K-068

複数台PCを用いた没入型ディスプレイのための ソフトウェア環境に関する一検討

A Software Environment for Immersive Projection Displays with Multi-PCs

石田 善彦† 橋本 直己† 佐藤 誠†
Yoshihiko Ishida Naoki Hashimoto Makoto Sato

1. はじめに

近年，没入型ディスプレイの映像生成のためにPCクラスタが多く利用されるようになっている．PCクラスタを用いると，高精細な映像を実時間で描画できるシステムが安価に実現できる一方で，PC間での並列・分散処理が必要となり，従来とは異なるプログラミングモデルに基づいてアプリケーションを開発しなければならない．そのため，アプリケーションが複雑化して開発が困難になり，また，従来のアプリケーションがそのままでは利用できないという問題が生じている．そこで本稿では，これまでの開発環境やソフトウェア資源を有効に活用できるように，スタンドアロンで動作することを前提としたアプリケーションを，クラスタベースの没入型ディスプレイで実行可能にするソフトウェア環境を提案する．

2. 関連研究

これまで没入型ディスプレイ環境でアプリケーションを実行するために様々な開発・実行環境が提案されている．Chromium [1] は，グラフィックライブラリの描画命令を操作し，各PCに必要な描画命令を分散して送信することで，没入型ディスプレイに対応できるフレームワークである．ソースコードを修正しなくてもクラスタ環境で実行できるという利点がある一方，複雑なシーンの描画では通信量が増大するため，ネットワークがボトルネックになることが指摘されている．また，VRJuggler [2] のようにクラスタをサポートするVRのためのアプリケーションフレームワークも提案されている．これを利用することにより，クラスタ環境を意識せずにアプリケーション開発を行うことが可能であるが，既存のアプリケーションに関しては移植が必要なため，これまでのソフトウェア資源を活用しづらいという問題がある．

3. API介入処理による没入型ディスプレイへの対応

本稿では，API(Application Programming Interface)呼び出しをアプリケーションと実行環境とのインタフェースとして捉え，このインタフェースに介入することによって没入型ディスプレイへの対応を実現するシステムを提案する．本システムでは，ソースコードを修正することなく，既存のアプリケーションを没入型ディスプレイ環境で実行することが可能である．

クラスタベースの没入型ディスプレイにおいて整合性のある表示を行うためには，大きく分けて2つのソフトウェア処理が必要となる．1つは，PC間でプロセスを同期する処理で，もう1つは，没入型ディスプレイのマルチプロジェクションに対応するための描画の変換処理である．多くのシステムでは，開発者がこれらの処理をアプリケーション内にコーディングすることで対処するが，本稿ではソースコードの修正等の作業を行うことなく対応することに主眼を置いているため，アプリケーション側での対処はできない．そこで，本システムでは，アプリケーション自体ではなく，アプリケーションと実行環境とのインタフェースに着目し，相互間の処理に介入することによって必要な処理を実行する．具体的にインタフェースとしては，アプリケーションがOSやグラフィックライブラリの機能を利用するために呼び出すAPIが挙げられる．本システムでは，このインタフェースに対して介入処理を行うミドルウェアを導入することで，没入型ディスプレイへの対応を実現する．このようにシステム側の処理で没入型ディスプレイに対応する手法は，ソースコードを最適化して対応した場合と比較すると性能は劣るものの，実用上有効な性能で描画できることが報告されている [3]．

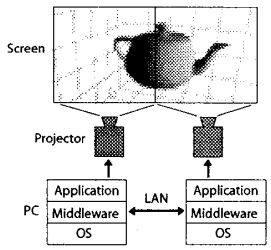


図1: システム構成

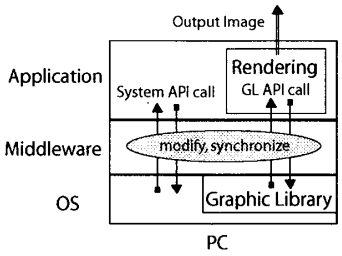


図2: ソフトウェア構造

図1に本システムの構成を示す．各PCにはプロジェクタが接続され，PC間は同期通信のためにLANが接続されている．本システムは，PCクラスタの各PC上にアプリケーションのコピーを配置し同期通信を行いながらアプリケーションを実行する．対象とするアプリケーションとしては，OpenGL等の標準的な3D描画ライブラリで実装されているプログラムを想定しており，多くのアプリケーションに対して適応が可能である．

描画PCのソフトウェア構造を図2に示す．アプリケーションがシステムAPIやグラフィックライブラリのAPIを呼び出すと，ミドルウェアに実行が移り，実際にAPIを実行する前後に没入型ディスプレイに対応するための処理を挿入する．このようにAPI呼び出しに介入する機構は，アプリケーションがAPIを参照するのに用いるインポートテーブルを書き換えることで実現できる．以下，ミドルウェアで行う処理の詳細について述べる．

3.1 PC間のプロセスの同期

各PCで実行されるアプリケーションは，描画に不整合が発生しないために，それぞれ処理が一致していることが必要である．そのためには，プロセスの実行フローや演算結果が全PCで同一になるように操作しなければ

†東京工業大学 精密工学研究所

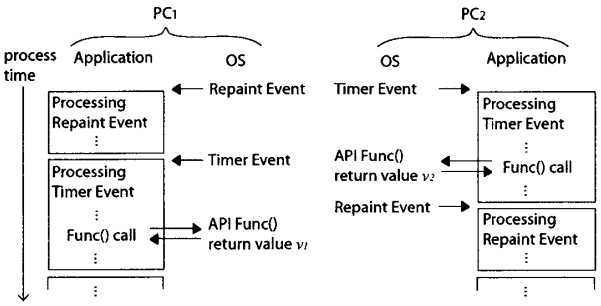


図 3: イベントや API 呼び出しを同期しない場合

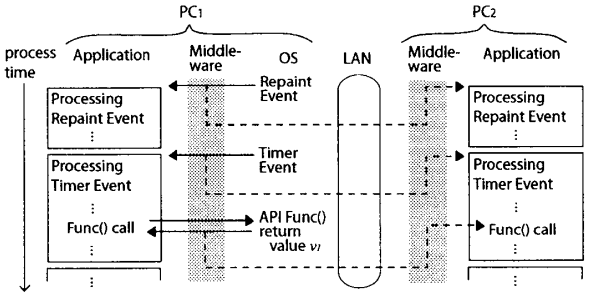


図 4: イベントや API 呼び出しを同期した場合

ならない．複数のスレッドを持つプログラムの場合例外が存在するが，一般に，プログラムの実行フローは実行環境とのやりとり，つまりシステム API の呼び出しや OS が発行するイベントの処理によって一意に決定される．システム API を実行したときの結果はプロセス毎に異なる可能性があり，また，発生したイベントがキューに投入される順序やタイミングは各プロセスで一意ではない (図 3)．そのため，本システムでは，システム API の返値とイベント処理について同期を取ることによって，各プロセスの処理を一致させる (図 4)．

システム API の返値がプロセス毎に異なる例としては，ミリ秒精度のシステム時間や高分解能タイマの値を取得する API が挙げられる．これらの値は，呼び出すタイミングによって差異が生じ，また，実行する PC 毎に値が異なる可能性がある．そこで，ミドルウェアは，これらの API が呼ばれたとき同期処理を行い，全プロセスで値が等しくなるようにする．また，イベント処理に関しては，イベントキューからイベントを取得する API に対して同期を取ることによって，全てのプロセスでイベントの発生を一致させることが可能である．

3.2 没入型ディスプレイのスクリーン構成への対応

没入型ディスプレイのスクリーンは一般に矩形に分割され，各 PC は割り当てられた領域を描画するように構成されている．そのため，正しく没入型ディスプレイ上に表示するには，スクリーン構成に合うように視点や視錐台を PC 毎に設定することが必要である (図 5)．本システムでは，グラフィックライブラリの API 呼び出しに介入して各 PC に適切な視点や視錐台の設定を行う．スクリーン構成に応じたパラメータを設定することで，様々なマルチプロジェクションディスプレイに対してスケーラブルに対応することが可能である．

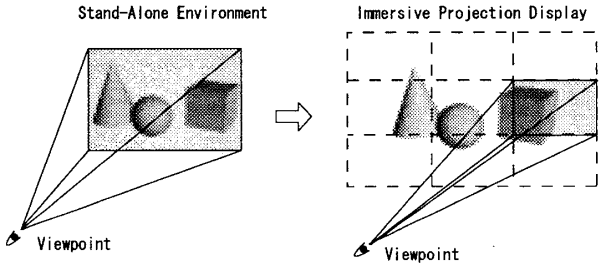
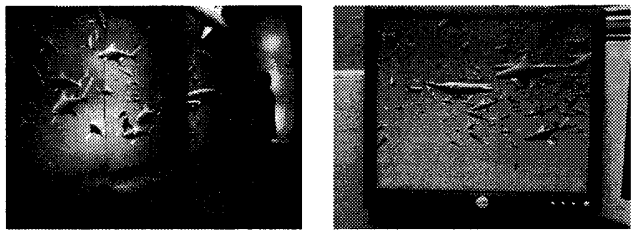


図 5: 視点および視錐台の変換

4. 実装

24 台の PC とプロジェクタで構成される没入型ディスプレイである D-vision において，提案システムを実装した．ネットワーク環境は 100Mbps の Ethernet を利用した．図 6(a) は，本システムを利用して D-vision で OpenGL のデモアプリケーションを実行した例である．同期通信を行なう必要があるのは主に描画関連の API だけで，通信負荷を低く抑えることができ，アプリケーションは実用的な速度で動作した．図 6(b) は同じアプリケーションを本システムを用いずにデスクトップ環境で実行した際の例である．本システムを用いることで同一のアプリケーションを異なるディスプレイ環境で適切に実行できることが確認できた．



(a) D-vision　(b) デスクトップ環境

図 6: アプリケーションの実行例

5. おわりに

本稿では，PC クラスタを用いた没入型ディスプレイに既存のアプリケーションを導入できるソフトウェア環境を提案した．今後は，高更新周期の入力デバイスを用いるアプリケーション等，ネットワーク負荷が高いアプリケーションに対応できるように，通信処理の効率化について検討していきたい．

参考文献

[1] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, J. T. Klosowski: "Chromium: A Stream Processing Framework for Interactive Rendering on Clusters", Proceedings of SIGGRAPH 2002, 2002.

[2] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, C. Cruz-Neira: "VR Juggler: A Virtual Platform for Virtual Reality Application Development", IEEE VR, 2001.

[3] Y. Chen, H. Chen, D. W. Clark, Z. Liu, G. Wallace, K. Li: "Software Envirionments For Cluster-based Display Systems", IEEE/ACM International Symposium on Cluster Computing and Grid, 2001.