理命令を実行する画像処理エンジン．既に存在するオブジェクトに処理を施すことにより新しいオブジェクトを生成する．またオブジェクトの内容を表示する処理も含まれる．なおIPUはネットワーク内で複数実行させることが可能である．

・VPE (Visual Programming Editor)

ユーザとのインタフェース．ユーザはVPE上で目的とする処理過程をアイコンを用いてプログラミングする．画像オブジェクトや処理モジュールを表すアイコンをマウスで画面上に張り付け，処理モジュールを線で結合することによりプログラミングを行う．

・OM (Object Manager)

画像オブジェクト，IPUの管理および処理スケジューリングを行うプロセス．VPEから送られてくる命令を解析し，適切なIPUへ処理を振り分けることにより処理の分散化を行う．

各プロセスの関係を図3に示す．

2.4　イメージプロセッシングユニット (IPU)

IPUは画像オブジェクトを保持し，OMから送られてきた処理内容に応じたメッセージをオブジェクトに伝え，演算処理を行う画像処理エンジンである．現在，オブジェクトに実装されているメソッドは，四則演算，(逆)フーリエ変換，フィルタ処理，画像の表示，ファイルへの読み書きなどである．なおVIOSでは，画像(定数，1次元を含む)を入出力にもつ処理モジュール単位で，処理のモジュール化を行うことが可能である．

処理モジュール(メソッド)の追加は，ユーザが新しい処理をC++で記述し，IPUにライブラリーとして結合することにより可能である．なおIPUのワークステーションへの実装方法は3.で示す．

IPU上での処理は「複数のオブジェクトを操作して新しいオブジェクトを生成する」ことを基本単位としている．すなわちいったん生成されたオブジェクトは，

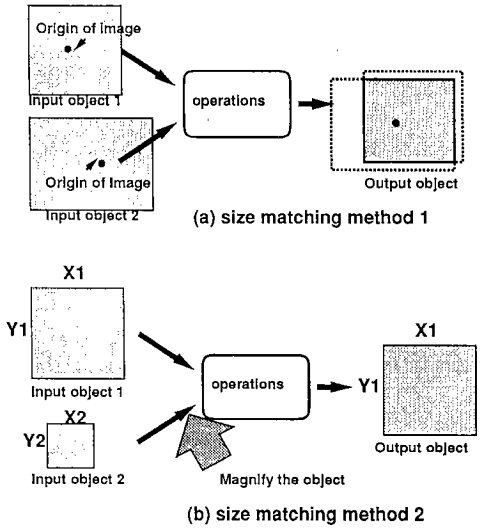

図4　オブジェクトサイズの変換規則
Fig. 4　Object size transformation method.

原則的には他の処理によって内容を変更されることはない．すなわち画像オブジェクトのレベルで単一代入の規則を有するものとする．

生成されるオブジェクトの型は，もとのオブジェクトの型によってIPUが自動的に決定する．IPUでは，画素単位の型変換はC言語と同様な変換を行う．更に画像の縦横の画素数すなわち大きさが異なるオブジェクト間の演算におけるサイズ変換も行う．サイズ変換規則としては図4に示すように

・画像原点を重ね，共通領域を新しい画像の大きさとする(図4(a))．

・小さい画像の整数倍拡大で，大きい画像の大きさと一致する場合は，小さい画像の大きさを大きい画像の大きさに拡大する(図4(b))．

の二つを実装し，前者を標準型変換規則とした．

2.5　ビジュアルプログラミングエディタ (VPE)

VPEは，VIOSとユーザのインタフェースである．ユーザは目的の処理をVPE上で，図5に示すようにアイコン間を線で結ぶことによりプログラミングを行うことが可能である．またVPE上では任意のオブジェクトを指定することにより画像をウィンドウ上に表示することが可能である．

VPE上で作成された処理はVPE-OM間通信コード(以下VOPコード)に変換され，オブジェクトマネージャに転送される．VOPコードは，以下のようなフィールドからなるコードである．

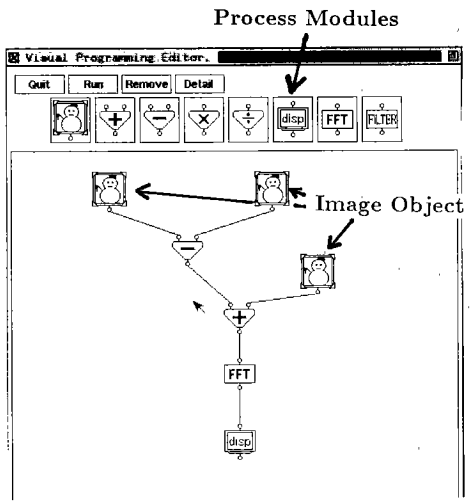

図5　Visual Programming Editor 上でのプログラミング
Fig. 5　Programming on visual programming editor.

（1）　第1フィールド（オペコード部分）にメソッド名とメソッドに対するメッセージを_で結合して記述する．

（2）　最後のフィールドに出力されるオブジェクト名を記述し，それ以前のフィールドにすべての入力オブジェクト名を記述する．

以下に VOP コードの一例を示す．

・FFT_I A W1

オブジェクト A を逆フーリエ変換した結果をオブジェクト W1 として生成する．

・FIL_L_10_10 W1 W2

オブジェクト W1 にカットオフ周波数 10/ナイキスト周波数，遷移幅 10 の低減フィルタ処理を行った結果をオブジェクト W2 として生成する．

・ADD A B W1

オブジェクト A，B を加算した結果をオブジェクト W1 として生成する．

2.6　オブジェクトマネージャ（OM）

OM は，画像オブジェクトがどの計算機の IPU 上に存在若しくは生成中であるかなどのオブジェクトに関する情報を管理し，更に VPE から送られた VOP を各 IPU に分散して実行するためのスケジューリングも行うプロセスである．

VIOS では分散実行の最小単位が，オブジェクトに対する演算であるため，実行単位の終了時間を予想することは困難である．従って，VIOS でのスケジューリングは，静的に行うのではなく，実行すべき処理に対する入力オブジェクトがアクセス可能になった処理から実行する動的なスケジューリング方式を採用した．OM の処理手順を以下に示す．

（1）　VPE から送られてきた VOP コードを解釈し，処理を実行待ちリストに登録する．

（2）　実行待ちリストの中から一つの処理を選び，コードの中に含まれる入力オブジェクトの状態（生成済み，生成中，未生成）を調べる．

（3）　（2）の結果，処理内容が実行不可能なら，実行待ちリストの最後に戻し，（2）に戻る．

実行可能な場合，OM が保有している情報から処理を依頼するのに最適な IPU を選択し，処理命令を送る．なお IPU の決定方法は **3.2** に示す．

（4）　実行待ちリストが空になるまで，（2），（3）を繰り返す．

以上の方法により OM は VPE から送られた処理内容を複数の IPU に割り振ることで処理の分散化を行う．

3.　分散処理の実装方法

3.1　RPC による分散処理の実装

VIOS では，各モジュールがネットワーク上の複数の計算機上で独立して動作する．従って各モジュールをプログラミングする際に，ユーザは他のモジュールとの通信機能も実装する必要がある．UNIX 上でネットワークを用いたプログラムを書く方法として，UNIX 4.2BSD で実装されたソケットを用いる方法がよく用いられる[6]．しかしこの方法はコネクションの確立から切断まで，すべてユーザが記述する必要がある．それに比べ RPC (Remote Procedure Call)[7] は，通常の関数呼出しとほぼ同様の手続きを用いて，ネットワーク間のプロセス間通信が可能である．そこで VIOS ではユーザが容易にモジュールの追加作成ができることに重点を置き，ユーザが記述する必要のあるプロセス間通信の手段としては RPC を採用した．

RPC はネットワーク間のプロセス間通信を容易に実現できる反面，

（1）　RPC では呼出し側はいったんプロシージャコールを行うと，呼び出した手続きが終了するまでウエイトサイクルに入るため，呼び出した手続きが終了するまで，他の処理を行うことができない．従って OM と IPU との間に RPC によるプロセス間通信を用いた場合，IPU での処理が終るまで，OM は停止する必要がある．これではネットワーク上に複数存在する IPU の制御は不可能である．

(2) プロシージャコールからリターンまでの時間が長いと TIMEOUT (規定値 25 秒) としてエラーとなる.

(3) 1度に転送可能なデータ量が少ない (規定値 8 kByte)

など種々の制約を受ける.

一般に画像処理は, 次に示す特徴を有すると考える.

・オブジェクトを構成するデータ量が非常に多いため (数百 kByte から数 MByte), 1回の RPC では転送できず, 数回に分ける必要がある.

・処理の演算時間が長い場合が多い (数秒から数十分).

この二つの制約は VIOS のプロセス間通信を RPC を用いて実現する際の問題点となった. そこで VIOS では RPC を用いた分散処理を以下の方法で実装した.

(1) OM から処理を受け取った IPU は図 6 に示すように共有メモリを確保した後, 子プロセスを生成 (fork) すると同時に, RPC の終了を OM に通知する. この時点で OM, IPU 間の RPC は終了することになる. 生成された子プロセスは親プロセスとは非同期に処理の実行を行い, 処理の終了後生成されたオブジェクトを共有メモリを用いて親プロセスに引き渡す. 更に処理の終了後 RPC を用いて OM に処理の実行終了のメッセージを送った後消滅する. このように, IPU を共有メモリにより結合された複数のプロセスを用いて構成することにより, OM は IPU に依頼した処理の終了を待つ必要がなく, 独立に処理を続けることが可能となる.

RPC による三つのプロセスの間の通信の過程を図 7 に示す.

(2) プロセス間での基本的な命令の通信には RPC を用い, IPU 間での画像オブジェクトの転送には UNIX4.2BSD で実装されたソケットを用いた. 転送方法を以下に示す (図 8 参照).

〈case 1〉 OM から送られてきた命令の中で, 必要とする入力オブジェクトが, ローカルな IPU にすべて存在する場合は, オブジェクトの転送は行わず, 直ちに目的の処理を行う.

〈case 2〉 必要とする入力オブジェクトが, ローカルな IPU でそろわない場合.

(a) 処理を実行する IPU は必要なオブジェクトを保持している IPU から RPC を用いてオブジェクトの情報および, ソケットの情報を得る. ここで転送元の IPU は新しい IPU を生成する.

(b) 転送先の IPU は得られたオブジェクトの情報を用いて, 共有メモリを確保し, 更に新しい IPU を生

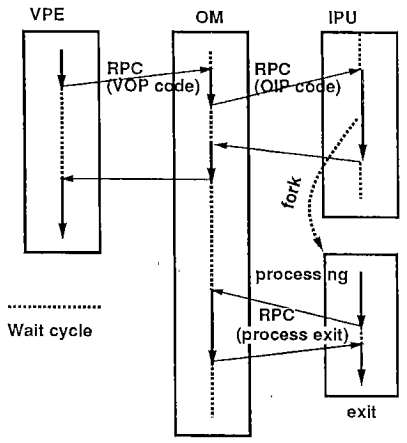

図 7 プロセス間通信の流れ図
Fig. 7 The diagram of process communication.

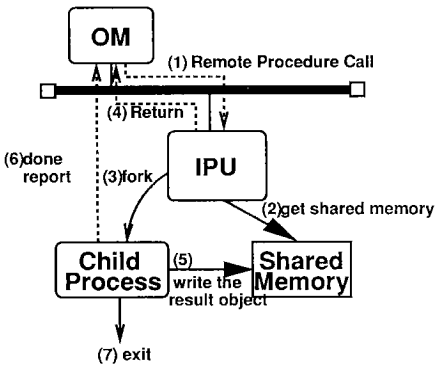

図 6 IPU の動作方法
Fig. 6 Receive RPC and fork the process cycle by IPU.

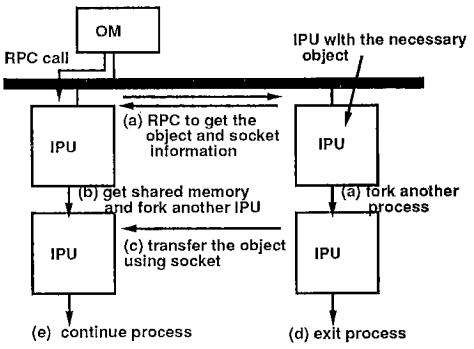

図 8 他の IPU からのオブジェクトの転送
Fig. 8 Object transfer protocol from another IPU.

成する．

（c） 生成された IPU 間でソケットを用いた通信チャネルの確保を行い，画像オブジェクトの転送を行う．

（d） 転送終了後，転送元の IPU は OM に対して RPC により処理の終了を伝えた後，消滅する．

（e） 必要な画像オブジェクトがそろった IPU は，目的の処理を実行する．

3.2 スケジューリング方法

2.6 で述べたように VIOS では OM 上で分散処理のスケジューリングを行う．スケジューリングに用いる情報としては，計算機の負荷とオブジェクトの転送の負荷を用いた．

しかし LAN 上の計算機の負荷は時々刻々と変化するため，適切なスケジューリングを行うには，OM が IPU の存在する計算機の負荷を正確に知る必要がある．従来，負荷情報を得るための試みとして，定期的同報通信方式と入札方式が一般的に用いられている[8]．この両方式の実装について今回 VIOS の実現を行った UNIX オペレーティングシステムを用いて考える．

定期的同報通信方式によるスケジューリングの実現方法として，clnt_broadcast システムコール（以下 clnt_broadcast()）を用いた方法が提案されている[9]．この方法は clnt_broadcast() による同報通信に対して，最も早く応答した計算機を最も負荷の軽い計算機とする方法である．

しかし clnt_broadcast() の応答時間は，応答するプロセスの状態（主記憶中への有無など）に影響され，必ずしも計算機の負荷の状態を反映しているとは限らない．また応答時間以外の要素，例えば画像オブジェクトの転送時間などを考慮した総合的なスケジューリングは困難である．

入札方式の場合は rwhod，若しくは rusers システムコール（以下 rusers()）によって得られる計算機の負荷情報を用いる方式が一般的である[10]．

rusers() を用いる方式によって得られた情報は，計算機のある時間内の平均負荷（以下 *LOAD*）の状態を正確に表していると考えることができる．本システムを実装した SUN-OS4.1.1 の場合式（1）で算出される[11]．

$$LOAD_{t+1}=LOAD_t\exp\frac{-1}{12}+\left(1-\exp\frac{-1}{12}\right)Run \qquad (1)$$

Run：実行および実行待ち状態のプロセス数

なお，式（1）による負荷の評価は 5 秒間隔で行われる．そこで VIOS では，OM 内部での負荷推定と，rusers() によって得られる負荷の二つを用い，瞬時的な負荷を予測する方法を用いた．式（2）に計算機の負荷（L_{IPU}）の算出式を示す．

$$L_{IPU}=NumIpuProcess+(LOAD-IpuLoad) \qquad (2)$$

NumIpuProcess：IPU から実行され，かつ現在実行中のプロセスの数．

LOAD：rusers() によって得られた負荷の観測値．（一定間隔（本実装では 30 秒）おきに IPU から OM に送る）

IpuLoad：IPU から実行されたプロセスによる負荷（*LOAD*）変化推定値

従って，右辺第 1 項は IPU で発生した負荷であり，第 2 項は IPU 以外で発生した負荷となる．なお *IpuLoad* は式（1）を用いて算出した．

次に転送負荷 L_{trans} の算出方法について示す．ある処理を他の IPU に転送する最適条件は，ローカルな計算機で処理を行う場合に比べ，早く処理が終了する場合である．つまり「転送時間＋処理時間＋結果の転送時間」の合計とローカルな計算機で行ったときの処理時間を比較する必要がある．しかし，各モジュールの処理時間を予想することは困難であるため，現在は種々のモジュールの平均処理時間を 10 秒と想定している．

この場合，転送負荷 L_{trans} はモジュールでの処理に必要な画像オブジェクトの大きさの関数と近似できる．従って，VIOS 上で，種々の大きさの画像オブジェクトの転送を行い，転送のための負荷を実験的に算出した．現在 1 MByte のオブジェクトの転送負荷は 0.25，0.5 MByte の場合は 0.2 である．

上記二つの負荷，すなわち各 IPU の瞬時負荷 L_{IPU} と転送負荷 L_{trans} を加算した式（3）を用いて各 IPU の負荷 L を求め，最小負荷になる IPU に OM が処理を依頼するスケジューリング方法を用いた．

$$L=L_{IPU}+L_{trans}(Size) \qquad (3)$$

Size：転送すべき画像オブジェクトの大きさ

4. VIOS の実現

以上の仕様をもとにして，分散型画像処理環境 VIOS の構築を行った．なおシステムの開発言語としては C＋＋(GNUg＋＋Ver1.39) を使用し，VPE は X11R4 をプラットフォームとして用いた．

4.1 分散スケジューリングアルゴリズムの評価

3.2 で示したスケジューリングアルゴリズムの評価を行った．まず，図 9 に示す処理の流れをもつスケジュー

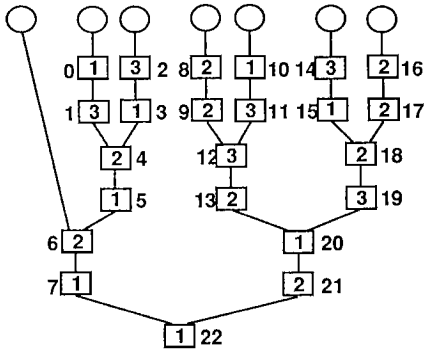

図9 スケジューリングアルゴリズムの評価に用いたプログラム

Fig. 9 Program using scheduling method evaluation. (each numeral inside a frame indicates processing time, each numeral outside a frame indicates module identification number)

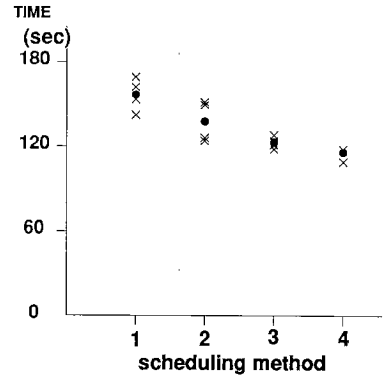

図10 スケジューリング方法による実行時間の比較

Fig. 10 Processing time using various scheduling algorithms. (× : processing time ● : average time)

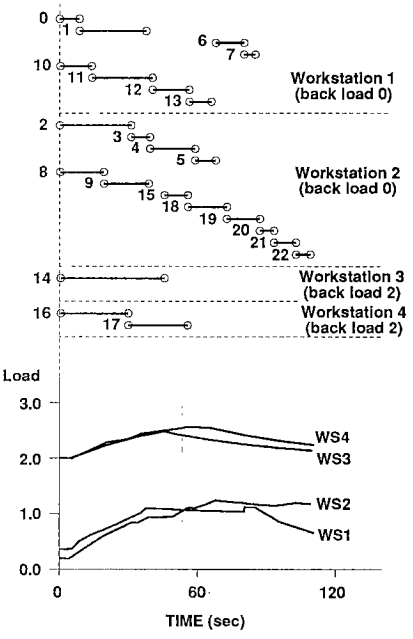

図11 各ワークステーションでのモジュールの実行過程と負荷(rusers())の推移

Fig. 11 The scheduling diagram and the *LOAD* value (rusers()) of each Workstation. (upper : The scheduling diagram lower : *LOAD* value of each Workstation)

リング評価プログラムをVPE上で作成した．なお，それぞれモジュール中の数字は，各モジュールの実行時間を示し，負荷1のモジュールの実行時間は，無負荷のSUN4-75上で実行させた場合，約4.7秒である．負荷2，3の場合はそれぞれ2倍，3倍の実行時間を必要とする．また画像オブジェクトの大きさは，1 MByteに設定した．

OM，VPE，四つのIPUをそれぞれ，別々の計算機(SUN4-75，SUNOS4.1.1)で実行し，更に四つのIPUにはそれぞれ，0，0，2，2の*LOAD*をあらかじめ付加した．なお，実験に用いたスケジューリング方法は，以下の四つである．

[方法1] clnt_broadcast()を用い，応答時間の一番早い計算機に処理を割り振る方法．

[方法2] rusers()によって得られた各計算機の負荷のうち，一番負荷の軽い計算機に処理を割り振る方法

[方法3] 方法2に加えて，**3.2**で示した式(2)を用いる方法．

[方法4] 方法3に加えて，画像オブジェクトの転送コストを考慮した式(3)を用いる方法

図10に各スケジューリング方法での実行時間を示す．なお計測はそれぞれ4回行った．×印は各回の実行時間を示し，●印は4回の平均実行時間を示す．方法1，2，3，4を適用した結果，それぞれ平均の実行時間は156.5, 137.5, 123.0, 116.0秒であった．

方法1は，方法2以下のrusers()を用いる方法に比べ実行時間が長いことが確認できる．つまり，clnt_broadcast()を用いるスケジューリングは計算機の瞬時の負荷を正確に測定することは困難であるため，適切なスケジューリングが行われていないためであると考えられる．更に他の三つの方法に比べて，VIOSで導入した方法4のスケジューリングは実行時間が速く，方法4が分散スケジューリング法として有効であること

が確認できた．

図11は，図9のプログラムの実行過程を上段に，IPUの動作をしている計算機の負荷(rusers()の値)の推移を下段に示した図である．上段の番号は図9のプログラム中の各モジュールの番号である．上段の実行過程の図から，各計算機の負荷に応じて，OMが適切なIPUを選んでスケジューリングしていることが確認できる．更に下段の負荷の推移図から，負荷分散が効果的に行われていることが確認できる．

表1 IPUの実行数と処理速度の関係

台　数	実験値(秒)	理想値(秒)	理想値/実験値
1	210	210	1.00
2	123	115	0.93
3	90	80	0.89

表2 IPUの実行数と処理速度の関係2
(コンピュータ断層法の再構成プログラムの実行)

台　数	実行時間(秒)	速度向上比率
1	139	1.00
2	75	1.85
3	54	2.57
4	47	2.96

次にIPUを実行する計算機を変化させたときの，実行時間の評価を行った．実行時間の測定には，図9のプログラムを用い，画像オブジェクトの大きさは1 MByteに設定した．表1にIPUの数を1から3まで変化させた場合の実行時間を示す．なおスケジューリング方法は方法4を用いた．更に各IPUの負荷を正確に知ることが可能な条件下で，最適なスケジューリングを行った場合の実行時間(理想値)も同時に示す．

2台，3台の場合における実験値と理想値の差が，オブジェクトの転送にかかるオーバヘッドである．複数のIPUで実行した場合，理想値の9割程度の実行速度を得ることが可能であった．

次にコンピュータ断層法(Computed Tomography)の再構成プログラム[12]をVIOS上に構築し，IPUの数

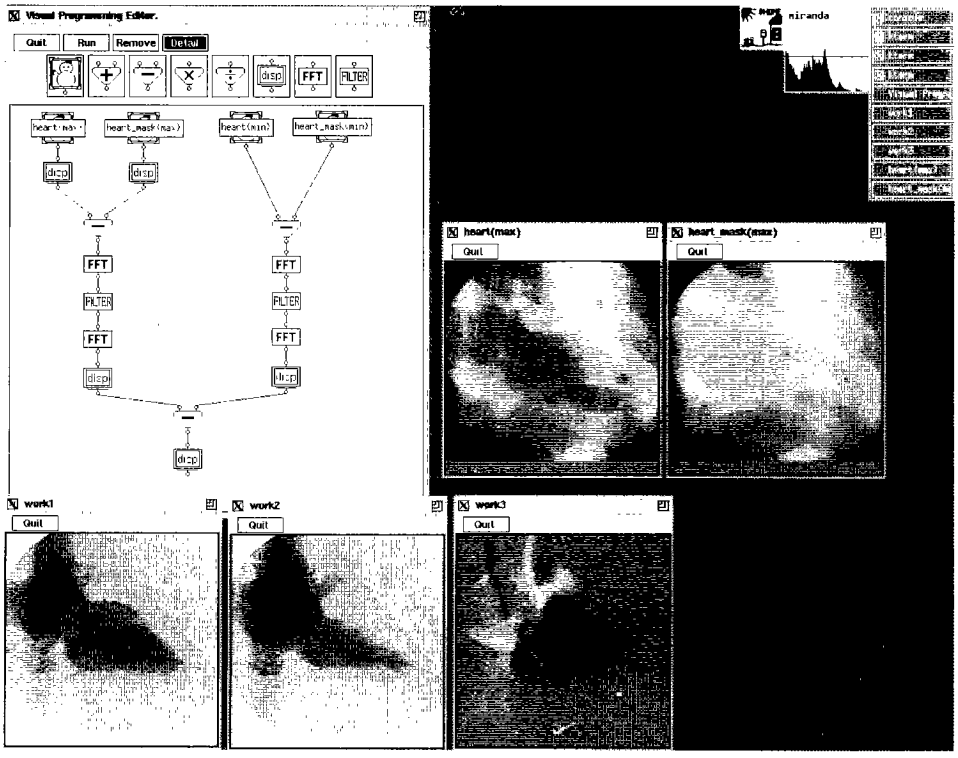

図12　VIOSの実現例
Fig. 12　Display image of VIOS.
(left upper is VPE window, the five images are image objects displayed by IPU).

を1から4まで変化させたときの実行時間を表2に示す．2台のIPUで実行した場合は，約2倍，4台のIPUで実行した場合に約3倍の速度向上ができることを確認した．

4.2 VIOSの表示例

図12はワークステーション上でVIOSを実行させたときの1画面である．VIOS上で記述した処理は，造影剤注入前後の拡張期心臓左心室造影像の差画像(heart(max)-heart_mask(max))にフーリエ変換，低域フィルタ処理，逆フーリエ変換を行った画像から，収縮期の左心室造影像に対して同様の処理(heart(min)-heart_mask(min))を行った画像を引くことにより，心臓の機能像(左心室拍出量)を算出する処理である．

5. むすび

オブジェクト指向を用いた分散型画像処理環境VIOSを提案した．本システムは，ネットワーク上にユーザとのインタフェースであるVPE，画像オブジェクトの処理を行うIPU，ネットワーク上に分散されたオブジェクト，IPUの管理，および処理のスケジューリングを行うOMの三つのプロセスを配置した分散システムである．更に分散スケジューリングアルゴリズムとして，rusers()によって得られるシステムの負荷と，更にIPUを実行中である計算機の負荷予測，および画像オブジェクトの転送コストを考慮した方法を採用した．実験により，clnt_broadcast()やrusers()のみでスケジューリングを行う方法に比べて，本方法が有効であることを確認した．更に実用例としてコンピュータ断層再構成法のプログラムを作成し，4台のIPUを用いて実験を行った結果，1台のIPUに比べて約3倍の処理速度を得ることが可能であることを確認した．本方式は特別なハードウェアを必要とせず，近年一般的になりつつある複数のワークステーションがネットワークを介して結合した分散システムに容易に導入できるため計算資源の有効利用に役立つものと考える．

今後，VIOSの拡張として次の点を検討中である．

・VPE上の簡易言語の開発

現在のVPEの仕様では，IPUにあらかじめ登録された処理のみが実行可能であり，処理の動的な追加は不可能である．しかしプロトタイピングツールとしての汎用性を考えた場合，より細かな処理を記述できる簡易言語が必要であると考える．そこで，処理をVPE上で定義できる画像処理用簡易言語の作成を検討中である．

・IPUの仮想計算機化

上記拡張に伴い，ユーザが定義した任意の処理をIPU上で実行するためには現状のVOPコードでは不十分である．そこでIPUをいろいろな制御構造をもち，しかも動的に手続きが定義可能な，仮想計算機として実装することにより，システムの拡張を柔軟に行えるようにする．

謝辞 本プロジェクトに参加し，システムの設計やプログラムの作成に多大な協力を頂いた本学大学院生加藤博之君，卒研生大野隆君に感謝致します．

文献

(1) 田村秀行，坂根茂幸，富田文明，横矢直和，金子正秀，坂上勝彦："ポータブル画像処理ソフトウエアパッケージSPIDERの開発"，情処学論，**23**, 3, pp. 321-328 (1982).

(2) 鳥脇純一郎，福村晃夫："画像処理用サブルーチンライブラリSLIPについて"，情処学論，**22**, 4, pp. 353-359 (1981).

(3) 鈴木秀智，鳥脇純一郎："3次元画像処理用サブルーチンパッケージSLIP-3Dについて"，昭60信学総全大．

(4) 佐藤宏明，岡崎 洋，河合智明，山本裕之，田村秀行："画像処理ワークステーションVIEW-Stationのソフトウエアアーキテクチャ"，情処学論，**31**, 7, pp. 1015-1026 (1990).

(5) 岡田 稔，北川英志，横井茂樹，鳥脇純一郎："3次元画像処理のための分散環境に関する検討"，信学技報，**PRU91**-23 (1991).

(6) 村井 純，井上尚司，砂原秀樹："プロフェッショナルUNIX"，アスキー出版局 (1986).

(7) Lyon B.: "Sun Remote Procedure Call Specification", Sun Microsystems, Inc. (1984).

(8) Tanenbaum A. S., van Renesse R.: "Distributed Operating Systems", ACM Computing Surveys, **17**, 4, pp. 419-470 (1985).

(9) 若林 進，山井成良："UNIXにおける動的負荷分散の試み"，jus 18th UNIX Symposium Proc, pp. 161-171 (Nov. 1991).

(10) Sun Microsystems Inc.: "Network Programming", Sun Microsystems Inc. (1988).

(11) Sun Microsystems Inc.: "System & Network Administration", Sun Microsystems Inc. (1988).

(12) Shepp L. A. and Logan B. F.: "The Fourier Reconstruction of a Head Section", IEEE Trans. Nuclear Science, **NS-21**, 7, pp. 21-43 (1974).

(平成3年6月24日受付，4年1月16日再受付)

松尾　啓志

昭58名工大・情報卒．昭60同大大学院修士課程了．同年松下電器産業(株)入社．平1名工大大学院博士課程了．同年名工大・電気情報・助手，現在に至る．画像処理，画像認識に関する研究に従事．工博．情報処理学会，ソフトウエア科学会，IEEE各会員．

和田　錦一

平3名工大・電気情報卒．現在，同大大学院博士前期課程（電気情報工学専攻）在学中．画像処理，音声認識に関する研究に従事．

岩田　　彰

昭48名大・工・電気卒．昭50同大大学院修士課程了．同年名工大・情報・助手，昭57年4月より昭58年10月まで，ドイツ連邦共和国ギーセン大学医学部医用情報研究所客員研究員．昭59名工大・情報・助教授．現在，名工大・電気情報・助教授．生体情報処理，医用画像処理，ニューラルネットワークに関する研究に従事．工博．日本ME学会，情報処理学会，IEEE各会員．

鈴村　宣夫

昭28名大・工・電気卒．民間会社勤務の後，昭38名大・工・助手．以後，講師，助教授を経て，昭49名工大・情報・教授．学科改組により，現在，電気情報工学科教授．この間，生体信号の計測，処理，生体関連の画像処理等の研究に従事．工博．