システムソフトウェアと
オペレーティング・システム 87－10
（２００１．　６．２８）

分散オペレーティングシステム Solelc における ウィンドウシステムの構成

丹羽 康裕 †　芝 公仁 †　大久保 英嗣 ††

† 立命館大学大学院理工学研究科　†† 立命館大学理工学部

従来のウィンドウシステムでは, 端末を単位としてディスプレイやマウスなどのハードウェア資源を管理している. したがって, アプリケーションが複数の端末を同時に使用するためには, プロセス間およびウィンドウシステム間の通信手段を独自に構築する必要がある. 我々が開発している分散オペレーティングシステム Solelc では, おのおのの端末が持つ資源を位置透過に管理するための機能をカーネルが提供している. したがって, この機能を使用することによって, ハードウェア資源の位置を意識することなく, 複数の端末を管理するウィンドウシステムの構築が容易となる. 本稿では, Solelc におけるウィンドウシステムの構成と処理方式について述べる.

キーワード: 分散オペレーティングシステム, ウィンドウシステム, マルチヘッド

The Structure of Window System on Solelc Distributed Operating System

Yasuhiro Niwa †　Masahito Shiba †　Eiji Okubo ††

†Graduate School of Science and Engineering, Ritsumeikan University
††Faculty of Science and Engineering, Ritsumeikan University

Since the conventional window system manages each hardware resource such as display and mouse as a unit of the terminal, it is necessary for each application to reconstruct the way of communication among processes and also among window systems in order to use plural terminals simultaneously. In Solelc distributed operating system which we have been developing, the kernel provides processes with functions for managing resources location transparently. Therefore, by using Solelc kernel functions, it has become to be easy to implement a window system which manages plural terminals. In this paper, the construction and processing scheme of the window system in Solelc is described.

key words: distributed operating system, window system, multi-head

1 はじめに

現在, 我々は, 分散オペレーティングシステム (以下 OS と記す) Solelc [1] の開発を行っている. Solelc は, 従来の OS とは異なり, 1 つの OS で同時に複数の計算機を管理することが可能である. 計算機毎にそれぞれ OS を動作させるのではなく, 1 つの OS ですべての計算機を管理することによって, システム全体を考慮した資源管理を行うことが可能となる.

Solelc では, OS を抽象化層とカーネルの 2 層に階層化している. 抽象化層では, 計算機資源の抽象化を行い, 位置透過な資源管理を可能とする環境を構築する. カーネルは, この環境上で動作するため, 位置透過な資源管理が可能となり, 任意の計算機上ですべての計算機を管理することができる. Solelc 上で動作するプロセスは, 位置透過に OS の機能を使用することができる. したがって, プロセスは実際に動作する計算機の性能や接続されたデバイスに制限されることなく, すべての計算機上の資源を使用することができる.

本稿では, 以上に述べた Solelc の特徴を活かしたウィンドウシステムの構成と処理方式について述べる. 従来のウィンドウシステムでは端末を単位としてハードウェアの管理を行っており, 複数のモニタを同時に使用するためには, プロセス間およびウィンドウシステム間の通信方法を検討する必要がある. これは, アプリケーション開発者にとって大きな負担となる. Solelc では, 位置透過な資源管理を可能とする環境がカーネルで構築されており, プロセスは位置透過に資源を使用することができる. これにより, Solelc では 1 つの端末を管理する従来のウィンドウシステムと異なり, 複数の端末を管理するウィンドウシステムを構築することが可能である. 位置透過に複数の計算機を管理するウィンドウシステムをユーザプロセスで構築することにより, アプリケーション開発者は端末間の通信や同期を考える必要はなくなり, 開発が容易となる. さらに, Solelc のウィンドウシステムでは, 新たな機能として, 特定ユーザ間での共有領域, マルチヘッド, デスクトップ分割の 3 つを実現している.

以下, 本論文では, 2 章で Solelc の概要, 3 章でウィンドウシステムの概要, 4 章でサーバの処理方式について述べる. 次に, 5 章で動作例を述べ, 最後に 6 章で本論文のまとめと今後の課題について述べる.

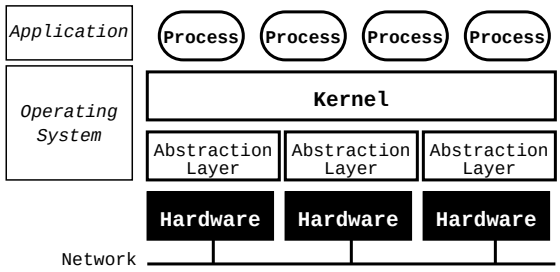

図 1 Solelc のシステム構成

2 Solelc の概要

Solelc の構成を図 1 に示す. Solelc では, OS が抽象化層とカーネルの 2 層に階層化されている. 下位層の抽象化層は, すべての計算機に配置されており, 各計算機上の資源を抽象化する役割を持つ. 抽象化層は, 他の計算機上の抽象化層と協調動作し, 位置透過な資源管理を可能とする環境を実現する. 上位層は 1 つのカーネルから構成され, システム全体の資源を管理する役割を持つ. カーネルは, 抽象化層が提供する環境上で動作するため, 任意の計算機上ですべての計算機の資源を管理することができる. また, カーネルはプロセスの実行環境を構築する. カーネルは, すべての計算機上のプロセスにサービスを提供できるため, プロセスも位置透過に動作することが可能である. 1 つのカーネルが, すべての資源を管理し, すべてのプロセスにサービスを提供するため, プロセスは任意の計算機上ですべての資源を利用することができる.

2.1 抽象化層

抽象化層は, 各計算機で 1 つずつ動作し, カーネルが動作するための環境を構築する. 抽象化層は, ハードウェアを直接操作する機能と, 資源の位置を管理する機能の 2 つの機能を持つ.

抽象化層は, CPU やメモリの管理を行う機能を持つ. CPU は, スレッドとして抽象化され管理される. カーネルやプロセスは, スレッドを用いることによって, 1 つの CPU で複数の処理を並行して実行できる. また, 抽象化層は, すべての計算機から共有される単一の仮想アドレス空間を構築し, これをカーネルやプロセスに提供する. 各計算機に接続された周辺デバイスも抽象化層によって操作される. 抽象化層は, デバイスドライバを持ちデバイス

を直接操作する機能を持つ. また, 割込みハンドラも抽象化層内に存在し, 割込み発生時には抽象化層がこれを取得し割込みの種類に応じた処理を行う.

また, 抽象化層は, 他の計算機上で動作する抽象化層と協調することによって, 資源の位置を管理する. 抽象化層は, システムで使用するすべての資源にシステム全体で一意な識別子を付与する. 資源とは, Solelc で管理される計算機とそれらが持つ CPU やメモリ, ディスクなどの周辺デバイスである. 抽象化層は, 識別子を用いて, どのディスクがどの計算機に接続されているかといった資源の位置情報を管理する.

2.2 カーネル

カーネルは, システム全体で 1 つであり, 抽象化層が提供する機能を用いて, システム全体の資源を管理する. 抽象化層が提供する機能は位置透過に使用することができるため, カーネルは任意の計算機上で動作可能である. また, カーネル内のすべての機能が同一の計算機上で動作する必要はなく, カーネルを構成する各モジュールをそれぞれ異なる計算機上で動作させることも可能である. この場合も, 抽象化層が実現する位置透過性によって, 各モジュールは他のモジュールと同一の計算機上で動作しているかのように処理を行うことができる.

カーネルは従来の OS が実現しているものと同様の機能を実現する. すなわち, プロセスの実行環境を構築し, 各プロセスにシステムの資源を適切に分配する. また, プロセスに対して, ファイル操作などのサービスを提供する. 計算機毎にカーネルを配置し, これらを協調動作させることによって資源管理やサービスを実現する場合, 複雑な処理が必要になりカーネルの構造が複雑になるといった問題がある. Solelc では, 1 つのカーネルですべての計算機を管理することによって, カーネルの構造を単純化している.

2.3 プロセス

プロセスは, ユーザ権限で動作するスレッドの実行環境である. プロセスは, カーネルによって実現され, カーネルが提供するサービスを使用して動作する. プロセスのサービス要求は, 抽象化層によって転送され, これを受け取るべきカーネルに通知される. そのため, プロセスやカーネルは, 互いの位置を意識することなく動作することができる.

3 ウィンドウシステムの概要

Solelc では, 位置透過な資源管理を可能とする環境が構築され, その環境上でカーネルを動作させることにより, 同時に複数の計算機を管理することを可能としている. 本章では, このような環境に適合し, 新たな機能を付加したウィンドウシステムの構成について述べる. ここで, 以下の説明で使用するモニタ, デスクトップ, ユーザの 3 つの概念を定義しておく.

- モニタ
  1 つの CRT もしくは LCD の画面全体である. X Window System[2] の定義するディスプレイと区別する.
- デスクトップ
  ユーザーから見て、ウィンドウの最背面にあたる部分であり, X Window System の定義するルートウィンドウである.
- ユーザ
  本ウィンドウシステムの利用者であり, 一意に識別する整数値としてユーザ ID が割り振られる.

3.1 Solelc におけるウィンドウシステム

本ウィンドウシステムは, 1 つのユーザプロセスとして動作する. ウィンドウシステムで必要となる機能を抽象化層, カーネル, プロセスの各々で実現している. グラフィックボードの設定, 参照, 変更は, ハードウェアを直接操作する機能を持つ抽象化層で行う. 計算機の資源情報の参照は, 位置透過な資源管理を行っているカーネルで実現する. ウィンドウおよびイベントの管理といったウィンドウシステムの中核部分は, プロセスとして実現する. 本ウィンドウシステムでは, 任意のモニタの特定領域ごとに処理を分割することにより, 以下の 3 つの機能を実現している.

- 特定ユーザ間での共有領域
  グループでの討議や共同作業の支援を想定し, 特定ユーザ間で同じ表示となる領域の生成を可能とする. 共有領域では, 複数のユーザのポインタを表示し, キーボード入力は各々のユーザが指定したウィンドウへの入力とする. 共有されていない領域から共有されている領域へのウィンドウの移動も可能である.

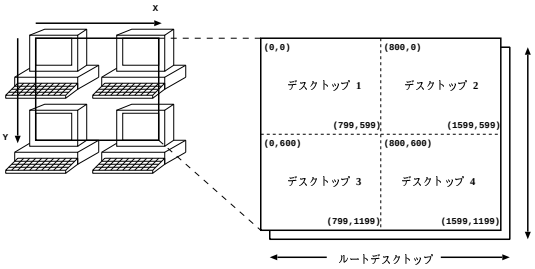

図 2　デスクトップの管理手法

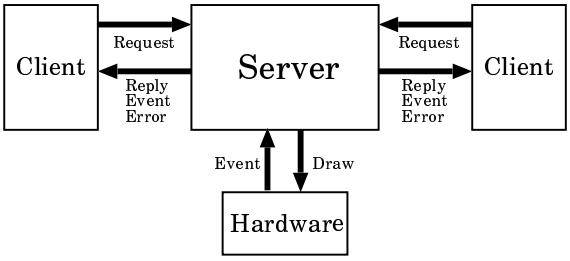

図 3　クライアントとサーバの役割分担

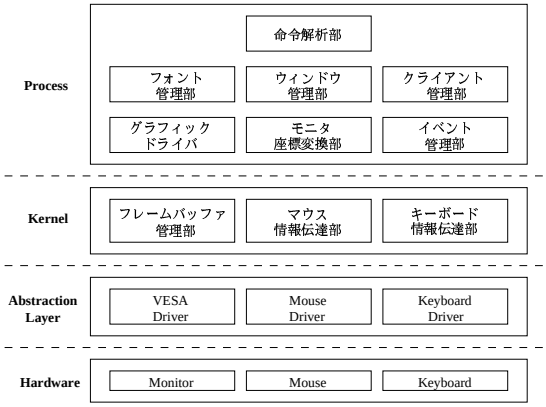

図 4　ウィンドウシステムの構成

- マルチヘッド
  複数のモニタを使用したプレゼンテーションや特定ユーザ間での共有領域の表示の支援を想定し, 複数のモニタ上に 1 つのルートデスクトップを実現する機能である (図 2 参照). ユーザは, 同時に複数のモニタを使用可能である. モニタ上の領域を横方向に繋ぐだけではなく, 縦方向に繋ぐことも可能である.

- デスクトップ分割
  ルートデスクトップ内に複数のデスクトップを生成する機能である. ユーザは 1 つのモニタ上で複数のデスクトップを使用することができる.

本ウィンドウシステムでは, ユーザは 1 つのキーボードとマウス, 複数のモニタを使用することができる. ユーザは 1 つのルートデスクトップを持つ. ルートデスクトップは複数のデスクトップを持つ. 共有領域は 1 つのデスクトップを共有領域に割り当てることにより実現される.

3.2　サーバ・クライアント方式

本ウィンドウシステムでは, X Window System に倣い, モニタ, キーボード, マウスを管理し, サーバとクライアントの 2 つで処理を分担する. サーバはウィンドウを表示し, クライアントはイベントを受け取って, ウィンドウイメージを生成する (図 3 参照). クライアントは複数起動可能であり, 各クライアントはサーバとメッセージをやりとりすることにより, 複数のウィンドウを複数のモニタ上に表示する. このように機能分割することにより, クライアントは自分自身が生成したウィンドウのみを管理すればよく, 他のクライアントが生成したウィンドウを知る必要はない. ポインタの位置などクライアントが管理する座標は, モニタに対する座標ではなく, 生成したウィンドウに対する座標となり, 管理が容易になる.

クライアントは, サーバに対して描画リクエストと情報リクエストを発行する. サーバは, クライアントに対して, イベントや情報リクエストへのリプライ, エラーレポートを通知する. クライアントとサーバ間のメッセージは, X Window System に倣い, X プロトコルを用いて, リクエスト, リプライ, イベント, エラーの 4 種類を規定している.

3.3　全体構成

本ウィンドウシステムの構成を図 4 に示す. 本ウィンドウシステムは, プロセス, カーネル, 抽象化層のそれぞれに機能を分割して実現している. サーバは, 命令解析部, クライアント管理部, フォント管理部, ウィンドウ管理部, イベント管理部, モニタ座標変換部, グラフィックドライバから構成される. それぞれの役割を以下に述べる.

- 命令解析部
  クライアントとの接続を検査し, 応答メッセージを生成する. クライアントからの命令は, 命令解析部でのみ受け付ける. 接続に関する要求以外の命令は, ウィンドウ管理部に送られる.

- ウィンドウ管理部
  ウィンドウの生成, 削除, 属性変更, 位置管理を行う. また, デスクトップの管理と共有領域の管理も行う. 命令解析部やイベント管理部から通知されたリクエストの中に含まれるウィンドウに対する座標をルートデスクトップに対する座標に変換する.

- フォント管理部
  フォント情報を管理する. ウィンドウ管理部からのフォントの要求により, フォントデータを提供する.

- モニタ座標変換部
  ウィンドウ管理部から通知されたリクエストの中に含まれるルートデスクトップに対する座標を, 各計算機のモニタに対する座標に変換する. 複数のモニタにわたる座標が含まれている場合は, 複数のリクエストに分解する.

- グラフィックドライバ
  モニタ座標変換部からの命令により, ビットマップイメージを生成し, フレームバッファに書き込む. VRAM がサーバのメモリ領域内に割り付けられていない場合は, フレームバッファ管理部に依頼し, VRAM をサーバのメモリ領域内に割り付ける.

- イベント管理部
  マウスポインタとウィンドウにおけるイベントマスクを管理する. マウス情報伝達部とキーボード情報伝達部からマウスとキーボードの出力データを取得する. ウィンドウ管理部から取得したアクティブウィンドウの情報から, イベントを発行する. 発行されたイベントは, クライアント管理部に送られる. ポインタが移動した場合は, ウィンドウ管理部にポインタ表示命令を送る.

- クライアント管理部
  ウィンドウとクライアントとの対応を管理する. クライアント管理部は, リプライ, イベント, エラーを該当するクライアントに通知する.

本ウィンドウシステムは, デスクトップを単位としてモニタの表示領域を管理し, ウィンドウ管理部, モニタ座標変換部の各々で, ウィンドウに対する座標からルートデスクトップにおける座標へ, ルートデスクトップに対する座標からモニタに対する座標へ変換する. 共有領域に指定されたデスクトップに対する描画リクエストの場合は, 共有領域に指定されたすべてのデスクトップに対してリクエストを生成する.

カーネルでは, フレームバッファ管理部, マウス情報伝達部, キーボード情報伝達部を実現している. それぞれの役割を以下に述べる.

- フレームバッファ管理部
  グラフィックドライバからの要求により VRAM をサーバのメモリ領域内に割り付ける. これにより, メモリの書き込みによって描画を行うことが可能となる. モニタ座標変換部からの命令により VESA の設定を参照・変更する場合は, VESA ドライバに命令を発行する. また, VESA ドライバから送られてきたモニタの解像度や色の深さなどの VESA 設定情報をモニタ座標変換部に送る.

- マウス情報伝達部
  各計算機のマウスドライバからマウスの出力データを取得し, イベント管理部に提供する.

- キーボード情報伝達部
  各計算機のキーボードドライバからキーボードの出力データを取得し, イベント管理部に提供する.

抽象化層では VESA ドライバ, マウスドライバ, キーボードドライバを実現する. それぞれの役割を以下に述べる.

- VESA ドライバ
  フレームバッファ管理部からの要求により, モニタの解像度や色の深さなどの VESA 設定情報の提供や変更を行う.

- マウスドライバ
  マウスからの出力データを取得し, これをマウス情報伝達部に提供する.

- キーボードドライバ
  キーボードからの出力データを取得し, これをキーボード情報伝達部に提供する.

マウス情報伝達部, キーボード情報伝達部が, 抽象化層で受け取った出力データを収集し, イベント

管理部がマウス情報伝達部, キーボード情報伝達部から出力データを取得することにより, すべての入力装置からのデータをイベント管理部で処理することが可能となる.

3.4 特徴

本ウィンドウシステムは, 同時に複数のデバイスを管理することができる. 特徴的な例として, マウスとフレームバッファについて述べる.

3.4.1 マウスの管理

マウスの管理はイベント管理部で行われる. イベント管理部は, ウィンドウシステムが管理を行うマウスの数だけスレッドを生成し, それぞれのスレッドが read システムコールを行うことにより, マウス情報伝達部からマウスの出力データを取得する. スレッドの処理手順を以下に示す.

1. /dev/psm** で表されるマウスのデバイスファイルをオープンする. ** には計算機 ID が設定される.
2. read システムコールによって, マウスの出力データを取得する.
3. マウスの出力データからマウスイベントを生成する.

処理手順 2, 3 を繰り返すことにより, マウスの動作を取得する. このとき, Solelc の位置透過な資源管理により, マウスの位置を意識することなく出力データを取得することが可能である. また, 本ウィンドウシステムの管理する計算機の数によらず, 同一の方法でマウスの機能を実現することができる.

3.4.2 フレームバッファの管理

グラフィックドライバは, 本ウィンドウシステムが管理を行うモニタの数だけスレッドを生成し, スレッドを各モニタに割り当てる. グラフィックドライバは, モニタ座標変換部から送られてきた命令を, 対象モニタを担当するスレッドに送る. スレッドは, ビットマップイメージを生成し, フレームバッファに書き込む. 描画を行うスレッドの処理手順を以下に示す.

1. /dev/fb** で表されるフレームバッファのデバイスファイルをオープンする. ** には計算機 ID が設定される.
2. グラフィックドライバから命令を受け取る.
3. グラフィックドライバからの命令を解析し, ビットマップイメージを生成する.
4. フレームバッファにビットマップイメージを書き込む.

処理手順 2, 3, 4 を繰り返すことにより, モニタへの描画を行う. このとき, Solelc の位置透過な資源管理により, モニタの位置を意識することなく描画可能である. また, 本ウィンドウシステムの管理する計算機の数によらず, 同一の方法でモニタへの描画を実現することができる.

4 サーバの処理方式

前章では, 本ウィンドウシステムを構成する各構成要素の処理方式について述べた. 本章では, デスクトップ分割, 特定ユーザ間での共有領域, マルチヘッドの 3 つの機能の処理方式について述べる.

4.1 デスクトップ分割

デスクトップの管理はウィンドウ管理部で行われる. ウィンドウの位置はルートデスクトップを基準として管理される. また, 任意のデスクトップ上でウィンドウを表示させることができ, デスクトップ間のウィンドウの移動も可能である.

デスクトップ管理レコードを表 1 に示す. デスクトップを一意に識別する整数値としてデスクトップ ID を定義する. デスクトップ ID は, 32 bit で管理し, 上位 16 bit にはユーザ ID が含まれる. 下位 16 bit が 0x0000 のデスクトップ ID をルートデスクトップとする. デスクトップの属性として, 共有領域 ID, ルートデスクトップにおける x, y の座標, デスクトップの大きさを持つ. 共有領域は, 1 つのデスクトップを共有領域に割り当てることにより生成される. デスクトップが共有領域となっていない場合は, 共有領域 ID を 0 とする. このように定義することにより, あらゆるデスクトップに対して一意な ID を設定することができる. また, 自由にデスクトップの位置や大きさを変更することも可能である.

表 1　デスクトップ管理レコード

フィールド	意味
desktopID	デスクトップ ID
sharedID	共有領域 ID
originX	ルートデスクトップにおける x 座標
originY	ルートデスクトップにおける y 座標
width	x 方向の大きさ
height	y 方向の大きさ

4.2　特定ユーザ間での共有領域

共有領域の管理は, ウィンドウ管理部で行われる. デスクトップ間のウィンドウの移動が可能であることから, 共有されていない領域から共有されている領域へのウィンドウの移動が可能である. 共有領域は, 権限を持つユーザのみ使用可能としている.

ウィンドウ管理部では, 命令解析部やイベント管理部から送られてきた描画命令に含まれるウィンドウに対する座標を, ルートデスクトップ ID とルートデスクトップにおける座標に変換する. 次に, 命令解析部やイベント管理部から送られてきた描画命令が, 共有領域に対して行われているか否かを確認する. 共有領域に対しての描画命令である場合は, 共有領域に指定されたすべてのデスクトップに対して描画命令を生成し, モニタ座標変換部に送る. そうでない場合は, そのままモニタ座標変換部に描画命令を送る.

4.3　マルチヘッド

マルチヘッドは, モニタ座標変換部で実現される. モニタ管理レコードを表 2 に示す. モニタを一意に識別する整数値としてモニタ ID を定義し, 属性として, モニタを使用するユーザの ID, マルチヘッドのレイアウト座標, 解像度, 色の深さを持つ. 図 5 の 4 つのモニタを使用してマルチヘッドを行った場合, レイアウト座標は, 左上の画面の場合 (0,0) 右下の画面は (1,1) となる. 解像度や色の深さはフレームバッファ管理部を呼び出し, VESA 設定情報から得る. モニタ ID は Solelc で割り当てられた計算機 ID と対応付けすることにより, すべてのモニタに対して一意に設定される.

モニタ座標変換部では, ウィンドウ管理部から通知された描画命令の中に含まれるルートデスクトップ ID とルートデスクトップに対する座標を, モニタ ID とモニタに対する座標に変換する. 複数のモニタにわたる座標が含まれている場合は, 複数のリクエストに分解する.

表 2　モニタ管理レコード

フィールド	意味
monitorID	モニタ ID
userID	モニタを使うユーザの ID
layoutX	マルチヘッドのレイアウト x 座標
layoutY	マルチヘッドのレイアウト y 座標
width	x 方向の解像度
height	y 方向の解像度
depth	色の深さ

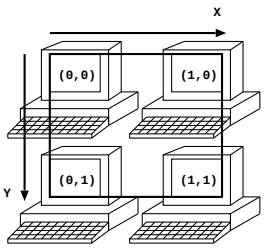

図 5　マルチヘッドのレイアウト座標

5　動作例

前章では, デスクトップ分割, 特定ユーザ間での共有領域, マルチヘッドの 3 つの機能の処理方式について述べた. 本章では, サーバがマウスのクリックをクライアントに通知し, クライアントの描画命令によって 2 画面にわたり線が表示される過程を説明する (図 6 参照).

1. マウスドライバは, マウスの出力データを取得し, データをバッファに置く.
2. マウス情報伝達部は, マウスドライバのバッファからデータを取得し, マウス情報伝達部のバッファに置く.
3. イベント管理部は, read システムコールによりマウス情報伝達部のバッファからデータを取得し, バッファのデータをマウスの動作に変換する. 前回のマウスボタンの状態を比較し, クリックされたと判断する. ポインタが存在するウィンドウのイベントマスクを参照し, マウスのクリックをクライアントに通知しなければならないと判断する.
4. ルートデスクトップにおけるウィンドウの座標を得るために, ウィンドウ管理部を呼び出す.
5. ウィンドウ管理部は, イベント管理部にウィンドウ座標を返す.

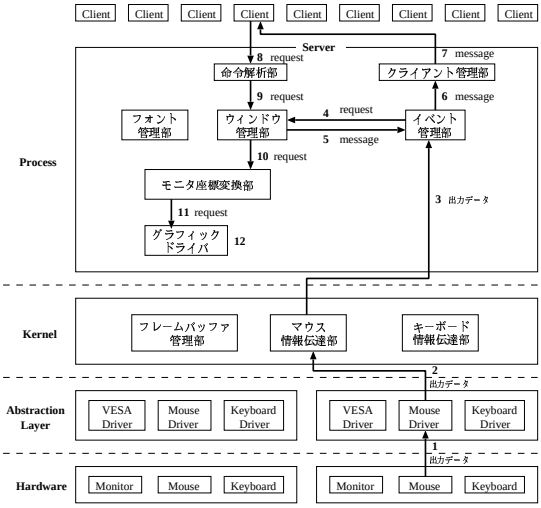

図 6　本ウィンドウシステムの処理の流れ

6. イベント管理部は, ルートデスクトップにおけるウィンドウの座標とルートデスクトップにおけるマウスの座標から, ウィンドウに対するマウスの座標を計算し, イベントを生成する. イベント管理部は, イベントをクライアント管理部に送る.

7. クライアント管理部は, クライアントにイベントを通知する.

8. クライアントは, イベントを取得し, マウスがクリックされたことを知る. クライアントは, 線を書くリクエストを発行する.

9. 命令解析部は, リクエストを解析する. 接続に関する要求以外の命令と判断し, リクエストをウィンドウ管理部に送る.

10. ウィンドウ管理部は, リクエストに含まれるウィンドウに対する座標をルートデスクトップ ID とルートデスクトップにおける座標に変換し, リクエストをモニタ座標変換部に送る.

11. モニタ座標変換部は, リクエストに含まれるルートデスクトップ ID とルートデスクトップにおける座標から, モニタ ID とモニタに対する座標に変換する. モニタ座標変換部は, 2 画面にわたる描画であると判断し, 2 つのリクエストを生成する. モニタ座標変換部は, リクエストをグラフィックドライバに発行する.

12. グラフィックドライバは, リクエストに含まれるモニタ ID とモニタにおける座標から, フレームバッファへ書き込む位置を計算する. グラフィックドライバはフレームバッファに書き込みを行い, モニタに線が描かれる.

ウィンドウ管理部で座標を変換することにより, 特定ユーザ間での共有領域やデスクトップ分割が実現される. モニタ座標変換部で座標を変換することにより, マルチヘッドが実現される. すなわち, 上記の過程を経ることにより, クライアントに機能を加えることなく, 特定ユーザ間での共有領域, マルチヘッド, デスクトップ分割が実現される.

6　おわりに

本稿では, 分散 OS Solelc におけるウィンドウシステムの構成と処理方式について述べた. 本ウィンドウシステムは, 1 つのユーザプロセスで構成され, カーネルの機能を用いることにより複数のモニタ, キーボード, マウスを管理する. デスクトップを単位としてモニタの表示領域を管理し, ウィンドウ管理部, モニタ座標変換部の各々で, ウィンドウに対する座標からルートデスクトップにおける座標へ, ルートデスクトップに対する座標からモニタに対する座標へ変換する. 共有領域に指定されたデスクトップに対する描画リクエストの場合は, 共有領域に指定されたすべてのデスクトップに対してリクエストを生成する. これらの機能によって, クライアントに機能を加えることなく, 特定ユーザ間での共有領域, マルチヘッド, デスクトップ分割の 3 つの機能が実現された. 今後は, 従来のウィンドウシステムでこれらの機能を実現した場合との性能比較を行う予定である.

参考文献

[1] 芝公仁, 大久保英嗣: “分散オペレーティングシステム Solelc の構成”, 情報処理学会研究報告, 2000-OS-84, Vol. 2000, No. 43, pp. 237-244 (2000).

[2] Adrian Nye 著, 石川和也 訳: “X プロトコル・リファレンス・マニュアル”, ソフトバンク株式会社 (1993).