情報処理学会論文誌 Vol. 52 No. 1 56–67 (Jan. 2011)

Webブラウザによる超高解像度可視化基盤の開発

横山昌平[†1] 石川博[†1]

近年，人類の創出する情報量は急速に増大しており，さらにセンサネットワーク等，莫大な情報を創出する新技術の普及が迫っている．そのような増大し続ける情報の「見える化」には，情報を閲覧するデバイスの高解像度・高精細化が重要である．本論文では複数のモニタを並べて仮想的な高解像度表示領域を得る Tiled Display Wall という手法による，高精細な「見える化」基盤技術を提案する．Tiled Display Wall の従来手法では OS レベルでの簡易な仮想デスクトップか，あるいは高度な分散レンダリングをともなう実装が必要であり，実際的なアプリケーションにおける開発・運用が容易ではなかった．その問題に対し，我々は，Tiled Display Wall のアプリケーションを軽量な Web 技術に基づいて容易に開発する基盤技術を提案する．Web 技術は開発者人口も多く，またすでに多数のサービスが提供されている．本論文では提案手法を用いた応用の例示として，Google Maps API 等既存の Web API を利用したマッシュアップによる超高解像度 Web アプリケーションの実例を紹介する．

Development of an Ultra-high-resolution Visualization Framework Based on Web Browsers

Shohei Yokoyama[†1] and Hiroshi Ishikawa[†1]

In this paper, we describe design and implementation of an ultra-high resolution visualization framework based on Tiled Display Wall. Our proposed system is based only on common Web technologies and uses only low-end consumer products for its components. Therefore the costs of construction, development and management for WDiM are very low and it can provide a method that decomposes high-resolution web applications that are implemented to compose distributed web services and APIs. We also illustrate applications of WDiM. One is for creating photo mosaics, and the other is an example of mashup of google maps API.

1. はじめに

我々人類の創出する情報量は増え続けており，その増加率の高さから情報爆発と呼ばれるまでに成長している．IDC の調査[2] によると 2006 年に生成されたデジタルデータの総量は 161 EB（エクサバイト）であったのに対し，2010 年には 998 EB へと成長する見通しが示されている．またそのデジタルデータの少なくない部分が Web を通じて，交換・共有されている．さらに近い将来には，センサネットワーク等，膨大なデータを創出する新技術の普及も迫っており，この爆発的に増え続けるデジタルデータに対応する処理基盤・利活用基盤のスケーラビリティ確保は重要な問題である．

近年「見える化」というキーワードが注目されている．見える化とは大規模なデータから情報を抽出して人が理解できる形で提示することを指す用語である．なんらかのデータが人間に理解可能な表現で提示されるまでのプロセスを考えると，データの蓄積，処理，閲覧という 3 つのフェーズに大別することができる．それら 3 つのフェーズすべてで，この爆発的に増加し続けているデータの利活用において，高いスケーラビリティを実現することは急務である．

たとえばデータの蓄積においては，古くは RAID により，複数のハードディスクに分散して保存する技術があり，また最近では Google File System[3] に代表されるように，複数のコンピュータにまたがった大規模なデータ蓄積手法が利用されている．これらの技術を使えば単体のハードディスク，あるいはコンピュータが保存できるデータ量を超えた大規模なデータを蓄積することができる．また，クラウド技術により一般の利用者がそれを簡単に利用できる環境も整いつつある．

データの処理に関しても，処理の並列化の手法が様々な粒度で提案されている．たとえば Google によって提唱されている MapReduce のフレームワークは，複数のコンピュータへデータを分割し並列処理を行う仕組みであり，1 つのサーバが処理できるデータ量をはるかに超えた大きなデータに対して，効率的な分散処理を行うことができる．

一方で，データの蓄積と処理に比べ，閲覧に関しては，データの爆発的増加への対応に関する議論は少ない．一般的にデータを閲覧するには，液晶画面等を用いる．いうまでもなく，コンピュータの画面で 1 度に閲覧できる情報の量は，画面の解像度（画素数）に限定さ

†1 静岡大学情報学部
Faculty of Informatics, Shizuoka University

れてしまう．その画面の解像度はここ 15 年で普及機器において XGA（1,024 px × 768 px）から Full HD（1,920 px × 1,080 px）程度にしか増加していない．エンタープライズ向けでは Full HD の 4 倍の解像度を持つ 4 K という規格の製品が開発されているが，それでも，ハードウェアの進歩はデータの増加スピードに比例しているとはいい難い．

対象となるデータ量が爆発的に増加している環境において，解像度がほぼ固定されている画面でのデータ閲覧が前提となるなら，我々が閲覧可能な情報量は，データ総量に相対して時間を追うごとに減少していくと考えることができる．そこで，我々は複数の画面を格子状に並べて，仮想的な高解像度画面を構成する Tiled Display Wall と呼ばれる技術に注目した．これはハードディスクを並べて RAID を構成したり，コンピュータを並べて MapReduce で処理させたりすることに似ている．つまり，安価なデバイスを複数利用して，それを一体運用することにより，スケーラビリティを確保するという手法である．

もちろん現在，多くの PC は，1 台の PC に複数の液晶ディスプレイを接続することができる．この構成の場合，簡単に高解像度画面を構築することができるが，それらすべての画面は 1 台の PC によって描画されることになる．多数の液晶ディスプレイをつなぐ場合，PC，グラフィックカードともにディスプレイの台数に応じた性能の製品を揃える必要がある．また，この手法の最大の問題点は画面の最大台数が OS やグラフィックカードの制約・性能を上回ることができないことにある．Tiled Display Wall を使えば，ハードウェアの制限を超えて，より高解像度の画面を構築することができるようになる．Tiled Display Wall は各 PC 間のネットワークがボトルネックになることを除けば，理論上，解像度の上限はない．

Tiled Display Wall を構築するためのツールとして Xdmx（Distributed Multi-head X server）がある．Xdmx を利用すれば，複数台の PC にまたがった X Window System の仮想的な画面を構築することができる．これにより OS やグラフィックカードの制約にかかわらず，複数台の PC を使い複数台の格子状に並べた液晶ディスプレイに，広大かつ高解像度の仮想デスクトップを構築することができる．この手法の利点は，X Window レベルで仮想化されるため，既存のアプリケーションをそのまま利用できることである．しかしながら，アプリケーション自体のロジックは 1 台のマシン上で動作するため，複数台の PC で構成されたシステムにもかかわらず，並列処理は行えず，サーバの性能がボトルネックとなる．データ量の爆発的増加に対応するスケーラビリティの確保を目指す本研究では，データの表示だけでなく，処理も並列化することが必須要件であり，Xdmx の利用は前提とはならない．また，デジタルデータの多くが Web を通じて交換・共有されている現状に鑑みると，単なる分散描画効率を考慮するだけでなく，既存の Web 技術との親和性を考慮したシステムを構築することが急務である．

そこで我々は，Web 上で交換・共有されている膨大なデータを見える化するための Web 技術に基づいた Tiled Display Wall 構築技術を開発した．本論文では，その実装の詳細を述べるとともに，Web コンテンツの並列描画の定量的に評価する．

本論文の構成は次のとおりである．続く 2 章で本研究で取り組む課題について述べ，また 3 章で提案するシステムの概要を述べる．4 章で本システムを構成するハードウェアに関して，そして 5 章で Tiled Display Wall を駆動するミドルウェアに関して詳述する．さらに実際に我々が実装したアプリケーションを 6 章で紹介し，7 章で提案手法の分散描画性能の評価を行う．さらに，8 章では関連研究について述べ，9 章で本論文をまとめる．

2. 研究の概要

本提案のアイデアは非常にシンプルである．我々は Web 上で共有される大規模データの可視化基盤として Tiled Display Wall を利用する．そして，その Tiled Display Wall の仮想的な超高解像度画面を構成するソフトウェアには Web ブラウザを使う．インターネットが普及し，人類の創出するデジタルデータの少なからぬ部分が Web を通じて交換・共有されている．その Web を閲覧するのに適したソフトウェアはいうまでもなく Web ブラウザである．そこで，Tiled Display Wall を構成する複数台のコンピュータ上で動作する複数の Web ブラウザインスタンスを，ネットワークを介して協調的に動作させ，Tiled Display Wall 全体として 1 つの超高解像度仮想画面を構成する手法を実現した．Web ブラウザが基盤となるため，既存の Web サービスのデータを容易に活用することができるうえ，利用者が有する超高解像度データも，既存の Web サービスの枠組みで共有し，活用することができる．またハードウェアや OS，Web ブラウザの種類に依存しないのも特徴である．Web ブラウザを Tiled Display Wall の描画基盤にするシステムは，我々の知る限り本提案手法が最初の試みである．

提案手法は単に複数台の液晶画面を使うというだけでなく，それぞれの画面が別々のコンピュータにつながる並列・分散協調環境で全体のスループットの向上を目指している．つまり我々は同じ Web コンテンツを複数台で分散描画することにより，1 台で描画するよりも処理時間を短縮させられるだろうという仮説を立てた．そしてフォトモザイクのレンダリングをベンチマークとし，実験で Web ブラウザの並列数による描画速度の変化を計測し，システム全体のスケーラビリティを検証した．その結果，Web ブラウザの並列数に比例して，

画面描画速度が向上することが確認された．

3. システムの概要

本研究は Tiled Display Wall を Web 技術だけで駆動させるミドルウェアの提案である．利用者はこのミドルウェアが提供する API を利用して，利用者データや Web 上のサービスを組み合わせて，Tiled Display Wall 上で動作する利用者自身の高解像度な見える化アプリケーションを開発することができる．

提案するミドルウェアの動作環境を**図 1** に示す．提案システムが前提とするハードウェア構成は，このように複数の PC により駆動される複数の液晶画面を格子状にならべた Tiled Display Wall を基本とする．また提案システムはこれらの Tiled Display Wall を構成する機器のほか，Web アプリケーションをホスティングする Web サーバ機，利用者とのインタラクションを担当する任天堂社の Wii やインターネットに接続された携帯電話・スマートフォンから構成される．

これらの前提を満たすテストベッドとして我々は，Full HD（横 1,920 ピクセル × 縦 1,080 ピクセル）の液晶ディスプレイ 16 面からなる Tiled Display Wall システムを構築した．以後我々の提案システムおよびこのテストベッドを WDiM（Wall Display in Mosaic）と呼ぶ[⋆1]．WDiM を構成する 16 台の液晶画面の背面にはそれぞれ Intel Atom プロセッサを

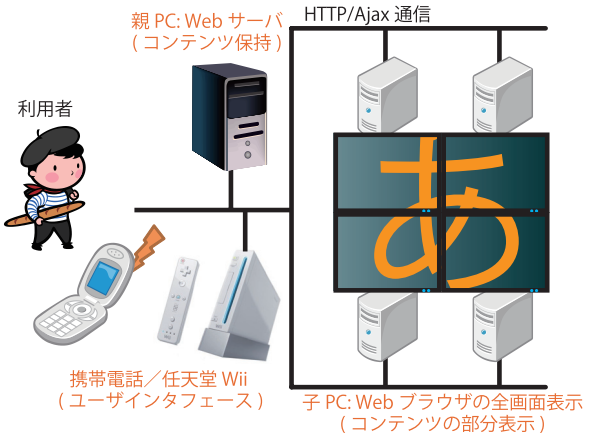


図 1　WDiM の全体像
Fig. 1　Overview of WDiM.

使った PC が接続されている．つまり 16 台の画面を 16 台の PC が駆動する構成である．このほかに，WDiM 上に表示されるコンテンツのユーザインタラクションを担当する任天堂 Wii，WDiM 上に描画するコンテンツを有し，また各画面を仮想的な 1 つの表示領域として同期させるための Web サーバ機を加えてシステムが構成されている．以後，説明のため，各液晶画面を駆動する PC を子 PC，Web サーバ機を親 PC と呼ぶ．

各子 PC，親 PC と Wii は LAN で接続されている．また WDiM はゲートウェイを介して，インターネットに接続しているため，各画面がそれぞれインターネットリソースにアクセスすることができる．つまり WDiM はシステムの内外を問わず，様々なサービス・コンテンツを利用することができる．実際に我々は Google Maps や Flickr といった外部のサービスを WDiM 上で表示するアプリケーションを開発している．また携帯電話等，外部のインターネット接続機器から WDiM へアクセスすることも可能である．

WDiM が従来の Tiled Display Wall と比べて特徴的なのは，画面の描画を Web ブラウザが行っている点である．子 PC はキオスクモードで Web ブラウザを立ち上げている．キオスクモードとは，ツールバーやステータスバー等の装飾がいっさいない全画面表示モードのことで，たとえば Internet Explorer では，コマンドライン引数に *-k* を与えて起動することでキオスクモードで利用できる．なお，子 PC は起動時に自動で Internet Explorer をキオスクモードで立ち上げるように設定するほかは，基本的には「買ってきた状態」の一般的な PC をそのまま利用する．子 PC の数は画面の数に比例するため，Tiled Display Wall の構築コストの重要な部分を占める．そこで WDiM ではこの子 PC 構築に係るコストをできる限り低減することを目指している．

図 2 は一般的な PC ならびに提案手法 WDiM で Google Maps の衛星写真を表示したスクリーンショットである．両者とも同じく東京ディズニーランドの全域を表示しているが，拡大画像を見ると分かるように，その解像度の差は歴然である．このように Tiled Display Wall 技術に基づき Web との親和性の高い提案手法を使えば，既存の Web サービスをそのままに，超高解像度 Web アプリケーションを Tiled Display Wall 上に表示することができる．

次章では，この WDiM の設計について詳述する．

⋆1 WDiM の画像・動画は http://shohei.yokoyama.ac/WDiM で公開している．

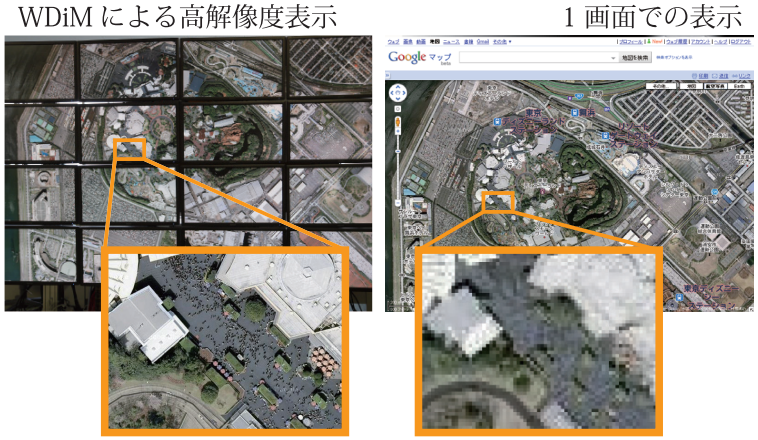


図 2　Wall Display in Mosaic の解像度
Fig. 2　The resolution of Wall Display in Mosaic.

4. WDiM の設計

提案手法を利用して Tiled Display Wall にコンテンツを表示するための必要ソフトウェアは Web ブラウザだけであり，基本的には OS や構成を問わず，幅広い Tiled Display Wall に対応できる．一方で Tiled Display Wall の構成には標準化された手法が存在しないため，様々な構成が提案されている．そこで我々は独自に提案手法との親和性の高い Tiled Display Wall の設計を検討した．本章ではその研究・開発テストベッドとして構築した 16 面の Tiled Display Wall，WDiM の設計について詳述する．

4.1 ハードウェア構成

WDiM は一般的なスペックの安価な機器で構成されている．各画面を駆動する PC（子 PC）は Acer の Aspire Revo である．Aspire Revo は Intel Atom プロセッサ，nVidia ION プラットフォームの製品で，性能的には一般的なコンピュータには劣るが，価格は 1 台 5 万円を下回るため「5 万円 PC」とも呼ばれている．また，性能が低いといっても Web やメールの閲覧には足るという意味から「Nettop」や「Netbook」とも呼ばれている．子 PC の役目は Web ブラウザの動作であり，「Nettop」のターゲットと一致する．

Web サーバ機（親 PC）は PHP の動作環境があれば OS，HTTPD ソフトウェアの種類は問わない．現在は画面の数が 16 枚であり，各画面の同期に必要な通信容量はそれほど多くない．とはいえ，ネットワークとサーバへの負荷は，WDiM を構成する子 PC の数に比例して高まる．しかしながら WDiM の特徴として使われている技術はすべて一般的な Web 技術であり，たとえばサーバの多重化やネットワークの最適化等，スケーラビリティを高めるための製品・ノウハウは数多く存在し，その利用を前提としている．

画面を並べるフレームは汎用のアルミ押し出しフレーム材を用いて，そこに VESA マウント規格の汎用品ディスプレイアームで，ディスプレイを固定している．さらに子 PC の Aspire Revo も VESA マウント規格で固定することができるため，これもディスプレイアームでフレームに固定している．画面の縦 1 列ごとに独立したフレームを用いており，キャスタも付いているので，4 列に分割して運ぶことができる．

このように，WDiM は低価格な汎用品を組み合わせて構築しており，Full HD パネルを 16 枚使った，超高解像度 Tiled Display Wall であるにもかかわらず，100 万円強の価格でこれを実現した．

4.2 ネットワーク構成

ネットワークの物理的な構成は次のとおりである．ネットワークは 1000BASE-T を利用し，親 PC，Wii，すべての子 PC，ゲートウェイ用ルーターが 1 つのハブを介して接続されている．ゲートウェイは上流 LAN を通じてインターネットへ接続している．ゲートウェイは，外部から親 PC の Web サーバへアクセスできるようルーティングテーブルが設定されている．

このネットワーク上で WDiM は，一部起動とシャットダウンに係る通信を除いて，HTTP を用いて通信を行う．このため WDiM の構成上，各画面を駆動する子 PC どうしで直接的な通信ができない．また親 PC から子 PC への通信も不可能である．前者の原因は異なる PC 上で動作する Web ブラウザどうしが直接的に通信できないことに起因しており，後者の原因は Web サーバが Web ブラウザを呼び出すことが原理的にできないことに起因している．これらは Web における一般的な制約であり，WDiM によらずすべての Web アプリケーションはこの制約を受けている．詳しくは 5.2 節で詳述するが，WDiM では子 PC が定期的に親 PC へ Ajax を用いてポーリングすることによって，通信を行う．子 PC から親 PC へは URL の Query String を用いて，親 PC から子 PC へは HTTP リクエストに対するレスポンスとして，データの送信を行っている．この通信の詳細および，子 PC どうしの仮想的な通信の実際の手続きは，WDiM が提供するミドルウェア内に隠蔽される．

4.3 タイル位置関係の定義

WDiM を構成する 16 台の子 PC は，すべて完全に同一の構成であり，ネットワーク・

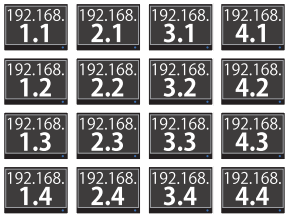


図 3　IP アドレスと画面の位置
Fig. 3　IP addresses and location of displays.

Web ブラウザの設定等も含め異なる点はない．これは WDiM 構築時の作業量低減と，設定情報が 16 台の PC に分散して存在することによる管理コストの高騰を抑えるためである．しかしながら子 PC はそれぞれが仮想画面中のどの範囲を描画するかを把握している必要がある．この設定は個々の子 PC ではなく，ルータ上で集中管理されている．

子 PC は起動時にルータから DHCP を使って自身の IP アドレスを得る．このとき，ルータが持つ特定の MAC アドレスに対し特定 IP のアドレスを割り当てる機能を使って，適切な IP アドレスを子 PC に割り当てる．適切な IP アドレスとは，仮想画面上での位置情報を含んだクラス C のプライベートアドレスである．**図 3** は仮想画面上の位置と割り当てられる IP アドレスの例である．

このように，左上を 192.168.1.1，右下を 192.168.4.4 と割り当てることにより，親 PC は接続元の IP アドレスを調べることで，仮想画面中のどの範囲の描画を担当する子 PC かを把握することができ，その範囲に応じた描画命令を下せる．また IP アドレスの上限，すなわち 192.168.255.254 に至るまで画面の増設に簡単に対応できる．

子 PC は Web ブラウザの立ち上げと同時に同セグメント内にある親 PC の Web ページを HTTP リクエストする．このとき，親 PC は接続元の子 PC の IP アドレスから，その子 PC の仮想画面中の位置を取得し，それを HTTP レスポンスとして，それぞれの子 PC へ返す．親 PC が子 PC へ返す情報は，子 PC の垂直水平位置情報のほか，あらかじめ定義されている仮想画面の幅と高さ，ディスプレイ 1 枚の幅と高さ，ディスプレイのベゼルによって隠される範囲の幅と高さ等がピクセル数換算で渡される．これの仕組みにより，各子 PC は超高解像度コンテンツの描画担当範囲を得る．

その後で改めて表示するコンテンツを親 PC へ問い合わせる．インターネットを通じ外部の親 PC のコンテンツを表示する場合も，位置定義にプライベート IP アドレスを使用しているため，位置取得は同セグメントの親 PC で行い，その後，外部の親 PC へコンテンツを問い合わせる．

コンテンツの描画は，5 章で仕組みを説明し，6 章で例示する．

4.4　起動とシャットダウン

WDiM の通信には HTTP のみを用いると前述したが，起動とシャットダウンだけは例外である．WDiM の起動とシャットダウンは基本的には電源スイッチの操作のみで行うことができるが，16 台ある子 PC すべての電源投入とシャットダウンを手動で行うのは現実的ではない．そこで，WDiM の立ち上げには Wake On Lan（WOL）を使っている．親 PC の起動プロセス時に子 PC へのマジックパケットの送信を行うことにより，自動で子 PC を立ち上げる．

16 台の子 PC はそれぞれ立ち上げと同時に Web ブラウザをキオスクモード（全画面表示）で起動する．この全画面表示された 16 の Web ブラウザ・インスタンスが，提案手法のミドルウェアにより協調して超高解像度コンテンツを描画する．

また，子 PC の立ち上げ時にシャットダウンのための小さなサーバプログラムをバックグラウンドで実行している．このサーバプログラムは特定のポートで，親 PC からのシャットダウンコマンドを待ち受けており，コマンドを受けるとシャットダウンを開始する．

5.　WDiM ミドルウェア

5.1　ミドルウェアの概要

本章では WDiM を駆動し，超高解像度コンテンツを表示しユーザインタラクションを制御するためのミドルウェアの設計について詳述する．本研究で開発したミドルウェアは，親 PC 上で実行される WDiM サーバ，Wii 等クライアント上で実行される WDiM コマンダ，子 PC 上で実行される WDiM レシーバという 3 層モデルを採る．開発者は，この WDiM コマンダ・レシーバを拡張することで，独自の高解像度コンテンツを WDiM 上で提供することができる．またコンテンツは複数定義し，切り替えて表示することができる．ミドルウェアの動作の全体像を**図 4** に示す．

WDiM サーバ，WDiM レシーバはともに PHP プログラムとして実装される．また画面の描画には主に HTML を利用し，その制御は JavaScript によって行われる．チャネルは少なくとも WDiM コマンダ用，レシーバ用の 2 つの PHP プログラムからなり，コンテンツの描画に必要な画像や FLASH 等のリソースを含むことができる．また WDiM は一般的な Web 技術のみで構築されているため，Web API や Web サービス等の外部のリソースへ

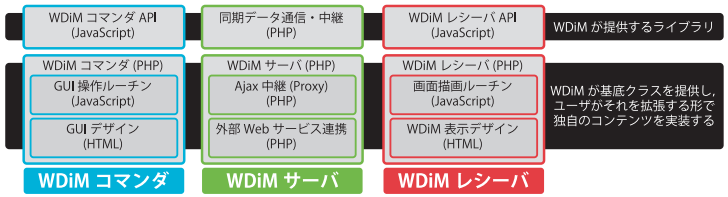


図 4　WDiM ミドルウェアの構成
Fig. 4　Components of WDiM Middle Ware.

アクセスすることも可能である．

WDiM サーバの主な役割は，外部の RESTful Web サービスを呼び出す際の Ajax リクエストの中継地点である．そのほか，親 PC での一元処理を行うルーチン等は WDiM サーバを拡張して実装する．

これらすべてのコンポーネントは，WDiM コマンダで受けたユーザ操作に応じて，WDiM の仮想画面へ表示されているコンテンツを制御するための WDiM コマンダ・レシーバ間通信機能および，各子 PC 間の情報通信・同期機能を持つ．

5.2　通信プロトコルとデータ同期

WDiM コマンダ–WDiM サーバ–WDiM レシーバ間の通信は Ajax による HTTP 通信によって行う．HTTP では Web ブラウザである WDiM コマンダと WDiM レシーバどうしの直接的な通信や，サーバ側から WDiM レシーバ，WDiM コマンダ側を呼び出すことが不可能である．そのため本システムでは WDiM コマンダ，WDiM レシーバが定期的にサーバをポーリングすることにより間接的なデータ通信を実現している．これら通信の手続きは，すべて WDiM が提供する JavaScript ライブラリ内に隠蔽されており，WDiM コマンダとすべてのレシーバ間のメッセージングを，コンテンツ開発者はネットワーク通信を意識せずにプログラミングすることができる．

図 5 に WDiM コマンダ–WDiM レシーバ間でメッセージングを行うプログラム（JavaScript）を示した．WDiM は同期データ，コマンド，定義済みコマンドという 3 種類のメッセージング方式を提供し，開発者は必要に応じてそれらを組み合わせてコンテンツを開発する．

同期データは，WDiM コマンダ–WDiM レシーバ間で同期された単一の値を持つメッセージング方式で，WDiM コマンダで値が更新されるとただちにすべてのレシーバへ通知される．上記コード例では，50 ミリ秒ごとに Wii リモコンの角度を取得し，同期データを更新

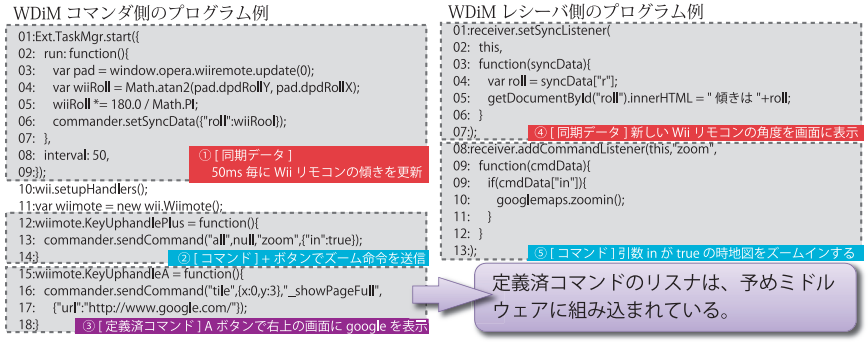


図 5　メッセージ送受信のコード例
Fig. 5　Example code of messaging.

している（図 5 ①）．更新されたデータはただちに WDiM レシーバへ通知され，同期される（図 5 ④）．

コマンドは，ユーザの明示的操作等，イベントの通知に用いられる．WDiM コマンダ上でコマンドが発行されると（図 5 ②），ただちに WDiM レシーバ上の対応するコマンドリスナが呼び出される（図 5 ⑤）．

定義済みコマンド（図 5 ③）は，基本的な操作に関してミドルウェア内であらかじめリスナを実装したもので，詳しくは後述する．

WDiM コマンダ側 13 行目，16 行目の sendCommand 関数は，第 1 引数にコマンドの送信方法，第 2 引数にコマンドの宛先，第 3 引数にコマンド名，第 4 引数にコマンド引数をとる．コマンドの受信はコマンド名を明示的に指定してリスナを登録する（レシーバ側 08 行目）．

次に，実行時における WDiM コマンダ–WDiM レシーバ間でのメッセージングについて述べる．提案システムで通信に利用する HTTP は基本的にはリクエスト・レスポンス型の単方向通信であり，Web ブラウザ間で直接的な通信を行うことはできない．すなわち，図 5 で示した，コマンドや同期データを直接レシーバへ届けることは不可能である．そのため，提案システムは WDiM レシーバと WDiM コマンダがそれぞれ，Web サーバをポーリングし，そのリクエスト・レスポンスへコマンドや同期データを載せている（**図 6**）．現在このメッセージングの送受信プロトコルは Ajax に基づいた非同期通信で行っているが，近い将来 HTML の新しいバージョン HTML5 とともに策定される WebSocket への置き換えを目

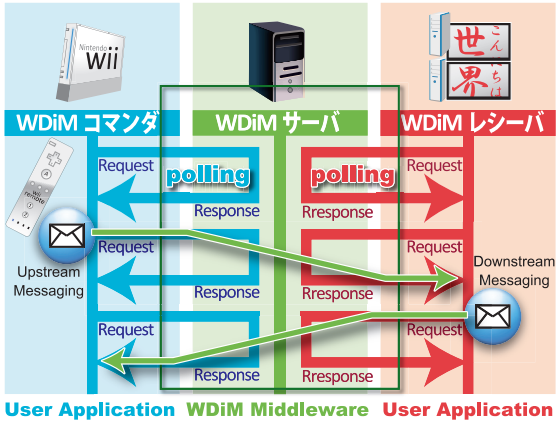


図 6　ユーザ・アプリケーション間のメッセージ伝達
Fig. 6　Messaging between user and application.

表 1　定義済みコマンド
Table 1　Built-in commands.

コマンド ID	コマンド引数	説明
_AppendImage	url, style	画像を表示
_AppendDiv	html, style	任意の内容の DIV 領域を表示
_AppendIFrame	url, style	任意領域に Web ページ表示
_ShowPageFull	url	各画面個別で全画面表示
_ShowPageMega	url	WDiM 全体で全画面表示
_ShowPageTrain	url, direction	長いページを表示

指している．WebSocket を利用すれば，Ajax による頻繁なポーリングを回避できメッセージ伝達のレイテンシを抑え，またサーバ負荷を低減させることができる．

5.3　定義済みコマンド

同期データとコマンドはコンテンツ開発者が自由に定義することができるが，基本的なコマンドは本システムによって提供される．**表 1** は現時点の実装で WDiM が提供するコマンドの一部である．この定義済みコマンドの充実は今後の課題であり，現時点では HTML ページ表示に係る基本的なコマンドのみを有する．

_AppendImage，_AppendDiv，_AppendIFrame タグはそれぞれ，レシーバの画面上に *<img>* タグ，* * タグ，*<iframe>* タグを追加するコマンドである．コマンド引数として，画像はページの URL の指定，あるいは直接 HTML ソースを記述できる．これらのタグの動的な挿入は Ajax を利用したページ遷移のない Web アプリケーションにおいて，非常によく利用される手法であり，提案システムでも定義済みコマンドとしてリスナの実装なしに利用することができる．

_ShowPageFull，_ShowPageMega，_ShowPageTrain コマンドを使えば，WDiM へ一般的な Web ページを表示することができる．たとえば，_ShowPageTrain コマンドは左上から下の方へ，折り返して右下まですべての画面を使って長い Web ページを表示する．これは Wikipedia 等を俯瞰する目的で利用できる．ただし現時点では，子 PC 上に表示されたコンテンツを直接マウスで操作することはできないため，これらの機能はあくまで簡易的な表示を目的としている．複数のマシンをまたがったマウス操作に関しては既存のツールもあるが，本論文ではあくまで Web 技術に基づいた Tiled Display Wall のミドルウェアに注目しており，主題を外れるため，ここでは検討しない．

5.4　画面の描画

子 PC 上での画面の描画は WDiM レシーバが行う．WDiM レシーバは PHP で書かれており，利用者はこれを拡張することでアプリケーションを開発する．ユーザのインタラクションは WDiM コマンダが受け，前述のメッセージングの仕組みにより，任意のあるいはすべての WDiM レシーバへ伝達される．WDiM レシーバには，そのメッセージに応じた処理を定義することができ，この仕組みによりユーザインタラクションが Tiled Display Wall 上に表示されているコンテンツを操作することが可能になる．

すべての子 PC は立ち上がりと同時に Web ブラウザを全画面表示し，WDiM レシーバに定義されている HTML コンテンツを表示する．すべての子 PC は同一の WDiM レシーバを実行しているが，それぞれの子 PC が Tiled Display Wall のどこに位置し，どの部分の描画を担当しているかの情報は JavaScript 変数としてあらかじめ渡されている．この変数をタイル環境変数と呼ぶ．このタイル環境変数とコマンダから送信されたメッセージとを適切に処理することにより，Tiled Display Wall の仮想的な超高解像度画面を描画する．**図 7** にレシーバから利用可能なタイル環境変数を記す．

画面に何を描画するかはアプリケーション次第であり，開発者が自由に実装できるが，分割描画に関しては定型処理も多く，我々はよく利用する描画機能に関してはあらかじめ API として簡単に利用できるよう提供している．たとえば全画面で衛星画像や地図等の GIS データを描画するために Open Geospatial Consortium（OGC）[6] の提唱する Web Map Service（WMS）に対応した．コマンダが表示したい地図の中心座標を送れば，その座標を中心と

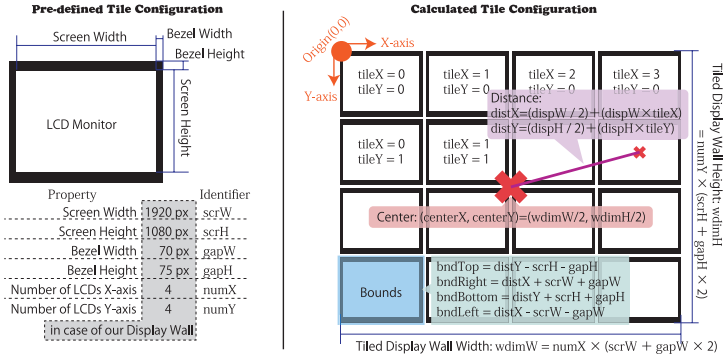


図 7　各 WDiM レシーバへ渡されるタイル環境変数
Fig. 7　The configuration variable of WDiM Receivers.

して子 PC がそれぞれの領域の衛星画像を分散描画する．この手法は衛星画像だけでなく一般的な画像にも適用できる．

Web コンテンツを分割描画する際の最大の問題点は，画像の取扱いである．たとえば 1,000 万画素級のデジタルカメラで撮影した 5 MB の写真を WDiM で表示することを考える．画像は幅 3,680 ピクセル，高さ 2,736 ピクセルで，すべての画素を省略せずに表示するには，Full HD のディスプレイが約 6 台必要になる．このとき，ファイル単位でデータをやりとりすることが前提となる HTTP 通信では，6 台の子 PC がそれぞれ 5 MB の写真を親 PC へ要求することになる．すなわち，5 MB の情報量のデータを可視化するために，ネットワークへ 30 MB のデータが流れることになり，効率的ではない．そこで，WMS ように必要な部分のみを各子 PC が要求し，サーバがそれを切り出して返すことにより，表示範囲外のデータがネットワークに流れることを防いでいる．また Google Maps のように画像を細かくタイリングすることにより，画像のスクロール時も差分データのみの取得が可能となる．

ベクタ画像の描画に関しては，Web ブラウザから利用可能なほぼ唯一のフォーマットである SVG を利用する．ただし現時点ではこの分散描画処理に関しての検討は行っていない．ベクタ画像の Web における分散描画処理は非常に大きな課題であり，本論文の範疇を超える．ただし我々はベクタで表現されている GIS データを表示するために OGC の WFS（Web Feature Service）への適用を検討している．

文字列の分割描画は，現時点では行っておらず，文字列の送信は描画しない範囲まで含めて送信される．これは文字列のフォントや修飾を含んだレンダリングが Web ブラウザに完全に依存しているためであり，WDiM に限らず Web コンテンツは本質的に，文字列がどのように表示されるのかをコンテンツ側でコントロールすることができない．そのため，冗長ではあるが，表示領域が重なるすべての子 PC へ，すべてのデータを重複して送信している．このことによりネットワーク負荷は増加するが，画像に比べて文字列送信に必要なデータ量は僅少であるため，無視できると考える．

動的な図形の描画に関してこれまでは Web ブラウザ上で行うことは困難とされ，Flash や Java アプレット等の外部プログラムがよく利用されてきた．最近では SVG や <*canvas*> タグのサポートが進んだことにより，広く利用できる環境が整いつつある．具体的には現在 Processing.js [7] の利用を試行している．Processing.js は Web ブラウザ上で <*canvas*> 上に図形描画を行う JavaScript のライブラリであり，Flash や Java アプレット等の外部プログラムに頼らず HTML だけでベクトル画像の描画を実現している．

もちろん，WDiM で提供されている API を使わなくてもレシーバ上に Flash や Java アプレットを配置することにより，それらアドオンの機能を利用することが可能である．ただし WDiM は HTML と JavaScript のみを利用した近年のコーディングスタイルを志向しているため，ミドルウェアとしてそれらのアドオンのサポートは行っていない．

このように WDiM を利用したコンテンツは，一般的な Web コンテンツと同様に，ラスタ画像，ベクタ画像，文字列を HTML で記述し，JavaScript で制御することで開発する．

5.5　外部 API の利用

WDiM はローエンドの機器と Web 技術を用いて構築されるため，描画能力・効率の制約は多い．しかしながら Web 技術のみを利用していることにより，既存の Web サービスとの連携が容易であることが利点となる．たとえば Google Maps や Bing Maps 等の地図や衛星画像を表示するサービス，Flickr や YouTube 等の画像・動画共有サービス，じゃらん Web サービス等の観光系 Web サービス等，公開されている膨大なサービスをマッシュアップすることにより，効率的なコンテンツの開発が容易である．

6.　アプリケーション

本章では WDiM の超高解像度画面を利用したアプリケーション例として，外部 API とのマッシュアップによる 2 つのコンテンツを紹介する．このコンテンツは一般ユーザへ高解像度画面の効果を分かりやすくデモンストレーションする目的で開発したものである．

6.1　Google Maps API を利用する応用例

図 8 は Google Maps API を用いた WDiM の応用例「フォトマップ」である．フォト

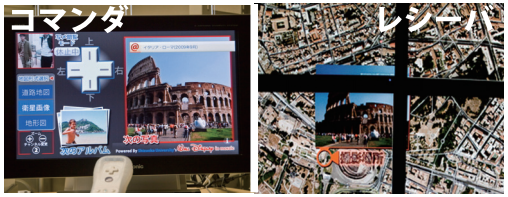


図 8　フォトマップ
Fig. 8　PhotoMap.

マップは EXIF に Geotag（緯度経度情報のメタデータ）を持つ写真ファイルを，その撮影場所の衛星写真とともに WDiM 上に表示するアプリケーションである．写真への Geotag の付与は，現在は携帯電話を中心に普及しており，またデジタルカメラと接続・併用するタイプの機器も安価で販売されている．

Google Maps も Flickr も Geotag の付いた画像を地図上にマップするサービスを提供しており，その機能自体は目新しいものではないが，このアプリケーションで注目すべき点は，16 台の子 PC 上の Web ブラウザが，互いに地図の異なる地点を描画し，全体として，1 つの広大な地図となっているところである．WDiM コマンダである Wii の Wii リモコンを操作することにより，写真を切り替えることができ，地図も写真の切り替わりとともに，次に表示される写真の撮影地点へとスクロールする．また携帯電話で撮影された Geotag 付きの写真をメールでシステムへ送信することにより，その場で WDiM 上へその写真を表示し，またその写真の撮影場所の衛星画像・地図を表示することができる．

このアプリケーションの動作を**図 9** にまとめる．アップロードされた画像から 1 つをユーザが指定することにより，ジオタグとして与えられた撮影場所の緯度経度情報を基に，WDiM 上に高精細な地図を表示する．WDiM の 16 台の子 PC には，地図全体の中心座標のみが与えられる．そして，それぞれの子 PC が，中心からの距離を計算し，画面の物理的な場所に対応した地図を Google Maps に要求し，その画像を表示すると全体として 1 つの高精細な地図として見える．

6.2　Flickr API を利用する応用例

図 10 は Flickr API を用いた WDiM の応用例「フォトモザイク」である．フォトモザイクは高解像度ディスプレイが，単なる大画面とは異なるということを効果的にデモンストレーションするために開発した．

フォトモザイクは，ユーザが与えた任意の画像（元画像）を，多数の小さな画像（タイル

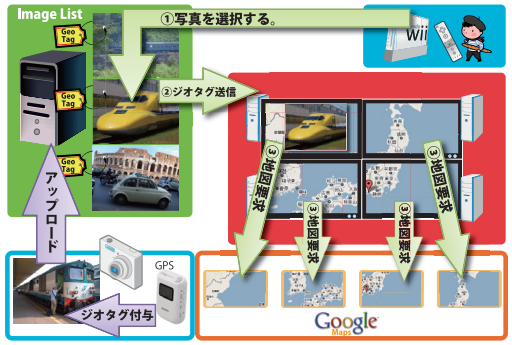


図 9　フォトマップのシステム
Fig. 9　System of PhotoMap.

図 10　フォトモザイク
Fig. 10　PhotoMosaic.

画像）の集合で表現する描画方法である．元画像を細かく分割し，各分割単位（セル）の画像特徴量から，そのセルと近似している画像特徴量を有するタイル画像を検索しその場所にあてはめる，これをすべてのセルにおいて繰り返すことで，モザイク画を生成する．フォトモザイクは遠くから見ると元画像に見えるが，近くによって見るとタイル画像の羅列に見えるという特徴を持ち，商用広告やアートとしてよく利用される表現手段である．

我々は，このフォトモザイクを，WDiM を使い並列処理を行うことにより，高速に生成するアプリケーションを実装した．従来このような画像を生成するためには大規模な画像セットが必要となるが，Web 技術との親和性の高い WDiM の特徴を活かし，Flickr API を使って Flickr の大規模画像リポジトリから，あらかじめ 100 万枚分の特徴量を抽出しそれを利用した．抽出するのは特徴量のみで画像自体は Flickr のサーバからオンデマンドで

図 11　フォトモザイクのシステム
Fig. 11　System of PhotoMosaic.

取得してフォトモザイクを生成する．

図 10 に表れているように，フォトモザイクは全体を俯瞰すれば，WDiM 全体にまたがった大きな画像に見えるが，近くによって見るとたくさんの小さな画像を見ることができる．このように WDiM を使えば，単に画像を拡大して大きく映すのとは異なり，莫大なデータを 1 度に可視化し，それをマクロ的にもミクロ的にも閲覧できる．

このアプリケーションの動作を**図 11** にまとめる．親 PC にはあらかじめ Flickr からクロールした 100 万枚の画像の特徴量が保存されている．今回利用したのは，画像を 9 つの部分に均等に分け，その各部分で色の平均をとってベクトル化している．色空間は RGB に彩度と明度を加えた 5 次元の特徴量を 16 bit で抽出している．すなわち 1 つの画像が 45 次元のベクトルとして表されている．

フォトモザイク生成の処理は次の手続きで行われる．

(1)　利用者が Wi-Fi に接続されたカメラで写真を撮ると，その画像が Flickr へアップロードされる．

(2)　親 PC がこの新たにアップロードされた画像を取得し，タイル画像の大きさに分割する．そして，それぞれの部分においてタイル画像と同様に特徴量抽出を行う．

(3)　特徴量のリストがすべての子 PC の Web ブラウザへ分割送信される．

(4)　各子 PC は，その特徴量リストに基づいて親 PC へ近似画像を要求する．親 PC は kd-tree に基づいた kNN（k 近傍検索）によって近似画像を検索する．この際，kNN サーチは複数並列で行われ，また子 PC も並列に検索リクエストを行う．現在の設定では kNN の検索リクエストを 10 の異なるポートで待ち受け，またマルチスレッドで並列処理を行っている．つまり 16 対 10 の多対多の通信が行われていることになる．これは RFC2616 の制限である「同一サーバへの同時アクセス数上限 2」を回避するとともに，親 PC の CPU リソースを有効活用することが目的である．

(5)　最後に kNN により発見した近似画像リストから直近に送っていないものを選び，そのサムネイル画像の URL を子 PC へ返す．子 PC はその URL に基づいて Flickr のサーバから画像を取得することにより，モザイクのタイルを得る．この処理の繰返しによりフォトモザイクすべてのタイル画像を得る．

次章ではこのアプリケーションを利用して WDiM の性能評価を行う．

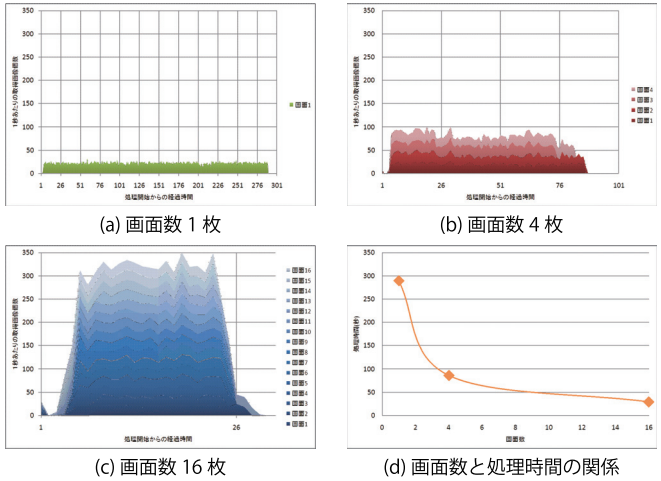
7. 描画性能の評価

本章では，Flickr とのマッシュアップによるフォトモザイク生成アプリケーションを用いて，提案手法の分散描画の性能を検証する．検証に用いた機器は我々の有する WDiM のテストベッド，すなわち 16 台の Full HD 液晶画面を有する Tiled Display Wall である．

本実験では 1 枚の画像が与えられてからフォトモザイク化して画面へ表示するまでの性能を，画面数 1 台の場合，4 台（縦 2 台 × 横 2 台）の場合，16 台（縦 4 台 × 横 4 台）の場合で計測することにより，複数台の PC にまたがった分割処理・分割描画の効率を評価する．フォトモザイク化のアルゴリズムは 6.2 節で説明した手法を用いている．

フォトモザイク化する画像は標準画像の Lenna と Pepper を横に並べたものである．また，フォトモザイクの各タイル画像は Flickr からクロールした約 100 万枚の画像から適切な画像を選択し表示する．描画するフォトモザイクは WDiM の上限である 33 メガピクセルとしている．すなわち，画面台数 1 台，4 台，16 台のいずれの場合も，与えた標準画像に基づいて約 6,000 枚のタイル画像を検索し，適切に並べる処理を行う．この際，16 台以外の場合では，フォトモザイク全体を画面に表示できないが，スクロール可能な領域に描画してすることでシミュレートした．この性能評価実験の結果を**図 12** に示す．

図中 (a) から (c) が描画開始から終了までの 1 秒あたりの取得タイル画像数である．(b)



(a) 画面数 1 枚　(b) 画面数 4 枚

(c) 画面数 16 枚　(d) 画面数と処理時間の関係

図 12　実験結果
Fig. 12　Result.

と (c) は複数の画面による画像取得枚数の積算値を示しており，また各画面の個別の取得数を色分けにして示している．また，図中 (d) が画面数と処理時間の関係を表したものである．

画面数 16 台の場合，秒間 250 枚から 300 枚のタイル画像を kNN 検索により取得し表示できている．それに対し画面数 4 台と 1 台の場合は，それぞれ秒間 80 枚および 25 枚程度のタイル画像しか取得できていない．また，フォトモザイクの描画が完了するまでの時間も，単位時間あたりの取得画像数の比と同じ傾向を見せている．この実験により，提案手法を利用すれば画面数に線形に比例する処理描画を達成できることが分かった．

提案手法を使えば Full HD16 面の Tiled Display Wall において，33 メガピクセルの巨大なフォトモザイクを Flickr から得た 100 万枚のサムネイル画像を使って 30 秒程度で表示できることが示せた．なお，この実験に利用したプログラムは単位時間あたりの画像取得枚数を計測する処理を組み込んだことにより，6.2 節で紹介したアプリケーションに比べ，2 倍程度のフォトモザイク生成時間がかかってしまっている．実際のフォトモザイク生成に係る時間は，カメラで撮影してから 35 秒，システムに画像が届いてから 19 秒ほどで完了する[⋆1]．

8. 関 連 研 究

提案手法を Tiled Display Wall の派生技術として見る場合，関連研究・システムは多数ある．NASA の Hyperwall[9] は横 7 台 × 縦 7 台の SXGA ディスプレイで構成された 64 メガピクセルの Tiled Wall Display である．また前述の LambdaVision は横 11 台，縦 5 台の UXGA ディスプレイを使い 100 メガピクセルの超高解像度ディスプレイを構成しており，それを駆動する SAGE[8] というミドルウェアも提案している．University of California, San Diego の HIPerSpace[1] は，より高解像度なディスプレイを利用することにより総画素数 225 メガピクセルの巨大な Tiled Wall Display を構築している．そのほか，超高解像度コンテンツ可視化に関する研究は Tao らのサーベイ[5] に詳しい．

これら既存研究は，高解像度化，高性能化を目指している．これまでは主に超高解像度コンテンツは，生命科学や地球科学等，一部の科学技術分野に特化した問題として扱われていた．そのため高価な機材と高度な技術により構成されている．しかしながら近年 Web 技術の急速な発展により，Google Maps が超高解像度衛星画像を Web ブラウザを通じて配信しているように，我々は身近に超高解像度画像を扱っている．提案手法は，そのような我々の周りの莫大な情報を，ローエンドの機材と一般的な Web 技術のみで，コストをかけずに可視化する手法として提案している．

高橋ら[11] は Web ブラウザ上で遠隔地にある OS のデスクトップ環境を再現する手法を提案している．これは VDI（Virtual Desktop Infrastructure）と呼ばれる技術を，Web ブラウザ上で実現する試みである．このように Web ブラウザは，OS を代替する基盤技術となりうるまでに重要度を増している．提案手法もまた SAGE 等の Tiled Display Wall 技術を，Web を基盤とした技術へと進化させるという側面を持っている．

提案手法を Web アプリケーションの分散実行環境として見る場合，Vandervelpen らの研究[10] が関連する研究としてあげられる．これらは分散ユーザインタフェース[4] に関する研究分野における課題を Web 技術で実現することを主眼としており，複数の Web ブラウザが協調して動作するという点では類似している．しかしながら，画面を格子状にならべて高解像度コンテンツを分散レンダリングする提案システムとは前提が異なる．

⋆1 19 秒でフォトモザイクを生成する様子は次の動画で見ることができる．http://www.youtube.com/watch?v=swTSMUV3BSA

9. まとめと今後の課題

本論文では「見える化」アプリケーションの基盤となる Web 技術に基づいた超高解像度可視化システム WDiM を提案した．WDiM は Web で公開されている様々なサービスとの連携が容易で，本論文においては Google Maps と Flickr とのマッシュアップを例示した．

今後は定義済みコマンドの整備や，ベクトル画像の効率的な分散レンダリング処理の実装を行う．また，より実際的な応用事例として，現在我々が共同研究として取り組んでいる，大規模なセンサネットワークにおけるセンシングデータの可視化や，地球科学分野における衛星画像の閲覧基盤として利用することを検討している．

参 考 文 献

1) DeFanti, T., Leigh, J., Renambot, L., Jeong, B., Verlo, A., Long, L., Brown, M., Sandin, D., Vishwanath, V., Liu, Q., Katz, M., Papadopoulos, P., Keefe, J., Hidley, G., Dawe, G., Kaufman, I., Glogowski, B., Doerr, K., Singh, R., Girado, J., Schulze, J., Kuester, F. and Smarr, L.: The OptIPortal, a Scalable Visualization, Storage, and Computing Interface Device for the OptiPuter, *Future Generation Computer Systems, The International Journal of Grid Computing and eScience*, Vol.25, No.2, pp.114–123 (2009).
2) Fantz, J.F., Reinsel, D., Chute, C., Schlichting, W., McArthur, J., Minton, S., Xheneti, I., Tonoheva, A. and Manfrediz, A.: The Expanding Digital Universe, *An IDC White Paper – Sponsored by EMC* (2007).
3) Ghemawat, S., Gobioff, H. and Leung, S.-T.: The Google file system, *ACM SIGOPS Operating Systems Review*, Vol.37, No.5, pp.29–43 (2003).
4) Melchior, J., Grolaux, D., Vanderdonckt, J. and Roy, P.V.: A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications, *Proc. ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (*EICS2009*), pp.69–78 (2009).
5) Ni, T., Schmidt, G.S., Staadt, O.G., Livingston, M.A., Ball, R. and May, R.: A Survey on Large High-Resolution Display Technologies, Techniques, and Applications, *Virtual Reality Conference 2006*, pp.223–236 (2006).
6) Open-Geospatial-Consortium. http://www.opengeospatial.org/
7) Processing.js. http://processingjs.org/
8) Renambot, L., Rao, A., Singh, R., Byungil, J., Krishnaprasad, N., Vishwanath, V., Vaidya, C., Nicholas, S., Spale, A., Charles, Z., Gideon, G., Leigh, J. and Johnson, A.: SAGE: The Scalable Adaptive Graphics Environment, *Workshop on Advanced Collaborative Environments* (*WACE'04*) (2004).
9) Sandstrom, T.A., Henze, C. and Levit, C.: The hyperwall, *Proc. International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pp.124–133 (2003).
10) Vandervelpen, C., Vanderhulst, G., Luyten, K. and Coninx, K.: Light-weight Distributed Web Interfaces: Preparing the Web for Heterogeneous Environments, *Proc. 5th International Conference on Web Engineering* (*ICWE2005*), pp.197–202 (2005).
11) 高橋一志，山本知仁：リッチクライアント技術を用いた仮想マシンモニタの提案と実装，情報処理学会システムソフトウェアとオペレーティングシステム研究会 第 19 回コンピュータシステム・シンポジウム，pp.171–172 (2009).

(平成 22 年 4 月 19 日受付)
(平成 22 年 11 月 5 日採録)

横山　昌平（正会員）

静岡大学情報学部助教．産業技術総合研究所を経て 2008 年より現職．2006 年東京都立大学大学院工学研究科修了，博士（工学）．電子情報通信学会，日本データベース学会各正会員．情報処理学会論文誌データベース編集委員（幹事補佐），情報処理学会誌編集委員，電子情報通信学会データ工学研究会幹事．

石川　　博（フェロー）

静岡大学情報学部情報科学科教授．東京大学理学部情報科学科卒業．東京都立大学を経て 2006 年より現職．東京大学博士（理学）．1994 年情報処理学会坂井記念特別賞，1997 年科学技術庁長官賞（研究功績者）受賞．情報処理学会データベースシステム研究会主査，情報処理学会（データベース）共同編集委員長，International Journal Very Large Data Bases Editorial Board，日本データベース学会理事を歴任．ACM，IEEE 各会員．情報処理学会フェロー．電子情報通信学会フェロー．