

SAGEにおけるアプリケーション操作インタフェースの拡張

桑原 世輝^{††} 市川 晃平^{††} 伊達 進^{††} 下條 真司^{§†}

大規模な科学データを高解像度に可視化する技術として SAGE が注目を集めている。この SAGE は、遠隔地にある大規模なデータをネットワークストリーミング技術により、ローカルサイトの高解像度表示装置に表示する技術である。今日、このような SAGE の有する特徴から、複数の研究機関の研究者らが大規模データの共有・可視化を通じて協調して研究を進めることのできる協調環境の構築を進める動きがある。しかし、そのような高まる期待にも関わらず、このような協調研究環境を構築しようとする際、SAGE には様々な技術的課題が残されているのが現状である。本論文では、著者らの SAGE の利用経験からこれらの技術的課題の整理・概説を行い、今後の研究課題について明確にする。そのあとで、それらの課題の 1 つである SAGE 利用時におけるアプリケーション操作およびウィンドウ操作における技術的課題に言及し、その解決策について示す。

An Application Control Interface for SAGE

Seiki Kuwabara^{††} Kohei Ichikawa^{††} Susumu Date^{††} Shinji Shimojo^{§†}

SAGE has been gathering researchers' attention as a visualization technology that enables enormous scientific data in a high-resolution manner. This SAGE is the technology that displays enormous scientific data on a local high-resolution display device through the network streaming of the scientific data from a remote site. Today, this characteristic of SAGE is making the development of collaboration environment where researchers who perform their own scientific research through the sharing and visualization of scientific data active. In this paper, we first summarize and briefly explain these technical issues around SAGE and then clarify the research issues from now on. Subsequently, the interface problem related to window and application operations, which is one of such technical issues, is mentioned and its solution is shown in this paper.

1 はじめに

近年の計測技術の向上やハイパフォーマンスコンピューティング技術の発展は、科学者が解析しなければならないデータを大規模化、複雑化している。実際、CT や MRI などの医療用測定装置を例にとってみても、その精度は飛躍的に向上してきている。また、我々が日常的に利用する一般的なデジタルカメラでさえ、その撮影精度は一千万画素級に達している。また、Grid [1] に代表されるハイパフォーマンスコンピューティング技術の発展は、広域な計算環境の構築を実現し、大規模なシミュレーションや解析計算の実行を可能としている。その結果、科学者は今日、より膨大かつ複雑な解析結果やシミュレーション結果を得られるようになっている。このようにデータの大規模化、複雑化が進んでいる今日において、そのデータの意味を直感的に表現できる表示技術の重要性がますます高まっている [2]。

Tiled Display Wall (TDW) は高解像度な画像や映像を表示可能にする装置である。TDW は複数のディスプレイとそれに接続される複数の計算機によって構成されており、ディスプレイを図 1 で示すようにタイル状に並べることによって高解像度表示を実現する。各ディスプレイに接続されている計算機は、他の計算機と画像や映像の表示タイミングの同期、および表示ウィンドウの配置やサイズを連携させ、複数のディスプレイ間で表示される画像や映像を制御している。この TDW を構成する各計算機を制御し、データの表示を行うミドルウェアとして SAGE (Scalable Adaptive Graphics Environment) [3] が注目を集めている。



図 1 大阪大学に設置された Tiled Display Wall

[†] 大阪大学大学院情報科学研究科

Graduate School of Information Science and Technology, Osaka University

^{††} 独立行政法人情報通信研究機構

National Institute of Information and Communications Technology

[§] 大阪大学サイバーメディアセンター

Cybermedia Center, Osaka University

SAGE は米国イリノイ大学の EVL (Electronic Visualization Laboratory) で研究開発が行われているミドルウェアである。この SAGE の特徴として、遠隔地で生成された高解像度かつ高精細な可視化結果をネットワークを介して TDW に転送し、表示可能な点があげられる。つまり、地理的に離れた研究者間でデータの共有・可視化を可能とし、協調して科学研究を実施できる環境を構築することが可能である。このような理由から現在、多くの研究機関や大学がバイオサイエンスやジオサイエンス等の科学分野において、SAGE による TDW を用いたデータ共有・可視化環境の構築、その利用が試みられている。しかし、SAGE は今日においても開発途上の技術であり、実際に上述した科学分野において、その研究現場で利用・運用するためには技術的課題が多く残されている。本論文では、広域分散環境下において SAGE を用いて実用的な協調研究環境を構築する上で問題となる技術的課題の整理を行う。また、その課題のうちアプリケーションの操作性の問題について言及し、その解決方法に関して議論する。

以下、第 2 章では SAGE のアーキテクチャおよび特徴とその動作を説明する。第 3 章では SAGE を用いた協調研究環境を構築する上での問題点をあげ、整理する。第 4 章では問題点のうちの一つである、アプリケーションの操作性の問題の解決に向けて構築したアプリケーションコントロールモジュールについて述べる。最後に、第 5 章で本論文をまとめる。

2 SAGE のアーキテクチャと動作

SAGE は TDW における表示環境の構築を支援するためのミドルウェアである。SAGE はネットワークを介し、ローカルやリモート環境で実行されたアプリケーションによる可視化結果をストリーミング表示することができる。

図 2 に実際に SAGE を用いた表示環境の例を示す。SAGE は次の 4 つの主な要素から成り立っている。Free Space Manager, SAGE Application Interface Library (SAIL), SAGE Receiver, SAGE UI である。図中 (a) はディスプレイを 5×4 のタイル状に配置した TDW である。(b) は TDW に接続されているディスプレイ描画ノードであり、ディスプレイ描画ノード上ではそれぞれに SAGE Receiver というプログラムが動作する。(c) は管理ノードであり、SAGE の管理・制御を行う Free Space Manager が動作する。(d), (e) はアプリケーションが実行されるレンダリングサーバで、ここで SAGE の API である SAIL とよばれるライブラリを使用したアプリケーションが動作する。(f), (g) はユーザ操作端末であり、図 3 に示される SAGE UI が動作する。

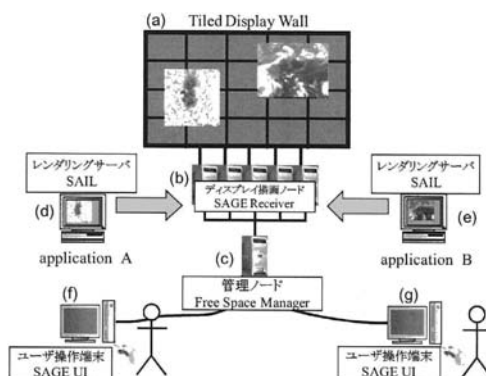


図 2 SAGE を利用した可視化環境

SAGE を用いた画像や映像の表示の流れは次の 3 ステップで行われる。

1. (d), (e) で動作するアプリケーションにリンクされている SAIL が Free Space Manager からウィンドウをどの位置に表示するべきかに関する情報を得る。
2. アプリケーションから受け取った可視化結果を適切な SAGE Receiver(b) にストリーミング送信する。
3. SAGE Receiver は TDW を構成するディスプレイに接続された各計算機で動作するプログラムであり、受け取ったストリーミングデータを各々 (a) の TDW に表示する。

Free Space Manager は SAGE の中心となるプログラムであり、TDW に表示されるウィンドウ管理を行っている。Free Space Manager がそれぞれのディスプレイに表示される画像や映像を同期し、制御することで TDW の一貫した高解像度画像表示を可能としている。

ユーザはウィンドウの移動やサイズの変更を (f) や (g) のユーザ操作端末上で動作する SAGE UI で行い、その操作はメッセージとして (c) の Free Space Manager に送られる。Free Space Manager はウィンドウの変更操作に伴い、表示位置やサイズを変更するために、SAIL と SAGE Receiver に制御メッセージを送信し、画像データを更新する。

3 大規模データ共有可視化環境構築時の課題

SAGE を利用して構築された TDW は、大規模なデータを細部の情報を失うことなく高解像度で表示したい科学者や研究者にとって有力な表示技術である。しかし、複数拠点間で SAGE による TDW を用いて大規模データ共有可視化環境を構築するためには、SAGE に

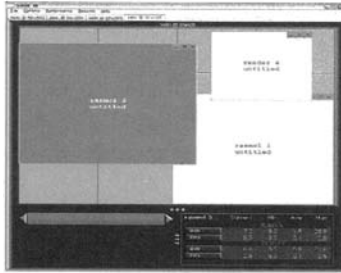


図 3 SAGE UI

は解決されなければならない多くの技術的課題が残されている。ここでは、我々の考える早急に解決すべき主要な 5 点の課題をあげ、整理する。

1. インタフェースの二重化

SAGE が提供するウィンドウ操作インタフェース SAGE UI はアプリケーションウィンドウの移動やサイズ変更といった操作のみを可能とし、アプリケーション自体の操作は受け付けない。アプリケーション自体の操作とは、アプリケーションウィンドウ内でのマウスクリックやドラッグ、キー入力等である。SAGE UI はこれらの操作を受け付けないため、ユーザがアプリケーションの操作を行いたい場合は、SAGE UI とは別にアプリケーションが提供するインタフェースを用いて操作を行わなければならない。

例えば、図 4 のような TDW の利用例を考える。例ではアプリケーションは Free Space Manager が動作している管理ノードと同一の計算機上で動作している。一方、ユーザは SAGE に関する操作をユーザ操作端末上で実行される SAGE UI 通じて行う。ユーザがアプリケーション自体の操作を行いたい場合、SAGE UI でこのような操作を行うことはできないため、アプリケーションが動作している計算機、つまりこの例では管理ノードの前に移動し、アプリケーションウィンドウに対し直接操作を行わなければならない。また、アプリケーションは必ずしも管理ノード上で実行されるわけではなく、遠隔地の高性能なレンダリングサーバ上で実行され、画像を TDW にストリームする場合もある。このような場合、ユーザは物理的に遠隔地のサーバにアクセスする手段はなく、アプリケーションの操作を自分自身で行うことは不可能となる。このようなインタフェースの二重化は SAGE の操作性を著しく低下させている要因となっている。

2. 同一アプリケーションの複数ユーザでの同時利用が不可能

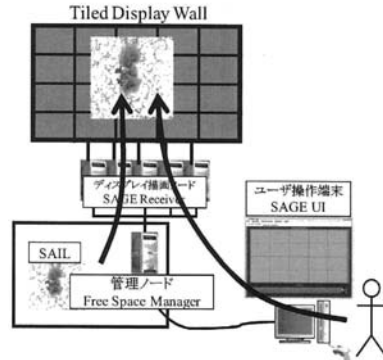


図 4 SAGE の利用例

高性能なレンダリングサーバでは、同じアプリケーションを異なるユーザが同時に使用する状況が発生することが考えられる。SAGE ではアプリケーションが生成する画像のストリーム先を指定するために、アプリケーション毎に図 5 に示されるような設定ファイルを作成し、ストリーム先の TDW の管理ノードの IP アドレスやポート番号を静的に明示している。アプリケーションと設定ファイルは一対一で対応しているため、ストリーム先が異なる場合は、設定ファイルを書き換えるか別の設定ファイルを用意する必要がある。このような設定方法のため、一つのアプリケーションを異なるユーザが同時に使用する場合に問題が生じる。

```
masterIP 192.168.0.1
nwID 1
msgPort 23010
syncPort 13010
nodeNum 1
appID 0
pixelBlockSize 1024 1024
blockThreshold 0
winWidth 1000
winHeight 1000
nwProtocol TCP
asyncUpdate false
```

図 5 アプリケーション設定ファイル例

例えば図 6 に示すように、ある遠隔地にあるレンダリングサーバ上のアプリケーションを使って、異なる地点にいるユーザ A とユーザ B が同時に画像を表示させたい場合を考える。アプリケーションと設定ファイルは一対一で対応しており、複数の設定ファイルを選択的に読み込むことはできない。このため、同一アプリケーションを使用する場合においてもアプリケーションを 2 つ、ユーザ A の環

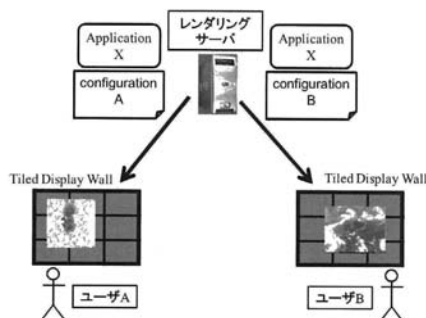


図 6 同一アプリケーションの同時利用

境に合わせた設定ファイル，ユーザ B の環境に合わせた設定ファイルをそれぞれ用意しなければならない。

3. データストリーミングのためのセキュリティの欠如

遠隔地へネットワークを利用して研究データ等の情報を他者へ公開する際には，閲覧者の特定など情報に対するセキュリティの確保が必要である。しかし，図 7 の複数地点でのデータ共有に関する設定ファイルの例に示されているように，SAGE は画像や映像のストリーム先の IP アドレスやポート番号のみを設定，管理しており，ユーザの指定・認証は行っていない。そのため，ストリームの受信先で SAGE を利用できるユーザであれば，誰でも画像を TDW に表示させることができる。研究機関で扱うデータには，機密情報や個人情報が含まれる場合もある。一ヶ所の TDW を複数のユーザが共同利用することは十分に考えられ，ストリームしたデータを表示可能なユーザを認証・認可する技術が SAGE に備わっていないことは，セキュリティの観点から大きな問題となる。

```

masterIP 192.168.81.121
slaveList 3 10.0.8.122 10.0.8.123 10.0.8.124
streamPort 41000
msgPort 42000
syncPort 43000
screenRes 1000 1000

rcvNwBufSize 16000000
sendNwBufSize 65536
MTU 1450

```

図 7 複数地点データ共有設定ファイル例

4. 拠点間のコミュニケーションツールの欠如

拠点間連携を行う上で，ユーザ同士の意思の疎通を図ることは大変重要である。SAGE には SAGE UI にチャット機能があるが，拠点連携を行う際のコミュニケーションツールとしては十分ではない。

ユーザが表示されたデータを協力して解析するためには議論をしなければならない。この場合においてチャットのような文字でのコミュニケーションではなく，音声での議論が必要であると考えられる。また，SAGE では相手の TDW にどのような映像が表示されているかを知ることはできない。さらに，画像の注目する部分を指しながら議論することは日常的に行われているが，SAGE においてはポインタを表示する機能はない。以上のような要望を満たすためには，お互いの TDW に表示されている画像をビデオカメラ等を利用して送信するといったことが必要になってしまう。

5. 表示画像に対する機能の欠点

画像や映像を解析する際に，画像の中で注目すべき部分や，いくつかの画像を比較している場合，それらを見たとおりに保存しておき，後で再び表示させる場合がある。一般的な計算機においては，いわゆるスナップショットと呼ばれる機能があり，ディスプレイに表示されている画面をそのまま画像データとして保存することができる。しかし，SAGE にはそのような機能はなく，TDW に表示されている画像を保存することはできない。したがって，以前と同様な状況を作り出すには，毎回アプリケーションにデータを読み込ませ，TDW に表示し再現しなければならない。

4 インタフェースの二重化に対する取り組み

3 節では SAGE を用いて協調作業環境を構築する際に，SAGE が抱える 5 点の問題を取り上げた。本研究では 3 節で述べた SAGE が抱える問題のうち，インタフェースの二重化の問題は協調作業を行う上で大きな障害になると考え，このアプリケーションの操作性の問題について注目し，その解決策に関して議論する。本研究では，この SAGE が抱えるウィンドウ操作とアプリケーション操作インタフェースの二重化の問題を解決するために，アプリケーションコントロールモジュールを構築した。本節では本モジュールの設計と実装について述べる。本モジュールは SAGE UI を拡張し，ユーザに SAGE 及び SAGE 上で動作するアプリケーションの操作を可能とする，一元化されたユーザインタフェースを提供するものである。本研究では本モジュールを設計する際，下記の 3 項目を設計方針として定め，設計・実装を行った。

● 現行システムとの互換性

構築するアプリケーションコントロールモジュールは現行のシステムとの互換性を確保する。本モジュールを導入することによって現行シス

テムにおけるメッセージフォーマットやインタフェースが使用不可能になることはなく、本モジュールを有効にしたシステムと現行システムが混在可能なように設計する。また、本モジュールを使用するユーザによって容易に本モジュールの使用及び不使用の選択ができるように設計を行う。

- 既存の SAGE アーキテクチャに対する変更の最小化

SAGE は現在も活発に研究開発が継続されているミドルウェアである。そのため、現在の SAGE の実装に大きく依存するモジュールの構築は将来の利用可能性が低くなる。そこで、本モジュールの機能は可能な限りモジュール内にカプセル化し、既存の SAGE アーキテクチャへ本モジュールを組み込む際に必要なプログラムの変更を最小限に抑えることを目指す。

- アプリケーションへの簡便な導入の実現

既存の SAGE 用アプリケーション資産を有効活用するためには、本モジュールを利用するにあたって、既存のアプリケーションへの変更を最小限に抑える必要がある。最小限に抑えることでアプリケーションの移行を容易にし、アプリケーションの再利用性を高める。

アプリケーションコントロールモジュールはアプリケーションイベント検知、アプリケーションイベント送信、アプリケーションイベント解析の 3 つの主な機能から構成されている。図 8 はこれら 3 つの機能がどのように SAGE に統合されているか、そしてアプリケーションイベントがどのようにしてインタフェースからアプリケーションに伝わるかを示している。アプリケーションイベント検知、アプリケーションイベント送信の機能をそれぞれイベント検知機構、イベントメッセージ送信機構として SAGE UI を拡張することによって実現している。また、アプリケーションイベント解析の機能をイベントメッセージ解析機構として、SAGE 用アプリケーションで使用されるライブラリである SAIL を拡張して実現している。

本アプリケーションコントロールモジュールは次のように動作する。まず、ユーザは SAGE UI を用いて操作イベント（アプリケーションウィンドウ内でのマウスクリックやドラッグ操作）を入力する（図中 1）。イベント検知機構は入力された操作イベントを検知する（図中 2）。検知されたイベントはイベントメッセージ送信機構によって Free Space Manager を経由して適切なアプリケーションまで中継可能なメッセージに変換され、Free Space Manager に送信される（図中 3）。Free Space Manager は受信したイベントメッセージを対応するアプリケーションに転送する（図中 4）。イベント

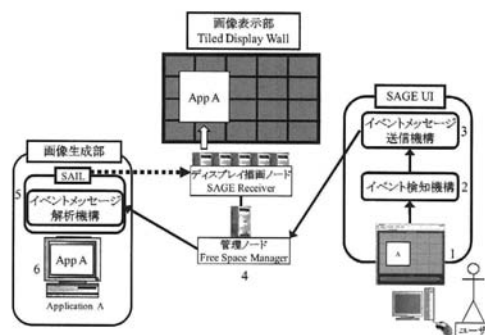


図 8 アプリケーションコントロールモジュールの概要

メッセージ解析機構は、受信したイベントメッセージを解析し、X Window System の操作イベントメッセージに変換し、アプリケーションにイベントの入力を行う（図中 5）。最後に、アプリケーションは入力されたイベントに対するアクションを実行する（図中 6）。このように、SAGE UI 上で発生したアプリケーションに対する操作を目的のアプリケーションの X の操作イベントとして届けることによって、SAGE UI を用いたアプリケーションの操作を可能とする。以下、本モジュールを構成するそれぞれの機構の詳細な設計を説明し、実装方法について述べる。

4.1 イベント検知機構

イベント検知機構は SAGE UI を用いてユーザが入力したアプリケーションに対する操作イベントを検知する。イベント検知機構は操作対象のアプリケーション、操作内容としてキー入力の状態、マウスボタンの状態、ポインタの座標を検知するイベントハンドラとして実装した。SAGE UI はその GUI 実装にクロスプラットフォーム GUI ライブラリである wxWidgets [4] を用いている。wxWidgets はマウスクリックやドラッグ操作に応じてウィンドウの移動、サイズ変更を行う標準的なウィンドウの実装や GUI パーツを提供するライブラリである。標準的なウィンドウの操作に対して独自の操作イベント処理を追加したい場合は、独自にイベントハンドラを実装し、ウィンドウに関連付けることによって処理可能である。SAGE UI は既に TDW 上のウィンドウ操作のための独自イベントハンドラを実装し、wxWidgets のウィンドウに関連付けている。本研究ではこの既存の関連付けを本研究で構築したイベントハンドラによって上書きし、アプリケーション操作の検知を可能にした。ただし、アプリケーション操作に無関係の操作の場合は SAGE UI が実装したイベントハンドラに処理を転送することによって既存のイベントハンドラも有効にした。このような設計はオブジェクト

指向設計の分野では Chain of Responsibility パターンとして知られており、主に GUI システムにおいて複数のイベントハンドラ間の独立性を高める設計方法の一つである。

図 9 に本機構のイベントハンドラと SAGE UI, デフォルトの wxWidgets のイベントハンドラの関係を示す。AppEvtHandler は本機構のために構築したハンドラである。MyEvtHandler は SAGE UI によって実装されたハンドラで、ウィンドウ操作に関するイベントを処理している。ShapeEvtHandler は wxWidgets のデフォルトのハンドラである。このようにアプリケーション操作検知に関する処理を一つのイベントハンドラとして独立して実装することによって、SAGE UI に対するソースコードの変更を最小限に抑えつつ、アプリケーションイベントを検知することが可能となった。

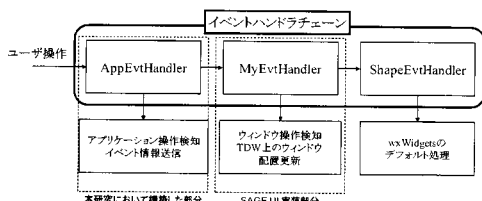


図 9 イベントハンドラチェーン

4.2 イベントメッセージ送信機構

イベントメッセージ送信機構はイベント検知機構から送られてきたイベント情報を基にイベントメッセージを作成し、Free Space Manager に送信する。

SAGE UI には既に Free Space Manager とメッセージを送受信するためのクラスである SAGEGate クラスが存在する。SAGEGate クラスは SAGE UI プログラムの様々な場所からユーティリティクラスとして利用されるクラスであり、クラスが実装するメソッドは相互に独立である。具体的には、ウィンドウサイズの変更、アプリケーションの実行、SAGE の終了などを伝えるメッセージの送信を行うメソッドが存在し、それらの間に依存関係はない。本研究ではこの SAGEGate クラスにアプリケーション操作イベント送信用のメソッドを追加しても、既存のアーキテクチャを大きく変更することはないと考え、この SAGEGate クラスを拡張し、操作イベント情報を送信するメソッドを追加した。これにより、既存の SAGE UI, Free Space Manager, アプリケーション間の通信経路を最大限に利用し、操作イベントを適切なアプリケーションに伝えるイベントメッセージ送信機構の実装を行った。

4.3 イベントメッセージ解析機構

イベントメッセージ解析機構は Free Space Manager を介して SAGE UI から送られてきたアプリケーション

ン操作イベントに関するメッセージを解析し、操作対象となるアプリケーションの X のイベントキューに X のイベントを挿入する。Free Space Manager を介して送信された操作イベントメッセージは、既存の SAGE のメッセージと同様に SAIL 内部にて受信される。本研究では操作イベントメッセージを解析するために appMessageHandler を構築し、SAIL 内で受信したメッセージをこの appMessageHandler に転送するようにした。ただし、本モジュールを利用するために初期化を行った場合のみ SAIL で受信された操作イベントメッセージを appMessageHandler に転送する設計を行うことで、既存の SAIL に対する変更を最小限にした。このような設計により、既存の SAGE アーキテクチャに大きな変更を加えることはなく、本モジュールを使用したアプリケーションと、そうでない既存のアプリケーションを混在して利用することが可能である。

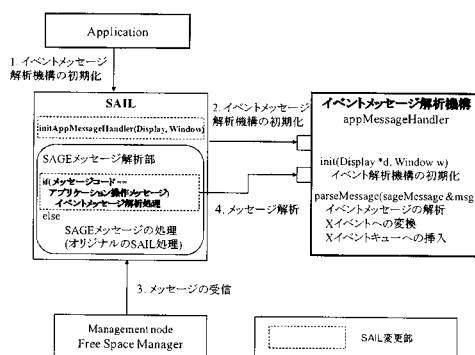


図 10 イベントメッセージ解析機構

図 10 にアプリケーションにおけるイベントメッセージ解析機構の構造を示す。イベントメッセージ解析機構はアプリケーションの display ID と window ID によって初期化される。アプリケーションにおいて一度この機構が初期化されると SAIL は Free Space Manager を介して SAGE UI から送られてきた操作イベントメッセージを appMessageHandler に転送するようになる。既存のアプリケーションを本モジュールに対応させるためには、この初期化のためのメソッドを呼び出すただ一つのコードをアプリケーションのソースコードに追加するのみでよく、既存のアプリケーションに対する変更は最小限に抑えられている。アプリケーション開発者は SAIL 内部でメッセージが受信され、それが appMessageHandler に転送され解析されるプロセスに関して意識する必要がない。イベントメッセージ解析機構は、図に示されている 'parseMessage' でメッセージを解析し、メッセージを X のイベントに変換し、アプリケーションに入力する。そのため、アプリケーション

4.4 評価

この全測定は、一般的な 100Mbps の LAN 環境内に組まれた TDW 上で行った。TDW は 2 面のディスプレイとディスプレイ描画ノードとしての 2 台の PC、1 台の管理ノードと SAGE UI が実行されているユーザ操作端末から構成されている。測定の為に用いたこれらの各 PC は Intel Pentium 4 2.53GHz のシングルプロセッサと 1GB のメモリを搭載している。OS として CentOS 4 を採用している。

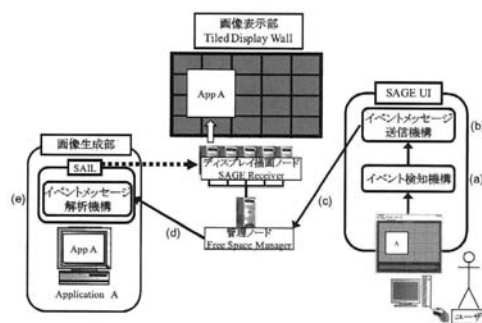


圖 11 測定環境

たアプリケーションの操作において、違和感の無い応答を得ることができると言え、本モジュールは有用であると考えられる。

5 まとめ

謝辭

参考文献

- [1] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal Supercomputer Applications*, Vol. 15, No. 3, pp. 200–222, Fall 2001.
- [2] T. Ni, G. S. Schmidt, O. G. Staadt, M. A. L., R. Ball, and R. May. A survey of large high-resolution display technologies, techniques, and applications. In *Proc. IEEE Virtual Reality Conference 2006 (VR 2006)*, pp. 223–236, March 2006.
- [3] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh. High-performance dynamic graphics streaming for scalable adaptive graphics environment. In *Proc. the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC06)*, CDROM, November 2006.
- [4] J. Smart, K. Hock, and S. Csomor. *Cross-Platform GUI Programming With wxWidgets*. Prentice Hall PTR, September 2005.
- [5] テレビジョン学会. テレビジョン・画像工学ハンドブック. オーム社, 第1版, December 1980.