Vol.2011-HPC-130 No.22
2011/7/28

タイルドディスプレイミドルウェアSAGE用アプリケーション表示アダプタの提案

多田大輝[†1]　市川昊平[†2]
伊達進[†3]　下條真司[†3]

近年，大規模データ可視化技術として，タイルドディスプレイが注目されており，その実現技術として SAGE と呼ばれるミドルウェアがスケーラビリティやパフォーマンスの高さから科学研究への応用に期待されている．しかし，SAGE は既存の可視化アプリケーションをそのままでは利用できないという相互運用性に問題がある．本研究では，この問題に対して，可視化アプリケーションを変更することなく，アプリケーションから可視化データを取得し，SAGE によるタイルドディスプレイに表示可能とする手法を提案し，それを実現する SAGE アダプタの実装を行った．本稿では，この SAGE アダプタについて詳しく述べると共に，既存のアプリケーションを用いた動作検証結果を示す．また，実用上十分な性能が得られることを評価結果と共に報告する．

An SAGE adapter for applications on SAGE-enabled tiled display

Taiki Tada,[†1] Kohei Ichikawa,[†2] Susumu Date[†3] and Shinji Shimojo[†3]

Tiled Display has recently gathered wide attention as a possible way of realizing large-scale visualization. In particular, SAGE is a representative middleware to build up such a tiled display; it is highly expected to be utilized for scientific researches due to its scalable and high performance nature. However, SAGE has an interoperability problem that existing visualization applications are not available without modification of the implementation. To tackle this problem, we proposed SAGE adapter, which allows users to utilize the existing visualization applications in SAGE without modification. This paper describes the details of the SAGE adapter, and introduces an example of an application using SAGE adapter. Moreover the paper reports some evaluation results, which indicate that the SAGE adapter has enough performance for practical use.

1. はじめに

近年の計算機性能の向上，コンピューティング技術の発展により，科学研究において，シミュレーションの大規模化が進んでいる．それに伴い，シミュレーション結果として出力されるデータも大規模化・複雑化しつつあり，データを直感的に理解，分析可能とする可視化技術の要求が高まっている．とりわけ，複数のモニタとそれらに接続された複数の汎用の PC を連携させ，あたかも一つの大きなディスプレイのように使用することで，高精細，高解像度に科学データを表示できるタイルドディスプレイ技術が注目されている．現在，このようなタイルドディスプレイを実現するためのミドルウェアが数多く研究開発されている．[1)–4)]

そのようなタイルドディスプレイを実現するミドルウェアの中でも，とりわけ，イリノイ大学・EVL で研究推進されている SAGE (Scalable Adaptive Graphics Envirnment)[5)] は科学研究への応用に期待が高まっている．SAGE は，遠隔地の計算機で生成された可視化データをストリーミング送信することで，複数の遠隔地にある研究組織間や大学間での情報共有を可能にする．また，各可視化アプリケーションが生成した可視化データは，そのデータを表示すべきモニタを制御する計算機へ直接送信され，タイルドディスプレイを構成する計算機のいずれか一つにデータの表示処理が集中することはない．そのため，SAGE はスケーラビリティがあり，容易にタイルドディスプレイの規模を拡張可能としている．これらの特徴から，SAGE の科学研究への応用例[6),7)] が数多く報告されている．

しかし，SAGE は現在も盛んな研究開発下にあり，既存の可視化アプリケーションとの相互運用性に問題がある．その一例として，SAGE を用いたタイルドディスプレイへ可視化データを表示するためには，SAGE が提供する独自の API を用いなければならないことがあげられる．すなわち，既存の可視化アプリケーションによる可視化データを，SAGE によるタイルドディスプレイに表示するためには，アプリケーション自体をその独自の API を用いて可視化するように変更しなければならない．変更の際に生じる開発コストは大き

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†2 大阪大学情報基盤本部
Central Office for Information Infrastructure, Osaka University
†3 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University

く，科学研究における科学データ分析を妨げる要因となっている．

この問題を解決するため，我々は既存の可視化アプリケーションを変更することなく，その可視化データを取得し，SAGE のインタフェースへ受け渡すことで，タイルドディスプレイへの表示を行う SAGE アダプタの開発を行った．この SAGE アダプタを用いることで，既存のソフトウェア資源を最大限に有効活用した高精細，高解像度の可視化を行うことができ，効率的な科学データ分析を行える環境を実現できる．

以下，本論文では，第 2 節において，SAGE の抱える既存の可視化アプリケーションとの相互運用性の問題を明確にするために，SAGE について概説し，技術的問題を明らかにする．第 3 節では，その技術的問題に対して本研究で提案する SAGE アダプタについて説明する．第 4 節では，SAGE アダプタの実装の詳細について述べる．第 5 節では，提案した SAGE アダプタを既存の可視化アプリケーションに対して適用した結果について報告すると共に，その実用上の性能に関する評価について報告する．最後に，第 6 節にて，本研究のまとめと今後の課題について述べる． s

2. SAGE のアーキテクチャと問題点

SAGE は，科学研究への応用に大きく期待されている一方で，既存のアプリケーションとの相互運用性に問題がある．本節は，まず，SAGE のアーキテクチャを概説し，その後，そのアーキテクチャにより生じる問題点について具体的に述べる．

SAGE の構成を図 1 に示す．SAGE は，Free Space Manager，SAGE Receiver，SAGE UI，SAIL (SAGE Application Interface Library) の 4 つの要素から構成されている．Free Space Manager は管理用ノード上で動作し，タイルドディスプレイ上に表示されているアプリケーションウィンドウの位置を管理し，SAGE を構成する他の要素にそのデータを通知する役割を担う．SAGE Receiver は，各表示ノード上で動作し，可視化アプリケーションからストリーミングで送信されたデータを受信し，そのデータを逐次タイルドディスプレイに表示する役割を担う．SAGE UI は，Free Space Manager と相互通信することで，タイルドディスプレイ上に表示されたアプリケーションウィンドウの移動，および拡大縮小を可能にするユーザインタフェースプログラムである．SAIL は，SAGE が提供している独自の API である．この API を用いて可視化データを送信することで，可視化データが，対象となる表示ノード上の SAGE Receiver へと送信される．

SAIL は，Free Space Manager の管理するウィンドウの位置情報を参照し，データ送信の宛先となる表示ノードを判断し，対象となる表示ノードにのみ直接的にデータ送信を行う

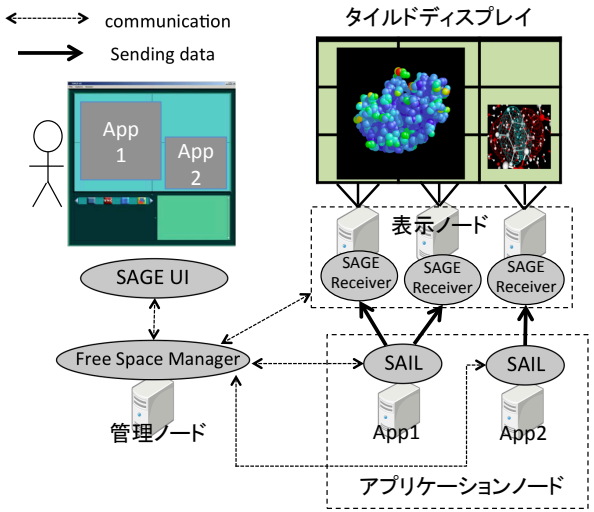

図 1 SAGE の構成.

ように機能する．このような SAIL と Free Space Manager の相互連携により，可視化データの生成とタイルドディスプレイへのデータ表示処理は各可視化アプリケーションごとに分散して行われる．そのため，SAGE は，特定の計算機に可視化データ生成およびその表示処理が集中しないアーキテクチャになっている．このようなアーキテクチャにより，SAGE は，タイルドディスプレイへの高速なデータ表示を実現し，かつ解像度に合わせて柔軟に構成を拡張可能である．

このように，SAGE は，高解像度な可視化データ表示のパフォーマンスとスケーラビリティを重要視したアーキテクチャを取っている．このハイパフォーマンスかつスケーラブルなアーキテクチャを実現する上で，SAIL は重要な役割を果たしている．SAGE アーキテクチャ上では，可視化アプリケーションは，この独自に設計された API である SAIL を利用し，可視化したデータを送信する必要がある．実際に SAIL を用いて可視化データを送信する手順は以下のとおりである．SAIL は可視化データ送信の仕組みとしてダブルバッファ方式を用いており，タイルドディスプレイに今まさに表示しているフレームの画像データを格納するフロントバッファと，可視化データを準備するバックバッファの 2 つのバッファを提供している．アプリケーションが可視化データを送信する際には，まず，バックバッファ

を getBuffer メソッドによって取得する．そして，そのバックバッファに送信したい可視化データを書き込む． 可視化データの書き込みが終了したら，swapBuffer メソッドを呼び，バックバッファとフロントバッファを切り替える．この処理を繰り返すことにより，可視化データをストリームデータとしてタイルドディスプレイに送信する．

しかし，既存の可視化アプリケーションをこの SAIL を用いた実装に変更し，SAGE を用いたタイルドディスプレイへの表示に対応するためのコストは大きい．既存の可視化アプリケーションを SAGE に対応させるためには，まず，アプリケーションのソースコード群から，可視化データの生成に関わる部分を特定する必要がある．そして，生成された可視化データを SAIL が提供するバックバッファへ書き込み，swapBuffer を呼び出す処理を追加する必要がある．ただし，可視化データの生成に関する実装は各可視化アプリケーションごとに独自の実装が行われており，一定した方法はない．したがって，各アプリケーションごとに，このような実装の分析と修正を加える処理が必要となり，開発コストが大きくなる傾向がある．また，一方で，バイナリ形式でのみ提供されているアプリケーションにおいては，そもそもソースコードを変更することができないため，SAGE で利用することは不可能である．このような現状から，科学データの可視化，複数の組織間でのデータ共有を目的とした研究者による SAGE の利用が妨げられている．

3. 提案手法

前節で着目した問題点に対し，本研究では可視化アプリケーションのソースコードを変更することなく，SAGE によるタイルドディスプレイでそのまま表示可能とする，SAGE 用アプリケーション表示アダプタ (以下，SAGE アダプタ) を開発する．本研究で対象とする可視化アプリケーションは，X アプリケーションとする．科学研究に用いられる可視化アプリケーションの多くが，X ウィンドウシステムをベースに開発されている[8)–10)] ことを考慮し，X アプリケーションを対象とする．提案する SAGE アダプタの構成の概要を，図 2 に示す．SAGE アダプタは，X 互換インタフェースをアプリケーションに提供し，そのインタフェースを通じて取得したデータを SAGE に適合するように変換し，SAIL を介してタイルドディスプレイへ可視化データを送信する．

本研究では，SAGE アダプタにおける X 互換インタフェースを提供する目的として，X virtual frame buffer (Xvfb)[11)] を活用した．Xvfb は，グラッフィクスカードに実装されたビデオメモリ上ではなく，計算機に実装されたメインメモリ上にフレームバッファ領域を確保する仮想的な X サーバである．したがって，この仮想的な X サーバに接続して実行された X アプリケーションの描画結果はメインメモリ上に確保されたフレームバッファに格納される．本研究では特に以下の 3 点の理由から，Xvfb を SAGE アダプタにおける X 互換インタフェース部として採用した．1) 実装しているグラッフィクスカードが対応している解像度や，システムが実行している X の設定に制限されることなく，メインメモリサイズを活かして高解像度な可視化データを生成できる点．2) 可視化データをメインメモリから直接的に高速に取得可能とする点．3) メインメモリ上に確保するフレームバッファを UNIX System V 共有メモリインタフェースでアクセス可能な共有メモリとして確保可能な点．これらの特徴により，Xvfb は対象となる X アプリケーションを仮想的に高解像度な X サーバに接続させ，その可視化データを，他のプロセスに対し，高速にアクセス可能な共有メモリとして公開可能としている．

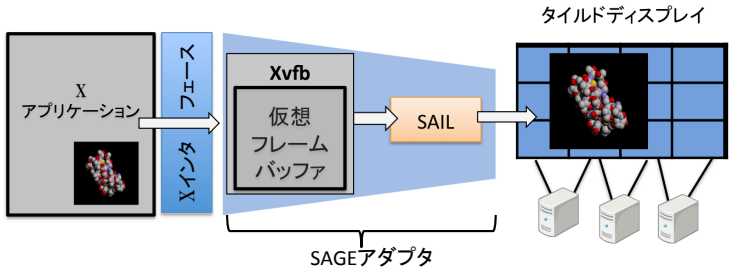

図 2 SAGE アダプタ概要.

本研究で開発する SAGE アダプタは，この Xvfb を利用し，対象となる X アプリケーションの可視化データを共有メモリを介して高速に取得する．SAGE アダプタは取得した可視化データを逐次 SAIL に適合する形式に変換し，SAIL を用いて表示ノードへ送信するという処理を繰り返すことで，可視化アプリケーションの可視化データをタイルドディスプレイにストリーム表示する．この一連の処理の中で，対象となる X アプリケーション自身に変更を加える必要はない．SAGE アダプタは X 互換インタフェース経由で可視化データを受け取り，SAIL に合う形に変換して送信するという，まさにアダプタとしての働きを行う．

```
$ ./SAGEadapter -s 1500x1500 /usr/bin/applicaition <args...>
```

図 3 SAGE アダプタ実行時のコマンド例.

4. SAGE アダプタの実装

以下，SAGE アダプタの実装の概要について述べる．提案する SAGE アダプタは，図 3 に示すコマンドラインインタフェースを提供する．ユーザは，コマンド実行時に，タイルドディスプレイ上でのアプリケーションの表示サイズおよび起動する可視化アプリケーションの実行ファイルへのパスを指定する．このようにコマンドを実行すると，SAGE アダプタが Xvfb と指定されたアプリケーションを起動し，アプリケーションの可視化データを，タイルドディスプレイ上に指定されたサイズで表示する．

SAGE アダプタは，以下に述べる 5 つの処理によって動作する．1) Xvfb の起動．2) 指定された可視化アプリケーションの起動．3) Xvfb によってメインメモリ上のフレームバッファ領域に格納された可視化データを取得． 4) 取得した可視化データを SAIL に適合するデータ形式へと変換． 5) 変換したデータを SAIL を介して表示ノードへ送信．この 3)，4)，5) を高速に繰り返し実行することで，SAGE アダプタは，アプリケーションの可視化データをタイルドディスプレイへストリーミング表示する．

しかし，3)，4)，5) の繰り返しを単純に実装した場合，無駄な可視化データを大量に送信することになり，タイルドディスプレイの表示ノードが接続されているネットワークが高負荷状態になってしまう．これは，SAGE アダプタでは可視化アプリケーション内の詳細な実装を外部からは把握できず，可視化データの更新のタイミングを正確に把握できないためである．可視化アプリケーションのソースコードを直接修正して SAGE に対応させた場合は，可視化データの更新のタイミングを完全に把握した上で，更新された可視化データのみをタイルドディスプレイに送信することができる．一方で，SAGE アダプタでは，可視化データの更新の有無に関わらずデータを送信し続けることになり，ネットワーク帯域を消費し尽くしてしまう．その結果，タイルドディスプレイの表示ノードが接続されているネットワークが高負荷状態になり，パケットロスが発生し，表示ノードで受信した可視化データの一部が表示されないといった問題が生じる．

この問題に対して，本研究では，簡易的な可視化データの更新チェック機能を実装した．具体的には，3) で新規に取得された可視化データが，一巡前に取得された可視化データから

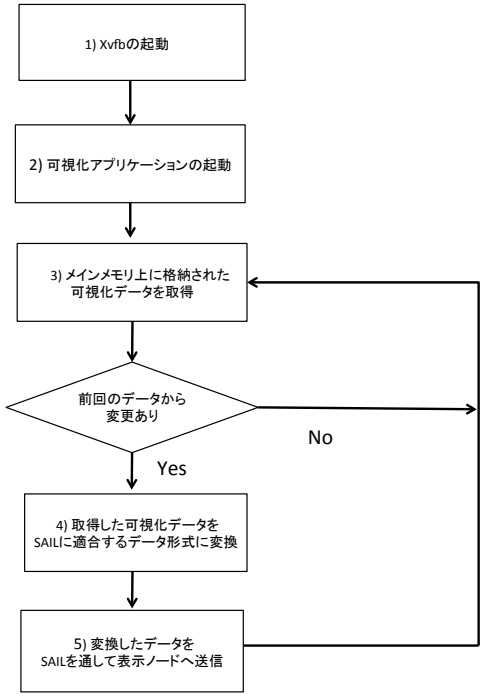

図 4 SAGE アダプタの処理.

変更されていない場合は，表示ノードへのデータ送信を行わないように実装を行った．この処理の流れをフローチャートに表すと，図 4 に示すような流れになる．つまり，更新チェック機能の結果によって 4)，5) の処理を行うかどうかを判別するようになる．この簡易な更新チェック機能では，可視化アプリケーションのソースコードを直接修正する場合と比べると，可視化データを最も適切なタイミングで送信できるわけではない．例えば，可視化アプリケーションが，ある 1 フレームを描いている途中であっても，Xvfb のフレームバッファ上でデータの更新が検知されれば，その描きかけのフレームのデータを送信してしまうため，無駄が生じる場合がある．ただし，科学研究における可視化アプリケーションでは，ある静止したオブジェクトを，ユーザからの拡大・縮小や視点の変更などの操作を受けて，可視化データを更新するものが多い．したがって，ユーザの操作がない間は，表示されている

可視化データに変更がなく，このような簡易な更新チェック機能であっても，送信データの削減に大きな改善が期待できる．

以下，1) から 5) までの各ステップを詳細に説明する．

1) の処理では，まず Xvfb を起動する．Xvfb はメインメモリ上にフレームバッファを確保する仮想的な X サーバであり，a) 起動する仮想的な X サーバのディスプレイ番号，b) 解像度，c) メインメモリ上に確保するフレームバッファのタイプ，のパラメータを指定して起動する必要がある．SAGE アダプタでは，現在未使用の X のディスプレイ番号を利用し，ユーザから指定されたアプリケーションの表示サイズを Xvfb の解像度として指定する．また，メインメモリに確保されたフレームバッファ領域を SAGE アダプタのプロセスから読み出し可能とするため，フレームバッファ領域を共有メモリとして確保するように指定して Xvfb を起動する．Xvfb の起動後，共有メモリへのアクセスに必要な共有メモリ ID は標準出力に出力されるため，SAGE アダプタでは Xvbf の標準出力を読み取り，共有メモリ ID を取得しておく

次に，2) の処理では，対象となる可視化アプリケーションを起動する．ここで，可視化アプリケーションを起動する際にはシステムのデフォルトの X サーバに接続させるのではなく，Xvfb により提供される仮想的な X サーバに接続させる必要がある．SAGE アダプタでは 1) で利用した Xvfb のディスプレイ番号を指定して，対象となる可視化アプリケーションを起動する．こうすることによって，対象となる可視化アプリケーションの可視化データは，Xvfb によって確保されたフレームバッファに格納される．また，SAGE アダプタ起動の際に指定された表示サイズでアプリケーションを可視化するために，X の API を通じて，可視化アプリケーションを Xvfb 内で最大化表示するように操作する．

3) の処理では，1) の Xvfb の起動時に得た共有メモリ ID を元に，Xvfb のフレームバッファ領域にアクセスし，可視化アプリケーションの可視化データを取得する．この 3) に続く更新チェック機能では，一巡前にこのようにフレームバッファより取得されたデータと，今回取得されたデータを比較し，相違がないかチェックする．その結果，相違が見出された場合のみ，以降の 4) 以降の処理に進み，まったく相違が見られない場合は，再び 3) の処理を行うように実装した．

4) の処理では，Xvfb のフレームバッファより取得した可視化データを SAIL に適合するデータ形式へと変換する．SAIL が受け付けるデータ形式は，各ピクセルごとの RGB 値を羅列したビットマップイメージデータそのものである．それに対して，Xvfb を介してフレームバッファ領域より取得できるデータは，Xwd 形式である．Xwd 形式も基本的にはビットマップイメージを含んでいるが，ヘッダ情報なども含んでいるため，直接 SAIL に受け渡すことはできない．具体的には，Xwd 形式は，ヘッダ，カラーマップ，ビットマップイメージデータの 3 つの要素から構成されている．したがって，ビットマップイメージ部分を抽出して，SAIL に受け渡す必要がある．ビットマップイメージデータが格納されているメモリ領域の先頭アドレスの位置は，ヘッダ内に格納される各要素のデータサイズの値から算出することが可能である．この 4) の処理では，算出されたビットマップイメージデータの位置アドレスを次の 5) の処理に受け渡す．

5) の処理では，まず，SAIL が提供する getBuffer メソッドにより，SAIL のバックバッファを取得する．そして，4) で取得したビットマップイメージデータを，このバックバッファへと書き込む．次に，swapBuffer メソッドを呼ぶことでバックバッファとフロントバッファの切り替えを行い，書き込まれたデータをタイルドディスプレイへと送信する．以降は，この 3) から 5) の処理を繰り返し実行することで，可視化アプリケーションの可視化データを表示ノードへ送信する．

5. 動作検証と評価

本節では評価のため，まず，開発したアダプタの動作検証を行った．次に，SAGE アダプタの実用可能性を評価するため，SAGE アダプタの更新チェック機能による送信データの削減の効果と，更新チェック機能によって発生する，データ送信までにかかる時間のオーバヘッドを評価した．

5.1 動作検証

本節では動作検証を目的として，既存の可視化アプリケーションを用いた SAGE アダプタの動作を検証した．具体的には，既存の可視化アプリケーションとして，バイオサイエンス分野において一般的に利用されている 3D 分子閲覧ソフト Rasmol[12] を用いて，SAGE アダプタを介してタイルドディスプレイ上における可視化が可能であることを確認した．

図 5 に，動作検証環境として構築した 4 面のモニタからなるタイルドディスプレイの構成を示す．本動作検証環境は，Free Space Manager を動作させる管理ノード 1 台，各々2 つのモニタを接続した表示用ノード 2 台，アプリケーションを動作させる 1 台のアプリケーションノード，計 4 台の PC からなり，それぞれを 1Gbps のネットワークで接続した環境である．用いた PC のスペックは，Intel Core 2 Duo 6300 1.86GHz，メモリ 2GB であり，OS は，CentOS 5.5 を用い，SAGE のバージョンは，2.5 を用いた．

上記環境において，Rasmol を SAGE アダプタ経由で実行した．実行時に指定したオプ

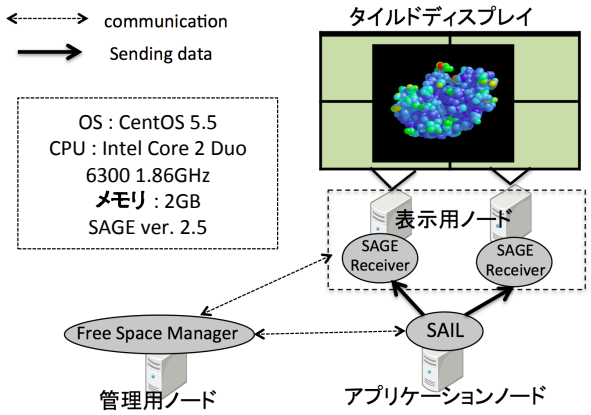

図 5 実験環境.

```
$ ./SAGEadapter -s 2000x2000 /usr/local/bin/rasmol 7lyz.pdb
```

図 6 SAGE アダプタ実行時のコマンド例.

ションを図 6 に示す．表示サイズとして 2000x2000 (ピクセル) を指定し，7lyz.pdb というタンパク質の分子構造データを読み込むように指定している．動作させた様子は図 7 に示す．図 7 に示すように Rasmol の描画結果がタイルドディスプレイに高解像度に表示できることが確認できた．このタイルドディスプレイ上における可視化を行う際には，Rasmol の実行バイナリファイル自身には全く変更を加える必要がなかったことも確認できた．また，Rasmol をマウスなどで操作し，表示されている分子画像を回転や，拡大・縮小行った場合でも，違和感なく表示の更新ができていることを確認できた．

5.2 SAGE アダプタの実用性の評価

本節では，SAGE アダプタの実用性の評価のために，更新チェック機能による送信データの削減の効果と，そのオーバヘッドを評価する．最初に，SAGE アダプタを Rasmol に適用し，更新チェック機能によって，アプリケーションの操作を行っていない期間においては，表示ノードに送信されるパケット量が削減されているか，評価した．次に，更新チェック機能によって生じる，データ送信の処理においてオーバヘッドとなる時間を測定し，評価した．

図 7 Rasmol に SAGE アダプタを適用して動作した様子.

5.2.1 送信データ削減量の評価

送信データ削減量の評価においては，Rasmol に SAGE アダプタを適用し，Rasmol を操作して，表示の回転や拡大・縮小を行っている期間と，操作を行っていない期間に関して，その時の表示ノードへと流れる送信パケット量の変化を測定した．送信パケットの変化の測定には，Ethereal[13] を用いた．図 8 にその結果を示す．

図 8 は，横軸にアプリケーションが起動されてからの経過時間（秒）をとり，縦軸にその時に観測されたパケット数をパケット数/秒で示している．図 8 より，アプリケーションの操作が行われている期間においては 2000 から 3000 パケット/秒のパケットが送信されていることが確認できる．一方で，アプリケーションの操作が行われていない期間においては，送信されるパケットがない．つまり，この結果から，更新チェック機能により，アプリケーションの操作が行われていない期間に関しては，表示ノードへ送信されるパケット量が大きく抑えられていることが確認できる．

5.2.2 オーバーヘッドの評価

SAGE アダプタのオーバヘッドの評価においては，可視化アプリケーションを直接修正して SAGE に対応した場合の処理時間に比較して，SAGE アダプタが追加的に行なっている処理にかかる時間の比を評価する．具体的には，可視化アプリケーションを直接修正して

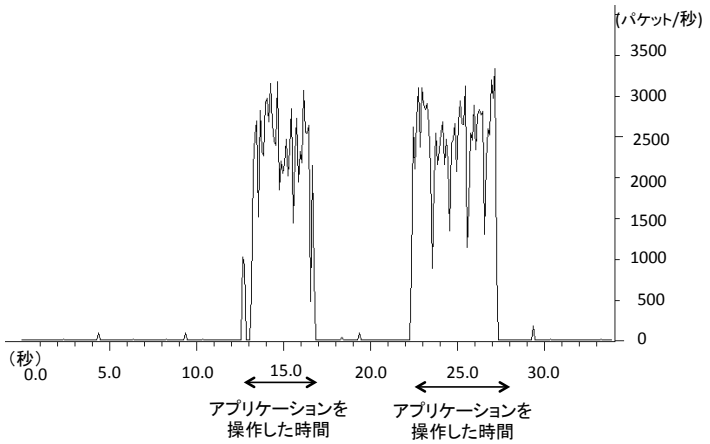

図 8 更新チェック処理による送信パケット削減効果.

SAGE に対応する場合は，1) アプリケーションが生成した可視化データの取得処理，2) 可視化データを SAIL に適合するデータへの変換処理，3) SAIL を介してデータを送信する処理を加える必要がある．SAGE アダプタでは，これらに加えて，4) 前フレームと可視化データの比較を行い，データを送信すべきかどうかを判断する更新チェック処理を行っている．SAGE アダプタでは，この 4) の処理時間分だけ，可視化データの送信時間が余分にかかることになる．本評価では可視化データをタイルドディスプレイに表示するまでにかかる 1) から 3) までの合計処理時間に比べて，SAGE アダプタのオーバヘッドと考えられる 4) の処理時間の比を評価する．

本評価のために，評価用の簡易な X アプリケーションを作成し，これを評価の対象とした．4) の更新チェック機能は可視化データの左上から右下に向かって各ピクセル値を前フレームのデータと比較する．したがって，可視化データの更新が左上に近いところで発生すれば，より短い時間で検出されるのに対し，右下付近で発生する場合は更新チェックにより時間がかかる．そのため，評価対象のアプリケーションの性質によっては 4) の処理時間にバラつきが生じる．評価のために作成した X アプリケーションは，正方形のウィンドウを表示し，毎フレームごとに必ずウィンドウの中央において可視化データの更新を行うように

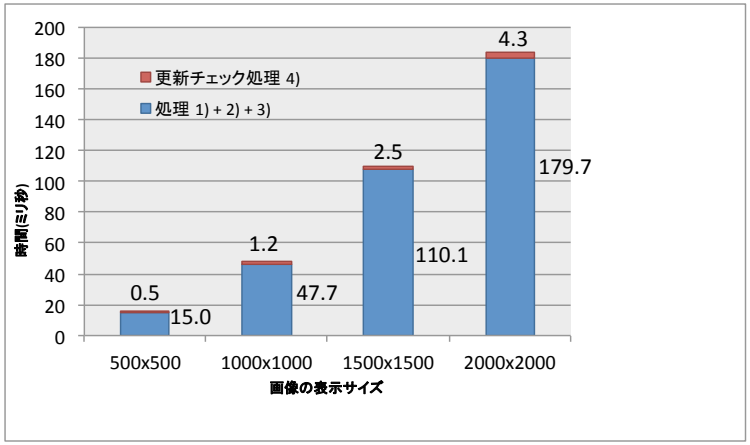

図 9 更新チェック処理にかかる時間の評価.

実装し，このような測定のバラつきを排除した．

評価は，可視化対象のアプリケーションの表示サイズを 500x500, 1000x1000, 1500x1500, 2000x2000 と変えた際のそれぞれにおいて，1) から 3) にかかる処理時間と，4) の処理時間を測定した．図 9 に測定結果を示す．

測定の結果から，更新チェックに有する時間は，いずれの表示サイズでも，1) から 3) の合計時間に対して，5 %以下の時間しか占めていない．したがって，SAGE アダプタの処理によって生じるオーバヘッドは，実用上，十分に小さいことが分かった．

6. ま と め

本研究では，SAGE と既存の可視化アプリケーションとの相互運用性の問題に注目し，既存の可視化アプリケーションに変更を加えることなく，その可視化データを取得し，タイルドディスプレイへの表示を行う SAGE アダプタの開発を行った．本研究では，メインメモリ上にフレームバッファを確保する仮想的な X サーバである Xvfb を活用し，X アプリケーションに変更を加えることなく，その可視化データを取得する方法を提案し，この手法に基づいて SAGE アダプタを実装した．

本研究では，開発した SAGE アダプタを既存の可視化アプリケーションに適用することで，アプリケーションの実行バイナリファイルに全く変更を加えることなく，その可視化

データをタイルドディスプレイに表示できることを確認した．また，SAGE アダプタの実装に組み入れた更新チェック処理により，表示ノードのネットワーク負荷を軽減し，また SAGE アダプタのオーバヘッドも実用上問題ない程度であることを示した．

今後の課題としては，X ウィンドウシステムベース以外のプラットフォームで開発された可視化アプリケーションへの対応が，挙げられる．また，今回の SAGE アダプタでは Xvfb をそのまま使用して，アプリケーションに対する X 互換インタフェースを提供したが，X 互換インターフェースを独自に開発することで，アプリケーションによる描画のタイミングを完全に把握し，より高速なタイルドディスプレイへの可視化データ表示が実現できると考えられる．

参 考 文 献

1) Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M. and Hanrahan, P.: WireGL: a scalable graphics system for clusters, *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, pp.129–140 (2001).
2) : Distributed Multihead X Project, X.org (online), available from ⟨http://dmx.sourceforge.net/⟩ (accessed 2011-01-24).
3) Moreland, K., Avila, L. and Fisk, L.: Parallel unstructured volume rendering in ParaView, *in Proc. of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, the International Society for Optical Engineering*, p.64950F (2007).
4) Ni, T., Schmidt, G., Staadt, O., Livingston, M., Ball, R. and May, R.: A survey of large high-resolution display technologies, techniques, and applications, *Virtual Reality Conference, 2006*, IEEE, pp.223–236 (2006).
5) Renambot, L., Rao, A., Singh, R., Jeong, B., Krishnaprasad, N., Vishwanath, V., Chandrasekhar, V., Schwarz, N., Spale, A., Zhang, C. et al.: Sage: the scalable adaptive graphics environment, *Proceedings of WACE 2004* (2004).
6) Renambot, L., Jeong, B., Hur, H., Johnson, A. and Leigh, J.: Enabling high resolution collaborative visualization in display rich virtual organizations, *Future Generation Computer Systems*, Vol.25, No.2, pp.161–168 (2009).
7) DeFanti, T., Leigh, J., Renambot, L., Jeong, B., Verlo, A., Long, L., Brown, M., Sandin, D., Vishwanath, V., Liu, Q. et al.: The OptIPortal, a scalable visualization, storage, and computing interface device for the OptiPuter, *Future Generation Computer Systems*, Vol.25, No.2, pp.114–123 (2009).
8) Abram, G.: OpenDX Overview, *Visualization Development Environments*, pp.27–28 (2000).
9) Ramachandran, P. and Varoquaux, G.: Mayavi: Making 3D data visualization reusable, *Proceedings of the 7th Python in Science conference (SciPy 2008) http://code. enthought. com/projects/mayavi*, pp.51–56.
10) Cotter, C. and Gorman, G.: Diagnostic tools for 3D unstructured oceanographic data, *Ocean Modelling*, Vol.20, No.2, pp.170–182 (2008).
11) : XVFB manual page, X.org (online), available from ⟨http://www.x.org/archive/X11R6.8.1/doc/Xvfb.1.html⟩ (accessed 2011-01-24).
12) Sayle, R. and Milner-White, E.: RASMOL: biomolecular graphics for all., *Trends in biochemical sciences*, Vol.20, No.9, p.374 (1995).
13) : Ethereal, Ethereal, Inc (online), available from ⟨http://www.ethereal.com/⟩ (accessed 2011-01-24).