情報処理学会論文誌 Vol.54 No.2 551–560 (Feb. 2013)

タイルドディスプレイシステム向け マルチコアレンダリングサーバの設計と実装

峰松 美佳[1,a)] 後藤 真孝[1] 川村 卓也[1] 松澤 茂雄[1]

受付日 2012年3月24日, 採録日 2012年11月2日

概要：同時により多くの情報を表示可能とするタイルドディスプレイシステムが注目を集めている．本論文では，ディスプレイの大型化に柔軟に対応することを目的とした，マルチコアレンダリングサーバの設計と実装について述べる．また，我々の特徴技術である，多段並列処理手法と背景補正ダブルフレームバッファ（FB）の効果を検証する．8 CPU コアサーバを用いて性能評価した結果，フレームレートが，並列処理を行わない場合と比べて 7.3 倍，通常のダブル FB を利用する場合と比べて 1.3 倍向上することが確認できた．また，関連研究である Xvnc と比較すると 5 倍のフレームレートが得られた．

キーワード：タイルドディスプレイシステム，画面転送方式，並列処理

Design and Implementation of Multi-core Rendering Server for Tiled Display Systems

Mika Minematsu[1,a)] Masataka Goto[1] Takuya Kawamura[1]
Shigeo Matsuzawa[1]

Received: March 24, 2012, Accepted: November 2, 2012

Abstract: A tiled display system which enables to display large amount of data is gaining attention. In this paper, we propose multi-core rendering server architecture for a tiled display system which flexibly adapts to different display sizes. The proposed technology introduces multistage parallel processing method and background correction double FB (Frame Buffer). Evaluation using 8 CPU core server showed that the frame rate of the proposed technology increased by 7.3 times compared with single thread version, increased by 1.3 times compared with conventional double FB version, and increased by 5 times compared with Xvnc.

Keywords: tiled display systems, screen transfer technology, parallel processing

1. はじめに

近年，計算機システムの高性能化にともない，扱うデータの大規模化や多様化が進んでいる．同時により多くの情報を表示することで，計算機システム利用者の作業効率を向上させることができるため，表示モニタの大型化の需要は大きい．運用監視業務やデジタルサイネージでは，1 辺が数メートルに及ぶようなモニタを必要とする場合もあるが，大型のモニタは高価なためコストが問題となる場合がある．このため，複数台のモニタをタイル状に配置し，1 つの大きなディスプレイとして扱うタイルドディスプレイシステムが注目を集めている [1], [2], [3].

タイルドディスプレイシステムの例を**図 1** に示す．タイルドディスプレイシステムは，アプリケーションからの描画命令に従い画像情報を生成するレンダリングサーバと，画像情報を各モニタに表示するための表示ノードを必要とする．モニタと表示ノードの台数を増やすことでディスプレイの大型化を実現できるためディスプレイサイズのスケーラビリティが高いという特徴がある．

タイルドディスプレイシステムには，大型ディスプレイとして重要な要求が 3 つある．まず，様々な適用先が考

[1] 株式会社東芝研究開発センター
Corporate Research and Development Center, TOSHIBA Corporation, Kawasaki, Kanagawa 212–8582, Japan
[a)] mika.minematsu@toshiba.co.jp

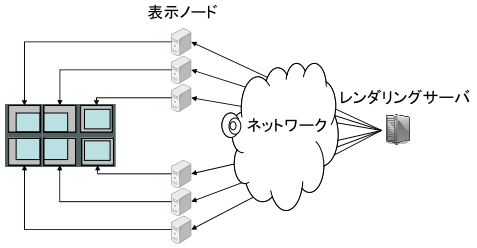


図 1 タイルドディスプレイシステムの例
Fig. 1 An example of a tiled display system.

えられるため，高い汎用性が必要であり，特に既存のアプリケーションを改変せずに利用可能なことが重要である．次に，大型化にともない，モニタ数に比例して増える表示ノードの設置や設定，管理を容易にするために，表示ノードをシンプルに構成可能なことである．さらに，システムとしての表示能力を担う，レンダリングサーバの描画能力や画像情報の出力性能，画像情報の通信性能が重要であり，レンダリングサーバの処理能力を生かした構成とすることが重要である．

これらの要求を受けて，我々は，既存のアプリケーションを改変する必要がなく表示ノードをシンプルに構成可能とした，画面転送方式によるタイルドディスプレイシステムを開発している．本論文では，高い表示能力を実現するためにマルチコアに対応したレンダリングサーバの設計と実装について述べる．

以下，2 章で関連研究について述べる．3 章で設計について述べ，4 章で実装について述べ，5 章で評価について述べる．最後に，6 章でまとめる．

2. 関連研究

タイルドディスプレイを用いた表示システムには，SAGE（Scalable Adaptive Graphics Environment）[4], [5]，SGE（Scalable Graphics Engine）[6]，WireGL [7]，Chromium [8]，DMX（Distributed Multihead X）[9]，Xvnc（X Virtual Network Computing）[10] 等があげられる．

SAGE は，分散レンダリングに対応した SAGE 対応アプリケーションが，SAGE ライブラリを経由して描画命令を出力することにより，複数のレンダリングサーバで描画処理が並列処理されるシステムである．生成された画面情報は，各表示ノードへ適切に分割されてストリーミング転送される．SAGE では，アプリケーションを改変して SAGE API を埋め込む必要があり，システム構成に応じてレンダリングノード数等を指定する，アプリケーション用の設定ファイルを記述する必要もあるため，汎用性が低い．

SGE は，複数のレンダリングサーバが生成した画面情報を 1 台の表示ノードへストリーミング転送し，特殊なハードウェアを用いて 1 つの大きな画面情報として表示するシステムである．SGE では，特殊なハードウェアを必要とするため，汎用性が低い．また，8 枚のビデオカードまでしか対応していないため，スケーラビリティが低い．

次に，WireGL や Chromium は，分散レンダリングに対応した OpenGL アプリケーションが出力する OpenGL 描画命令をキャプチャし，複数のレンダリングサーバに転送することにより，描画処理を並列処理するシステムである．WireGL や Chromium では，OpenGL アプリケーションに特化しているため，汎用性は高くない．また，表示ノードで OpenGL 描画処理を行うため，すべての表示ノードが等しく 3D レンダリングを行うため同一の能力を必要としており，同一の設定も必要となる．

次に，DMX は，X Window System の X サーバ拡張機能モジュールとして提供されている．Frontend サーバは X アプリケーションが出力する X 描画命令をキャプチャし，分割したデスクトップ領域を担当する Backend サーバに適切に転送する．Backend サーバは受信した X 描画命令に従って描画処理を行う．Backend サーバは互いに並列に処理を実行する．DMX は，既存の X アプリケーションを使用できるため汎用性は高い半面，Backend サーバがレンダリングサーバと表示ノードを兼ねており，すべての表示ノードで X アプリケーションからの描画命令を処理できる能力と設定が必要となる．

最後に Xvnc は，X サーバを改変しており，X サーバが描画した画像情報をフレームバッファ（FB）から取得して，圧縮画像として表示ノードに転送する．Xvnc は，DMX と同様に X アプリケーションを改変なしに利用することが可能である．また，画面転送方式を採用しているため，表示ノードをシンプルに構成することが可能である．しかし，Xvnc は，ディスプレイの大型化に対応するように設計されていないため，大型のディスプレイを使用する際には画面転送性能が低いという問題がある．

3. 設計

3.1 設計方針

ディスプレイの大型化に柔軟に対応するためには，既存のアプリケーションを改変せずに利用可能なこと，表示ノードをシンプルに構成可能なこと，レンダリングサーバの処理能力を生かした形で表示能力を実現することが重要である．そこで我々は，X Window System をベースに，レンダリングサーバと表示ノードの間には画面転送方式を用い，マルチコア CPU やマルチ CPU に対応したレンダリングサーバを用いることにする．**図 2** に，提案手法のシステム構成を示す．

まず，X Window System をベースに設計することで，すべての X アプリケーションを改変することなく利用可能となる．また，X Window System という普及したプラットフォーム上で開発を進めることにより，将来にわたり安定性や機能の拡張性を維持したまま性能向上を進めることが

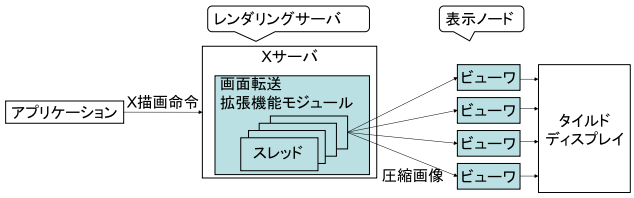


図 2 提案手法のシステム構成
Fig. 2 The architecture of the proposed technology.

できるという利点もある．

次に，レンダリングサーバと表示ノードの間には画面転送方式を用いることで，表示ノードへの要求を緩和する．Chromium や DMX のように描画処理を表示ノードで行う場合は，モニタ数に比例して増えるすべての表示ノードで描画処理が可能であることを保証しなければならない．たとえば X Window System では通常の描画処理以外に，透過処理，3D レンダリング，オフスクリーンバッファへの描画処理等を実現するための様々な拡張描画機能が用意されており，X サーバとなる表示ノードで描画内容によっては負荷の高い描画処理を行う必要があるため，表示するアプリケーションに依存して，そのつど処理性能や対応機能を考慮し直す必要がある．画面転送方式を用いれば，表示ノードへの要求を圧縮画像のデータサイズに見合った通信能力と圧縮画像の展開能力と表示能力のみに限定でき，ハードウェアスペックがより低いノードで構成することが可能となる．

最後に，レンダリングサーバはマルチコア CPU やマルチ CPU の処理能力を生かした形で実現することとする．SAGE，Chromium，DMX のように並列処理のために複数のレンダリングサーバを必要とする場合は，画像情報を複数のレンダリングサーバで描画するため，同じタイミングで表示すべき画像情報がずれて表示されるのを防ぐために，レンダリングサーバ間で同期処理が必要となる．分散システムでのタイミング制御は処理が複雑化しやすく，負荷も発生する．マルチコア CPU やマルチ CPU を搭載したレンダリングサーバを用いて内部の並列処理で性能を確保することで，処理のフェーズの同期が容易となる．また，システムに必要な計算機数を抑えるという利点もある．

3.2 レンダリングサーバ

3.2.1 レンダリングサーバの構成

前述のように，X サーバを用いて画面転送を実現することとした．実現方法としては，X サーバのコア部分を改変して画面転送に対応する方法，X サーバに画面転送に対応した拡張機能モジュールをロードして利用する方法，X サーバとは独立した画面転送アプリケーションを併用する方法が考えられる．X サーバ内部の情報を得ることで効率的な処理が可能なことや，既存の X サーバ上で利用可能なこと，実装コストを考慮した結果，拡張機能モジュールによる実現を採用する．

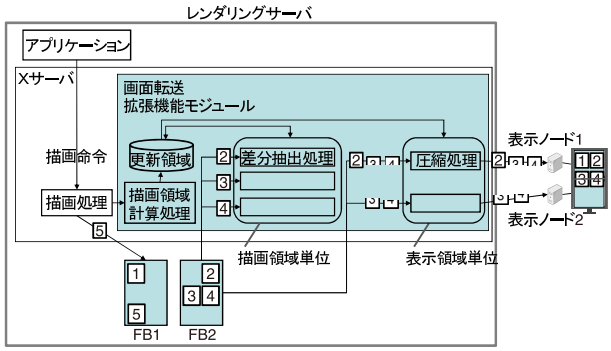


図 3 レンダリングサーバの構成
Fig. 3 The architecture of the rendering server.

本レンダリングサーバの構成を**図 3** に示す．図に示すように，X サーバが FB に描画し，画面転送拡張機能モジュールが FB から画像情報を取得し，表示ノードに転送する．画面転送処理の間も描画処理を可能とするために，ダブル FB 構成とする．具体的には，画面転送拡張機能モジュールは，X サーバの発行する描画命令をフックして，描画領域を計算し，矩形のリストとして更新領域を管理する．画面転送処理は，タイマをトリガに開始され，更新領域から差分領域を抽出し，差分領域に対応する画像情報を圧縮し，圧縮画像を転送する．

ディスプレイサイズを 4200 × 2100（SXGA+ 画面の 6 画面分）とした場合，4200 × 2100 の 24 ビットカラー画像を無圧縮で転送する場合，1 フレームあたり 201.9 Mbit あるため，30 fps で転送するためには，レンダリングサーバで 5.9 Gbps のネットワークが必要となる．タイルドディスプレイ専用のネットワークではなく既設の共用のネットワークを使用する場合は他のノードへ悪影響を与えないようにする必要があること，レンダリングサーバがクラウド上などリモートにある場合は間のネットワーク帯域を保証することが難しいこと，表示ノードの設置を容易にするために無線で構成する場合はネットワーク帯域が狭いことを考慮して，ネットワーク負荷をかけないことを前提とし，圧縮することとした．

レンダリングサーバにおける並列化手法の特徴としては，処理内容ごとに並列数を動的に変更する多段並列処理手法と，X サーバによる描画処理と画面転送拡張機能モジュールによる画面転送処理を効率的に並列処理することを可能とする背景補正ダブル FB 構成をとる．各々について，以降の項で述べる．

3.2.2 多段並列処理手法

多段並列処理手法では，画面転送処理の効率的な並列処理を実現するために，画面転送処理の処理ステップをパイプライン化し，各段の並列数を処理内容に応じて変更する．具体的には，図 3 に示すように，処理負荷が高い差分抽出処理および圧縮処理を並列処理する．差分抽出処理は，更

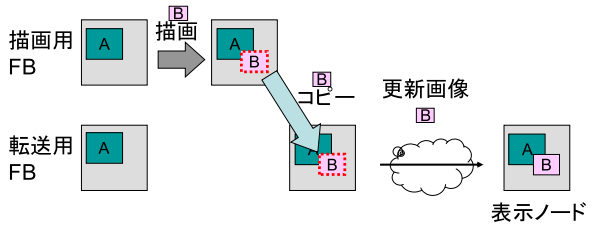


図 4 通常のダブル FB

Fig. 4 Conventional double FB.

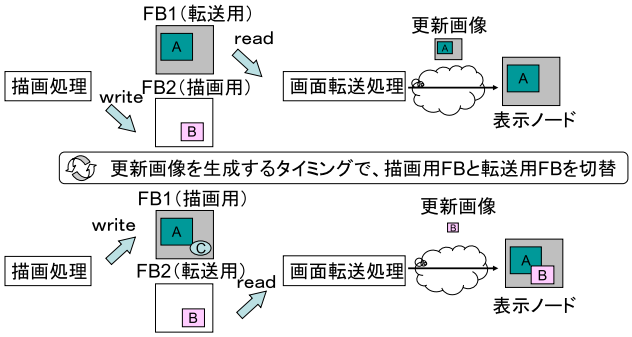


図 5 背景補正ダブル FB

Fig. 5 Background correction double FB.

新領域に含まれる矩形を，CPU コア数に応じて設定された差分抽出スレッド数に等分割して，描画領域ごとに並列処理し，圧縮処理は，圧縮方式は表示ノードごとに異なる可能性があるため，各表示ノードの表示領域ごとに並列処理する．図 3 に示すように，FB1 が描画処理中，FB2 が画面転送処理中であり，FB2 の更新領域として矩形 2，矩形 3，矩形 4 の 3 つの描画領域が含まれ，2 台の表示ノードで表示している例の場合，差分抽出処理は，3 つのスレッドでそれぞれ矩形 2，矩形 3，矩形 4 を並列処理し，圧縮処理は，2 つのスレッドでそれぞれ，図 3 の表示ノード 1 の表示領域に含まれる矩形 2 と矩形 3 の上半分と矩形 4 の上半分，表示ノード 2 の表示領域に含まれる矩形 3 の下半分と矩形 4 の下半分を並列処理することとなる．これにより，処理内容ごとに最適な単位で並列処理を行うことが可能となる．

3.2.3 背景補正ダブル FB

図 4 に示すように，通常のダブル FB では，描画用 FB と転送用 FB が固定されており，描画用 FB から転送用 FB に描画部分をコピーする間は描画処理および画面転送処理をブロックすることとなる．そこで，**図 5** に示すように，更新画像を生成するタイミングで，両 FB を描画用と転送用に交互に切り替え，描画部分の読み出しと描画の並列処理を実現することとした．これにより，描画処理および画面転送処理のブロック期間を短縮できる．また，メモリコピー処理負荷を軽減できる．

しかし，一般に，描画領域は矩形領域として検出されるため，実際には描画が発生していない部分の背景画像が不正確になる場合がある．具体例を**図 6** に示す．図 6 に示

図 6 背景を補正すべき描画の例

Fig. 6 An example of drawings that needs background correction.

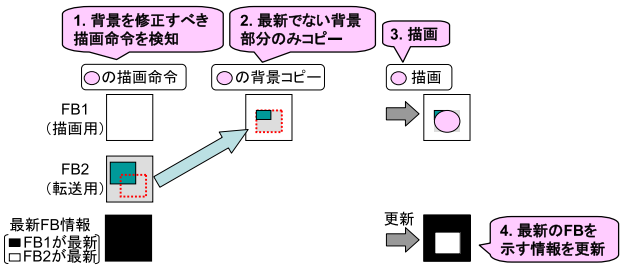


図 7 背景補正処理の流れ

Fig. 7 Background correction process.

すように，円，多角形，文字，点を描く場合等が該当する．この問題を解決するために，背景画像が正しくない可能性のある箇所について，描画の際に，正しい背景をコピーしてから描画することとした．これを**図 7** に図示する．背景補正処理の大まかな流れは以下のとおりである．

(1) 画面転送拡張機能モジュールが X サーバの発行する描画命令をフック
(2) 背景補正すべき描画命令で，背景の画像が最新でない場合，正しい背景をコピー
(3) X サーバによる描画
(4) 最新の FB を示す情報を更新

背景を補正する処理の詳細について，以下で述べる．

まず，背景を補正すべき描画命令を検知すると，描画領域を計算する．たとえば，楕円の描画命令では，長方形の座標および大きさが指定され，該長方形の内接円が描画される．そこで，指定された長方形が (x, y) から幅 w，高さ h，線の幅が lw とすると，描画領域は，$(x - lw/2, y - lw/2)$ から幅 $w + lw$，高さ $h + lw$ となる．また，点の描画命令では，複数の点の座標情報が配列で指定される．そこで，指定された座標情報の中から最少 x 座標，最大 x 座標，最少 y 座標，最大 y 座標を計算し，描画領域は，(最少 x 座標, 最少 y 座標) から幅最大 x 座標 − 最少 x 座標 + lw，高さ最大 y 座標 − 最少 y 座標 + lw となる．

次に，描画領域と転送用 FB が最新な領域の交わる領域を計算する．これが背景補正領域となる．たとえば，FB1 に描画中で，描画領域が $(0, 0)$ から幅 100，高さ 100，FB2 が最新な領域が $(50, 50)$ から幅 100，高さ 100 の場合，背景補正領域は，$(50, 50)$ から幅 50，高さ 50 の領域となる．なお，最新の FB を示す情報は，矩形のリストとして管理する．

最後に，背景補正を行う．たとえば，上述の例の場合は，$(50, 50)$ から幅 50，高さ 50 の領域に対応する画像情報を FB2 から FB1 にコピーする．

3.3 画面転送方式

画面転送方式には，クライアントである表示ノードからのリクエストベースのプロトコルではなく，サーバ主導のストリーミング型の画面更新プロトコルを採用した．これにより，レンダリングサーバで画面転送処理の開始タイミングや更新画像の送信タイミングを制御することができるため，描画内容に応じて画面転送処理を開始することや複数の表示ノードで表示タイミングを一致させるための同期処理を入れることが容易となる．同期表示には，レンダリングサーバでデスクトップ全体の画が1枚の画としてずれないように送信することと，全表示ノードで表示のタイミングを合わせることが必要となる．表示ノードごとにリクエストのタイミングがずれるリクエストベースのプロトコルでは，デスクトップ全体の画がずれないようにするためには，全表示ノードからのリクエストが到着するまでバックオフする等の工夫が必要となる．一方，サーバ主導のプロトコルでは，デスクトップ全体に対する画面転送処理の開始タイミングをサーバが制御するため，同期した画を送信することが相対的に容易である．なお，画面転送処理を開始するタイミングで描画用FBと転送用FBを切り替えることにより，同期した画を保持することを可能としている．また，表示ノードでは，サーバから指定された表示時刻に表示するタイムスタンプ方式 [11] 等の既存の方式を利用可能である．また，レンダリングサーバは，マルチコアに対応しており，内部の並列処理で性能を確保することができるため，1台のレンダリングサーバで処理可能なディスプレイサイズのスケーラビリティが高く，描画処理の並列処理のために複数のレンダリングサーバを必要とする既存研究と比較して，1台のレンダリングサーバでディスプレイ全体の画面情報を処理することにより，大型ディスプレイにおける同期処理も容易となる．なお，モニタ数としては，6〜16程度を想定している．

4. 実装

4.1 レンダリングサーバのソフトウェア構成

3章で述べた設計に基づいた，レンダリングサーバのソフトウェア構成を**図8**に示す．色つきの部分が新規に追加したモジュールである．Xサーバは，OSやハードウェア構成に非依存のDIX（Device Independent X）層，OS依存のOS層，ハードウェア構成に依存するDDX（Device Dependent X）層，Xサーバの機能やXプロトコルを拡張する拡張機能モジュールから構成されている．ネットワーク処理はOS層が担当し，描画処理はDDX層が担当する．そのため，今回実装したビデオドライバはDDX層に含まれる．また，表示ノードで画面情報を表示するビューワは，OS層およびDIX層を介して画面転送拡張機能モジュールとやりとりする．

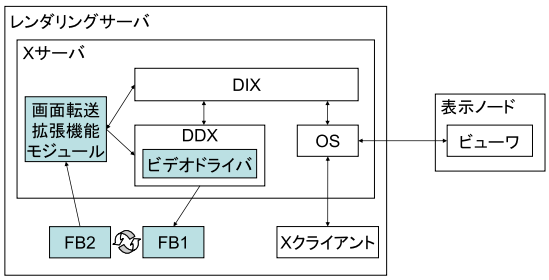


図8 レンダリングサーバのソフトウェア構成
Fig. 8 The software architecture of the rendering server.

4.1.1 ビデオドライバ

ビデオドライバは，仮想的に巨大なサイズのデスクトップを作成するために用いる．ビデオドライバは，既存のdummyビデオドライバをベースとして実装した．具体的には，dummyビデオドライバはメモリが4MB以内かつ2048 × 2048ピクセル以内のデスクトップしかサポートしていなかったため，大型のディスプレイに対応できるように改変した．

4.1.2 画面転送拡張機能モジュール

まず，画面転送拡張機能モジュールにおける大まかな処理の流れを以下に示す．

(1) 随時，Xサーバが発行する描画命令をフックして，描画領域を計算して記録
(2) タイマをトリガとして，画面転送処理を開始
  (ア) 描画領域に対応する画像情報から差分領域を抽出
  (イ) 差分領域に対応する画像情報に必要に応じてフォーマット変換処理および圧縮処理を実施し，更新画像を生成
  (ウ) 更新画像を表示ノードへ転送

画面転送拡張機能モジュールは，マルチスレッドのプログラムとして実装し，メインスレッド以外に，Xサーバに依存しない契機で画面転送処理を開始するためのタイマスレッド，ビューワに送信済みの画像情報として保持しているバックアップ画像とFBの画像情報を比較して差分領域を抽出する差分抽出スレッド，差分領域に対応する画像情報にピクセルフォーマット変換処理や圧縮処理を行い更新画像を生成する更新画像生成スレッドを用意した．このうち，メインスレッドおよびタイマスレッドはつねに割り当てられており，差分抽出スレッドおよび更新画像生成スレッドは動的に割り当てられる．

タイマスレッドでは，30 msごとにタイマを設定し，タイマが切れると画面転送処理を開始する実装とした．なお，タイマが切れた際にすでに画面転送処理が進行中の場合は何もせずに返る．

差分抽出スレッドは，CPUコア数からメインスレッド用に割り当てた1を引いた数を最大スレッド数とし，検出された描画領域数を等分割して各差分抽出スレッドに割り当

表 1 描画命令
Table 1 Drawing methods.

背景補正が必要ない描画命令	背景補正が必要な描画命令	
PutImage	**CopyArea**	**PushPixels**
CopyPlane	**PolyPoint**	**FillSpans**
Polylines	**PolyArc**	**SetSpans**
PolySegment	**FillPolygon**	**ImageGlyphBlt**
PolyRectangle	**PolyFillArc**	**PolyGlyphBlt**
PolyFillRect	**PolyText8**	**CopyWindow**
PaintWindowBackgroud	**PolyText16**	**RestoreAreas**
PaintWindowBorder	**ImageText8**	
ClearToBackground	**ImageText16**	

てる実装とした．また，更新画像生成スレッドは，ビューワごとに割り当てる実装とした．なお，圧縮方式には，コンピュータ画面を想定して，可逆圧縮方式を用いることとし，その中でも比較的圧縮率が良い zlib を採用した．圧縮レベルは，デフォルトの圧縮レベルとした．

表 1 に，X サーバのコア部分で提供されている描画命令を示す．背景が不正確になる可能性がある描画命令を太字で示している．すなわち，太字の描画命令が発行されると，背景補正処理が行われる．

5. 評価

5.1 評価環境

表 2 および**表 3** に，レンダリングサーバおよび表示ノードのスペックを示す．

図 9 に，評価環境を示す．デスクトップサイズは，4200 × 2100 ピクセル（SXGA+ 画面の 6 画面分）とした．レンダリングサーバと表示ノードは，100BASE-TX で接続した．なお，今回はレンダリングサーバ性能を評価することが目的であるため，1 台の表示ノードでディスプレイ全体の画像情報を受信し，表示ノードがボトルネックとならないように，受信した画像情報をそのまま廃棄する描画動作なしビューワプログラムを使用した．

5.2 評価内容

多段並列処理手法および背景補正ダブル FB の効果を測定するために，提案手法において並列処理を行わない「シングルスレッド版」および提案手法において通常のダブル FB を用いる「通常のダブル FB 版」を実装した．また，参考までに，関連研究である Xvnc も評価対象とした．我々の方式では圧縮方式として zlib を採用しているため，Xvnc でも zlib を用いた．**表 4** に，評価対象の基本情報を示す．なお，提案手法においては，カーネルの起動オプションに maxcpus=2 を指定して CPU コア数を 2 に制限した場合も評価対象とした．

描画アプリケーションには，指定した領域を 1 秒間に 30 回単色で塗りつぶすアプリケーションおよび動画再生アプリケーションを利用した．描画内容を**図 10** に示す．図に示すように，4200 × 2100 ピクセルの楕円，4200 × 2100 ピクセルの矩形，1350 × 1000 ピクセルの 6 つの矩形，動画を描画した場合の 4 通りについて評価した．動画再生時の描画領域は，2608 × 1955 ピクセルの動画領域と 4037 × 18 ピクセルのプログレスバー領域で，元の映像ソースは 30 fps で再生されている．

測定は，フレームレート，CPU 使用率，処理時間に対して行い，それぞれ，60 秒間の平均値を求めた．

表 2 レンダリングサーバのスペック
Table 2 Rendering server specifications.

型名	DELL PowerEdge 1950
OS	CentOS 5.2
X サーバ	X.Org X Server 1.5.3
CPU	Quad-Core Intel Xeon 2.66 GHz×2
メモリ	8 GB
NIC	Broadcom NetXtreme II (1 Gbps)

表 3 表示ノードのスペック
Table 3 Display node specifications.

型名	Dynabook SS RX1
OS	Windows Vista Business
CPU	Intel Core Duo U7500 1.06 GHz
メモリ	1.5 GB
NIC	Intel PRO/1000 (1 Gbps)

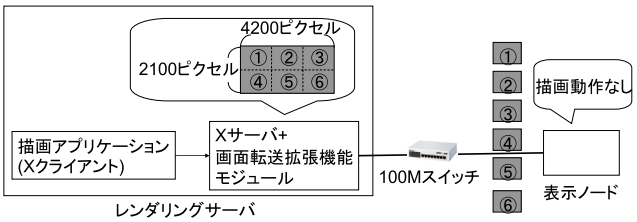


図 9 評価環境
Fig. 9 Evaluation environment.

表 4 評価対象
Table 4 Target of evaluation.

	コア数	スレッド数	FB 構成
提案手法 (8 コア)	8	最大 9	背景補正ダブル FB
提案手法 (2 コア)	2	最大 8	背景補正ダブル FB
通常の ダブル FB 版	8	最大 9	通常のダブル FB
シングル スレッド版	1	1	背景補正ダブル FB
Xvnc (TightVNC)	1	1	シングル FB

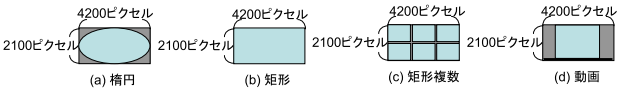


図 10 描画内容
Fig. 10 Drawing contents.

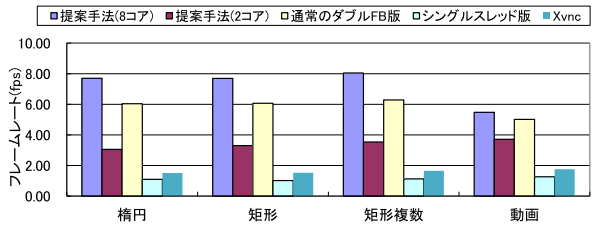


図 11 フレームレート
Fig. 11 Frame rate.

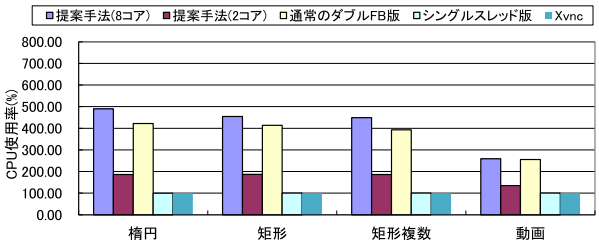


図 12 CPU 使用率
Fig. 12 CPU utilization.

表 5 楕円描画時の処理時間
Table 5 Processing time when oval is drawn.

	ダブル FB 処理(ms)	差分抽出処理(ms)	更新画像生成処理(ms)
提案手法 (8 コア)	30.20	0.15	115.30
提案手法 (2 コア)	30.17	4.80	318.40
通常の ダブル FB 版	26.32	0.17	115.49
シングル スレッド版	1.62	0.04	531.04

5.3 評価結果

5.3.1 フレームレート

図 11 に，フレームレートを示す．縦軸は，フレームレート（単位：fps）を示す．

いずれの描画内容の場合も，提案手法（8 コア），通常のダブル FB 版，提案手法（2 コア），Xvnc，シングルスレッド版の順にフレームレートが高いことが分かった．

5.3.2 CPU 使用率

図 12 に，CPU 使用率を示す．縦軸は，CPU 使用率（単位：%）を示す．

シングルスレッド版と Xvnc では，CPU コアを 1 つしか使用しないため，100%となっている．一方，提案手法と通常のダブル FB 版では，複数の CPU コアを活用している．

5.3.3 処理時間

表 5，**表 6**，**表 7**，**表 8** に，それぞれ，楕円，矩形，6 つの矩形，動画を描画した場合の 1 フレームあたりの処理時間を示す．縦列は，処理時間（単位：ms）を示す．

シングルスレッド版と比べ，提案手法では，更新画像生成処理時間が大幅に短縮されることが分かった．一方，差分抽出処理時間は，動画再生時以外はむしろ若干増加することが分かった．

表 6 矩形描画時の処理時間
Table 6 Processing time when rectangle is drawn.

	ダブル FB 処理(ms)	差分抽出処理(ms)	更新画像生成処理(ms)
提案手法 (8 コア)	0.22	0.15	112.39
提案手法 (2 コア)	0.10	0.40	299.19
通常の ダブル FB 版	26.29	0.16	112.21
シングル スレッド版	0.24	0.02	492.89

表 7 複数矩形描画時の処理時間
Table 7 Processing time when rectangles are drawn.

	ダブル FB 処理(ms)	差分抽出処理(ms)	更新画像生成処理(ms)
提案手法 (8 コア)	0.35	0.17	105.78
提案手法 (2 コア)	0.11	7.55	271.79
通常の ダブル FB 版	26.23	0.17	106.75
シングル スレッド版	0.28	0.02	461.90

表 8 動画再生時の処理時間
Table 8 Processing time when video is drawn.

	ダブル FB 処理(ms)	差分抽出処理(ms)	更新画像生成処理(ms)
提案手法 (8 コア)	0.02	26.70	129.10
提案手法 (2 コア)	0.00	23.68	234.08
通常の ダブル FB 版	17.60	26.58	130.75
シングル スレッド版	0.02	26.74	377.50

表 9 データサイズ
Table 9 Data size.

	無圧縮データサイズ(Mbit)	圧縮後のデータサイズ(Mbit)	通信データ試算(Mbps)
楕円	199.3	0.4	2.8
矩形	199.3	0.2	1.5
6 つの矩形	185.4	0.2	1.5
動画	117.7	5.1	28.2

また，通常のダブル FB 版と比べ，提案手法では，楕円描画時以外はダブル FB 処理時間が大幅に短縮されることが確認できた．なお，ダブル FB 処理時間とは，ダブル FB 間のコピー処理時間を示す．

5.3.4 データサイズ

表 9 に，それぞれ，楕円，矩形，6 つの矩形，動画を描画した場合の圧縮前および圧縮後の 1 フレームあたりの平均データサイズ（単位：Mbit）と，通信データ試算（単位：Mbps）を示す．

5.4 考察

まず，楕円，矩形，6 つの矩形を描画した際の結果について考察する．

図 11 に示すように，フレームレートは，並列化の効果により平均で 7.3 倍，背景補正ダブル FB の効果により平均で 1.3 倍向上することが確認できた．

また，関連研究である Xvnc と比較すると，フレームレートは，平均で 5 倍向上することが分かった．なお，Xvnc の方がシングルスレッド版よりもフレームレートが高い理由としては，TightVNC の実装では差分抽出処理を行っていないためバックアップ画像保存のためのメモリコピー処理負荷がないため，シングル FB のためダブル FB 処理負荷がないため，Xvnc はビューワ主導の画面更新手順を用いているため X サーバのタイマ処理を待たずにビューワからの要求を契機に次の更新画像を送信できるため，等の理由が考えられる．

ここで，ボトルネックについて考察する．

試作では，30 ms ごとのタイマを契機に画面転送処理を開始しており，前述のように，タイマが切れた際にすでに画面転送処理が進行中の場合は何もせずに返るため，タイマが切れた直後に画面転送処理が終了した場合は，30 ms 近く何もせずに待つこととなる．また，更新画像生成スレッドは，ビューワごとに割り当てているため，CPU コア数の方がビューワ数よりも大きい場合は，処理が割り当てられないコアが発生することとなる．そのため，図 12 に示すように，CPU リソースが使いきれていない．今後は，1 フレームあたりの処理時間が長い場合はタイマ値を短くする等，画面転送開始の契機を改善することと，圧縮処理負荷の高いビューワに対しては複数の更新画像生成スレッドを割り当てる等，スレッドの割当て方法を改善することにより，CPU リソースの利用効率の向上が期待できる．

また，表 9 に示すように，通信データ試算により，ネットワークはボトルネックになっていないことを確認した．また，描画内容が単純な程圧縮率が良いことが分かった．なお，動画の場合は，圧縮方式として，視覚特性を考慮した非可逆圧縮方式を用いることにより，大幅にデータサイズが低減される．そのため，領域ごとに色数等を基に適切な圧縮方式を選択することで，通信データ量が低減されることが期待できる．

ここで，提案手法（8 コア）とシングルスレッド版の性能の違いについて考察する．

表 5〜表 7 に示すように，並列化により，処理負荷の高い更新画像生成処理にかかる時間が平均で 77.5%削減できており，フレームレートの向上に寄与している．一方，差分抽出処理にかかる時間はむしろ若干増加している．差分抽出処理では，差分抽出のブロック単位で，横幅ごとにメモリ比較し，1 ビットでも異なるとそのブロックのメモリ比較をやめる．そのため，全画面変化の場合は，差分があることがすぐに判明するため元々処理時間が平均 0.03 ms と短く，マルチスレッド間の競合を避けるためのロック処理によりむしろ処理時間が増加していることが分かった．

また，前述のように，試作では，検出された描画領域ごとに差分抽出スレッドを割り当てる実装となっているため，矩形と 6 つの矩形を描画する場合では，実際に動作する差分抽出スレッド数が 1 と 6 になる．しかし，表 6 および表 7 に示すように，全画面変化の場合は差分抽出処理に時間がかからないため，差分抽出処理の並列化による効果が見られなかった．なお，図 11 に示すように，6 つの矩形を描画する場合の方が楕円や矩形を描画する場合よりもフレームレートが高いのは，更新画像の面積が若干小さく，圧縮時間が面積に比例して短くなっているためである．

なお，今回は，ネットワークに負荷をかけないことを前提として，zlib 圧縮方式と 100 Mbps のネットワークを用いて評価したが，圧縮処理負荷とネットワーク負荷はトレードオフの関係にあるため，ネットワーク帯域が余っている場合は，負荷の低い圧縮アルゴリズムを採用することや無圧縮で送信することにより圧縮処理負荷を低減させることが可能である．そのため，ネットワーク帯域が広い場合でも図 11 に示すような優位性が見られるかを確認することが，今後の課題としてあげられる．

次に，提案手法（8 コア）と通常のダブル FB 版の性能の違いについて考察する．

表 6 および表 7 に示すように，背景補正ダブル FB により，矩形および 6 つの矩形を描画する場合は，ダブル FB 処理時間が平均で 98.9%削減できていることが分かった．なお，ダブル FB 処理時間が 0 でないのは，カーソル描画の際に，カーソル画像の大きさである 117 × 185 ピクセルの領域に対して背景補正処理が発生しているためである．

一方，表 5 に示すように，楕円を描画する場合は，全画面に対して背景補正処理が発生するため，ダブル FB 処理時間が 14.8%増加している．しかし，図 11 に示すように，提案手法（8 コア）において，楕円を描画する場合も矩形を描画する場合も，フレームレートは同程度であることが分かった．一方，図 12 に示すように，提案手法（8 コア）において，楕円を描画する場合は矩形を描画する場合と比べて，CPU 使用率が 16.2%増加していることが分かった．以上の結果から，背景補正処理中も画面転送処理はブロックされないため，処理負荷は増加するがフレームレートには影響がないことが分かった．背景補正ダブル FB 構成では，背景補正処理による負荷により描画内容によっては通常のダブル FB 構成よりも性能が劣化することが懸念されていたが，全画面で背景補正処理が発生する場合でも，通常のダブル FB 構成の場合よりもフレームレートが高いことが確認できた．

なお，表 5 に示すように，楕円を描画する場合に，シングルスレッド版だけダブル FB 処理時間が短いのは，X サー

バによる描画処理，ダブル FB の切替え，画面転送モジュールによる画面転送処理，ダブル FB の切替え，X サーバによる描画処理，の順に処理がスケジューリングされており，片方の FB にのみ描画が発生しているため，背景補正処理が発生していないためであることが分かった．

また，図 12 に示すように，通常のダブル FB 版と比較して提案手法の CPU 使用率が平均で 55%高く利用効率が高いのは，通常のダブル FB 版では描画用 FB から転送用 FB に描画部分をコピーする間は描画処理および画面転送処理がブロックされるためである．

次に，動画を再生した際の結果について考察する．

表 5〜表 8 に示すように，動画再生アプリケーションを用いる場合は，単色で塗りつぶすアプリケーションを用いる場合と比較して，画面変化が少ないために差分抽出処理の負荷が高く，差分抽出処理時間が約 150 倍となっていることが分かった．なお，差分抽出処理の並列化の効果が出ていないのは，前述のように，本試作では，検出された描画領域ごとに差分抽出スレッドを割り当てる実装となっており，差分抽出スレッドが，面積の大きい動画領域に対して 1 スレッドしか割り当てられていないためである．そのため，差分抽出すべき領域を面積に応じて分割して差分抽出スレッドごとの負荷を均等にするなどして，差分抽出処理の並列化の効果を高めることが今後の課題としてあげられる．

なお，図 12 に示すように，動画再生時の CPU リソースの利用効率が他と比べて低いのは，flash 動画を再生するためのプロセスが別途 CPU を 8 コアの場合で 200%，2 コアの場合で 45%程度専有しているためだと考えられる．

ここで，提案手法（8 コア）と提案手法（2 コア）の性能の違いについて考察する．

まず，表 5〜表 8 に示すように，処理時間は，CPU リソースの増加に応じて短縮されることが分かった．具体的には，処理時間の大半を占める更新画像生成処理時間が，平均で 58.8%削減された．なお，提案手法（2 コア）において差分抽出処理時間が長いのは，CPU コア数よりもスレッド数が多いため，待ち時間が発生する場合があるためであることが分かった．

次に，表 5〜表 8 に示すように，背景補正ダブル FB の効果は，コア数に依存しないことが分かった．

次に，図 11 および図 12 に示すように，提案手法（2 コア）の方が CPU リソースの利用効率が高く，コアあたりの性能が高いことが分かった．これは，前述のように，更新画像生成スレッドをビューワごとに割り当てているため，提案手法（8 コア）では処理が割り当てられないコアが発生するためである．

最後に，適用可能なアプリケーションについて述べる．

表 5〜表 8 に示すように，処理時間の大半は圧縮時間が占めており，更新画像の面積が小さい程圧縮時間が短いことから，動きの少ない広告や運用監視業務では，現状でも，利用可能であると予測できる．具体的には，圧縮時間は，前述のように，更新画像の大きさおよび CPU リソース量に依存していることから，更新領域の大きさとフレームレートを掛け合わせた値と，必要な CPU リソース量には，相関関係があると推測できる．我々はモニタ数として 6〜16 程度を想定しているが，16 モニタの場合，ディスプレイ全体の 1〜2 割程度が更新されるような動きの少ない広告や運用監視業務では，評価で使用した 8 コアの PC サーバで，30 fps で利用可能であると考えられる．

一方，広告や運用監視業務でも，動画や監視カメラ映像をディスプレイ全体に表示したい場合もあると考えられる．このような用途では，現状では，元のフレームレートで再生することは困難であると予測できるため，表示ノードで動画圧縮方式に対応し，動画サーバやネットワークカメラから受信した動画ストリームは，レンダリングサーバでは FB に描画せずに，そのまま表示ノードへ転送する等，動画再生に適した方式を別途採用する必要があると考えられる．

6. おわりに

本論文では，既存のアプリケーションを改変する必要がなく，表示ノードをシンプルに構成可能とするために画面転送方式を採用した，マルチコアレンダリングサーバを提案した．また，8 CPU コアのサーバを用いて性能評価を行った．

評価の結果，画面転送処理の処理ステップをパイプライン化し各段の並列数を処理内容に応じて変更する多段並列処理手法と，描画処理および画面転送処理のブロック期間が短くコピー処理負荷が低い背景補正ダブル FB とを特徴とする提案手法において，フレームレートが，並列処理を行わない場合と比べて 7.3 倍，通常のダブル FB を用いる場合と比べて 1.3 倍向上することが確認できた．また，関連研究である Xvnc と比較すると 5 倍のフレームレートが得られた．

背景補正ダブル FB 構成では，画面転送処理のブロック期間が短いことが，フレームレートの向上に寄与していることが分かった．なお，背景補正ダブル FB 構成では，描画内容によっては通常のダブル FB 構成よりも性能が劣化することが懸念されていたが，ダブル FB 処理負荷が高くなる描画内容の場合でも，通常のダブル FB 構成の場合よりもフレームレートが高いことが確認できた．また，多段並列処理手法では，処理負荷の高い更新画像生成処理にかかる時間が短縮できることが，フレームレートの向上に寄与していることが分かった．

今後の課題としては，差分抽出処理および更新画像生成処理の並列化の効果を向上させることと，ネットワーク帯域が広い環境で評価することがあげられる．

謝辞 論文執筆にご協力いただいた皆様に，謹んで感謝の意を表する．

参考文献

[1] 岡崎裕介，須佐雄輝，碓井亮太，荒川 豊，岡本 聡，山中直明：広域分散コンピューティングを用いた遅延を考慮したスケールフリーディスプレイ構築法，電子情報通信学会技術研究報告，Vol.108, No.233, pp.19–24 (2008).

[2] Stolk, B. et al.: Building a 100 Mpixel graphics device for the OptIPuter, *Future Generation Computer Systems*, Vol.22, No.8, pp.972–975 (2006).

[3] Nguyen, H. et al.: Integrating Scientific Workflows and Large Tiled Display Walls: Bridging the Visualization Divide, *Proc. IEEE Conference on Parallel Processing Workshops* (*ICPPW 2011*), pp.308–316 (2011).

[4] Jeong, B. et al.: High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment, *Proc. ACM/IEEE Conference on Supercomputer* (*SC 2006*), pp.24–32 (2006).

[5] Luc, R. et al.: Enabling high resolution collaborative visualization in display rich virtual organizations, *Future Generation Computer Systems*, Vol.25, No.2, pp.161–168 (2009).

[6] Perrine, K. et al.: Parallel graphics and interactivity with the scaleable graphics engine, *Proc. ACM/IEEE Conference on Supercomputing*, pp.3–10 (2001).

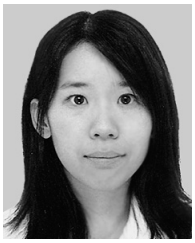
[7] Humphreys, G. et al.: Wiregl: A scalable graphics system for clusters, *Proc. ACM SIGGRAPH 2001*, pp.129–140 (2001).

[8] Humphreys, G. et al.: Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters, *Proc. ACM SIGGRAPH 2002*, pp.693–702 (2002).

[9] DMX Home page, available from ⟨http://dmx.sourceforge.net/⟩.

[10] Xvnc manual page, available from ⟨http://www.realvnc.com/products/free/4.1/man/Xvnc.html⟩.

[11] Qian, R. et al.: Presentation consistency in collaborative teleoperation systems, *Proc. IEEE Conference on Intelligent Robots and Systems* (*IROS 2005*), pp.703–707 (2005).



峰松 美佳

平成 17 年慶應義塾大学大学院政策・メディア研究科修士課程修了．同年（株）東芝入社．研究開発センターにて，リアルタイム通信処理の研究開発に従事．

後藤 真孝 （正会員）

平成 9 年九州大学大学院システム情報科学研究科情報工学専攻修士課程修了．同年（株）東芝入社．オペレーティングシステム，通信プロトコルの研究開発に従事．現在，（株）東芝研究開発センター主任研究員．

川村 卓也

平成 7 年大阪大学大学院工学研究科通信工学専攻修士課程修了．同年（株）東芝入社．通信プロトコルの研究開発に従事．現在，（株）東芝研究開発センター研究企画部企画担当参事．

松澤 茂雄

平成 2 年早稲田大学卒業．同年（株）東芝入社．研究開発センターにて，ネットワークシステムの研究開発に従事．