Mogul: 位置透過型分散共有ツールキットライブラリ

中澤 仁[1]　望月 祐洋[1]　徳田 英幸[1,2]
[1]慶應義塾大学政策・メディア研究科，[2]慶應義塾大学環境情報学部

協調作業支援アプリケーションを典型例とする分散マルチユーザアプリケーションの構築には，動的分散機構や状態共有機構の実現が必要である．本研究において開発された位置透過型分散共有ツールキットライブラリ (Mobile Graphical User Interface Library : Mogul) は，視覚的状態共有機能やホスト間での移動および複製機能を有するウィジェット群を提供する．Mogul を用いることによって，これらの機構の実装に伴うプログラマの負担を軽減し，ユーザに対して柔軟なアプリケーション環境を提供できる．本稿では，Mogul の設計，各種処理機構の実装および評価について述べる．

Mogul: Mobile Graphical User Interface Library

Jin Nakazawa[1]　Masahiro Mochizuki[1]　Hideyuki Tokuda[1,2]
[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University

Mobile Graphical User Interface Library : Mogul provides widgets which have functions of sharing status and migration, and simplifies the Java programming for implementing mechanisms of shared data or dynamic mobility in distributed multi-user applications. In this paper, requirements of the distributed multi-user applications are introduced and several issues on the problems of existing tools for constructing it are discussed. Furthermore, design, implementation and evaluation of Mogul is described.

1　はじめに

本研究では，視覚的状態共有機能と，ホスト間でのインスタンス複製および移動機構を持ったウィジェット群を提供する，位置透過型分散共有ツールキットライブラリ (Mobile Graphical User Interface Library : Mogul) を開発した．本研究の目的の第一は，分散マルチユーザアプリケーション開発者の負担軽減であり，第二は分散マルチユーザアプリケーションユーザに対する柔軟な利用環境の提供である．

Mogul を用いることによってアプリケーションプログラマは，視覚的状態共有機能や複製機能，移動機能を有する分散マルチユーザアプリケーションを極めて容易に開発できる．さらに，Mogul の提供する複製ウィンドウ間の一貫性制御機構や，ウィンドウに対するアクセスコントロール機構は，プログラマあるいはユーザによる柔軟なアプリケーション制御を可能とする．

Mogul の特徴は，上記機能を複製先あるいは移動先ホストの位置に対して透過的に実現している点である．ユーザあるいはプログラマは，本ライブラリが実現するウィジェットを利用して，宛先ホストを指定せずに複製や移動などの操作を行える．また，Mogul が内部に実現しているフロアコントロール機構やアクセスコントロール機構を用いて，柔軟な操作を行える．

本稿では，第 2 節において分散マルチユーザアプリケーションの機能要件および構築ツールについて考察する．第 3 節において，Mogul を用いて構築可能な様々な応用事例を挙げ，既存の構築ツールと比較して容易かつ柔軟に分散マルチユーザアプリケーションを構築できることを示す．第 4 節以降の各節において，ライブラリの設計，各種処理機構の実装および評価についても述べ，最後にまとめと今後の課題を述べる．

2　分散マルチユーザアプリケーション

本節では，分散マルチユーザアプリケーションの機能要件について述べた上で，構築ツールが満たすべきいくつかの特性について考察する．

2.1　機能要件

分散マルチユーザアプリケーションには，それを定義づける基本的な機能要件として，以下の 2 つが存在する．

動的分散機能　アプリケーションインスタンス間で，何らかのデータ構造をリモートホストに複製または移動する機能

状態共有機能　アプリケーションインスタンス間で，ウィンドウあるいは共有データの視覚的・内部的状態の共有を行う機能

例えば，電子会議システムなどの同期対面型協調作業支援アプリケーションにおいては動的分散機能に対する制約が高く，回覧板システムなどの非同期分散型協調作業支援アプリケーションにおいては状態共有機

能に対する制約が高い [1][2].

2.2 構築ツール

これらの基本機能の実現に伴うプログラマの負担を軽減することを目的として，いくつかの分散マルチユーザアプリケーション構築ツールが存在する．

プログラマの負担を軽減し，ユーザに対して柔軟なアプリケーション環境を提供するために，分散マルチユーザアプリケーション構築ツールは，以下に示す 3 つの性質を備える必要がある．

隠蔽性 構築ツールの提供する機構が，プログラマやユーザに対して隠蔽され，容易に利用できる

柔軟性 同機構に対して，さまざまな制御ポリシをプログラマやユーザが容易に指定できる

軽快性 CPU およびメモリを始めとする計算機資源やネットワーク資源を逼迫しない

3 Mogul

本研究では，分散マルチユーザアプリケーション構築ツールとして位置透過型分散共有ツールキットライブラリ (Mobile Graphical User Interface Library : Mogul) を開発した．本節では，Mogul の提供する機能の概要を述べ，既存の構築ツールに関する研究との比較を行う．

3.1 機能

Mogul は，移動性と共有性および各種処理機構を持ち，ライブラリとして提供されるウィジェット群 (以下 Mogul ウィジェットと呼ぶ) と，Mogul ウィジェットに対する操作機構をユーザコマンドとして提供するコマンド群とから構成される．これにより，分散マルチユーザアプリケーションのユーザインタフェース開発，ホストを跨ったアプリケーションインスタンス間でのデータ共有機構，およびデータ移動機構の実装，ユーザを対象とした柔軟な操作機構の実装を支援する．

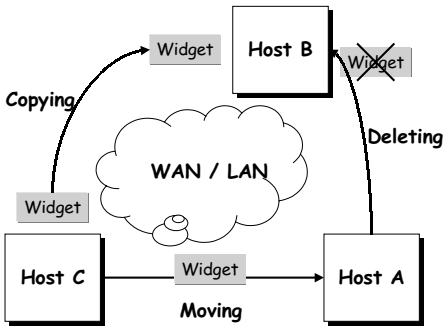

図 1: Mogul ウィジェットの移動性

移動性

移動性とは，生成した Mogul ウィジェットインスタンスをリモートホスト上に動的に移動あるいは複製する機能である．特定ウィジェットインスタンスに対して，移動および複製に関するアクセスコントロールを柔軟に指定可能である．図 1では，ホスト C からホスト B に対してウィジェット複製が，ホスト C からホスト A に対してウィジェット移動が，ホスト A からホスト B に対してウィジェット消去が行われている様子を表している．本機能により，分散マルチユーザアプリケーションの機能要件のうち，動的分散機能を実現できる．

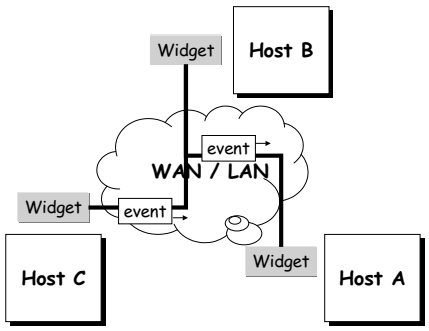

図 2: Mogul ウィジェットの共有性

共有性

共有性とは，生成したウィジェットインスタンス間で描画情報やイベントを共有するための機能である．例えばキーボードからの入力，マウスによる入力などがイベントに該当する．通常のスタンドアロンアプリケーションプログラムに対する追加工程なしで，ウィジェットインスタンス間でのイベントや描画情報の共有が可能であり，イベントあるいは描画情報を共有可能なウィジェットインスタンスの指定や，共有の粒度などを柔軟に指定可能であることが望ましい．図 2では，ホスト A，B，C 上に 3 つの共有ウィジェットが存在し，それらの間でイベントが共有されている様子を表している．本機能により，分散マルチユーザアプリケーションの機能要件のうち，状態共有機能を実現できる．

3.2 特徴

Mogul は，ユーザインタフェース開発やホスト間でのデータ共有機構などを含めた分散マルチユーザアプリケーション開発を，総合的に支援できることを最大の特徴とする．さらに，以下の各事項が Mogul の特徴として挙げられる．

コーディングの容易性

Mogul ウィジェットの提供する各種の機能は，ライブラリ内部に隠蔽して実現されている．すなわち，分散マルチユーザアプリケーションの実装に際して，ウィンドウおよびウィジェットのホスト間移動機能や複製機能，ウィジェット間の状態共有機能の実現に伴う，通常のシングルユーザアプリケーションに対するコードの追加は全く必要ない．

ホスト位置透過性

Mogul が提供している各種の処理機構は，Mogul ウィジェットのホスト位置に対して透過的に実現されている．すなわち，プログラマはアプリケーション記述段階で，Mogul ウィジェットのホスト位置を考慮する必要はない．特定の Mogul ウィジェットに対して，ウィジェットインスタンスがネットワーク上のどのホスト上に存在しても，各種操作を実行可能である．

柔軟な操作性

Mogul が提供している各種処理機構の制御ポリシの動的な設定機構は，プログラマに対してはライブラリとして，またユーザに対してはユーザレベルコマンドとして提供されている．制御ポリシとしては，フロアコントロールポリシや共有イベント直列化ポリシ，アクセス制御ポリシなどが考えられる．

3.3 利用例

Mogul を用いることによって，通常のシングルユーザアプリケーションプログラムを記述するのとほぼ同様に，移動性および共有性を備えた分散マルチユーザアプリケーションを構築できる．その例として本項では，一つのプッシュボタンのみからなるウィンドウアプリケーションを取り上げる．図 3にサンプルアプリケーションのスクリーンダンプを，図 4にサンプルアプリケーションのコードを示す．

図 3: サンプルアプリケーションのスクリーンダンプ

3.3.1 サンプルアプリケーションの機能

図 3に示したサンプルアプリケーションは，図 4に示す非常に単純なプログラムによって構成されている．このアプリケーションは Mogul を用いて実装されているため，動的分散性および状態共有性を有する分散マルチユーザアプリケーションとして動作する．

ユーザは，Mogul の提供するコマンド群を用いて，ウィンドウ全体または *Push me!*ボタンに対して以下に示す各種の操作を行える．

- ホスト間複製・移動
  post，*fetch*，*move* および *delete* の各コマンドを用いてホストを跨った複製，移動および消去が可能
- フロアコントロール
  floor コマンドを用いて，複製ウィジェットインスタンスに対する入力制限による状態の一貫性制御が可能
- アクセスコントロール
  acl コマンドを用いて，各操作に対するユーザおよびドメインを単位としたアクセスコントロールが可能

```
00 import JP.ad.wide.sfc.kmsf.mogul.widget.*;
01 import java.awt.event.*;
02
03 public class ButtonTest extends SharedFrame
04 implements ActionListener{
05   SharedButton button;
06
07   public ButtonTest(){
08     setTitle("ButtonTest");
09     button = new SharedButton("Push me!");
10     button.addActionListener(this);
11     add("Center", button);
12   }
13
14   void actionPerformed(ActionEvent e){
15     System.out.println("pushed!");
16   }
17
18   public static void main(String argv[]){
19     ButtonTest window = new ButtonTest();
20     window.pack();
21     window.show();
22   }
23 }
```

図 4: Mogul を用いた Java アプリケーションコード

3.3.2 シングルユーザアプリケーションとの相違

図 4に示したサンプルアプリケーションコードが，シングルユーザアプリケーションのコードと異なる点は，00 行の *import* 文で Java 言語標準のウィジェットライブラリを用いずに Mogul の提供するウィジェットライブラリを用いることを宣言している点と，03，09 行において各ウィジェット名に *Shared* というキーワードが付加されている点だけである．

このように，Mogul を用いることによって，分散マルチユーザアプリケーションの動的分散機能および状態共有機能を，通常のシングルユーザアプリケーションに対する追加コードを必要とせずに実現できる．

3.4 応用例

前項では，Mogul を用いたサンプルプログラムを取り上げ，分散マルチユーザアプリケーションが容易に構築できることを示した．本稿では，様々な応用例のうち代表的な 2 つを取り上げ，より複雑な分散マルチユーザアプリケーションに対しても Mogul が適用可能であることを示す．

3.4.1 回覧板システム

回覧板システムは，ウィンドウとしての視覚的実体を持つ何らかの情報が複数のユーザ間を順番に移動し，最終的に回覧開始者のもとに返却されるシステムである．

Mogul を用いることによって，回覧板ウィンドウは通常のシングルユーザアプリケーションと同等のプログラミングによって構築できる．また既存の Java アプリケーションを変換プログラムを用いて Mogul を

用いたものに変換し，回覧板として利用することも可能である．

3.4.2 共有ホワイトボードシステム

共有ホワイトボードシステムは，Mogul の共有性を利用する例である．プログラマは，通常のペイントアプリケーションを記述することによって，複数ユーザ間での共有を意識せずに共有ホワイトボードシステムを実現できる．

まず一人のユーザが共有ホワイトボードシステムを起動し，他のユーザに対して位置透過的に `post` コマンドを実行するか，他のユーザが `fetch` コマンドを実行することによって，当該ホワイトボードに対する操作を共有できる．

3.5 関連研究

分散マルチユーザアプリケーション構築時の複雑なプログラミング工程を軽減することを目的としたさまざまなツールが存在する．

Dewan(1993) はこれらのツールを以下に示す 8 つのカテゴリに分類している [4]．本項では，これらのうち本研究と関連の深いツールを取り上げ，本研究との差異を明らかにする．

- データベースシステム
- 分散システム
- メッセージサーバ
- 共有オブジェクトシステム
- 共有ウィンドウシステム
- マルチユーザツールキット
- マルチユーザ UIMS
- マルチユーザ UI ジェネレータ

3.5.1 共有オブジェクトシステム

Stefik(1987) による **Colab**[5] は共有オブジェクトシステムの典型例である．これには 3 つの利点がある．

- 複数のクライアントプロセスによって同時にアクセス可能なデータ構造
- 共有データに対する一貫性制御，アクセスコントロール機構を持つ
- 共有相手に対して透過的にオブジェクトの共有が可能

しかし，共有オブジェクトシステムはウィンドウシステムを用いたものではなく，ウィンドウに対する描画状態の共有機能などを分散マルチユーザアプリケーションに実現する際には，プログラマに対する負担は増加する．

3.5.2 共有ウィンドウシステム

Wahab(1991)[6][7] による **XTV** は，X プロトコルを多重化することによって，任意の X アプリケーションを複数ユーザ間で共有するための機構を提供する．また，既存の X アプリケーションをそのまま分散マルチユーザアプリケーションとして利用できるという利点がある．さらに，各ウィンドウに対する入力権限を制御するフロアコントロール機構を実現しており，柔軟性を高めている．

しかし **XTV** は，X プロトコルを多重化するのみであるからウィンドウのホスト間移動は実現されておらず，さらにウィジェット単位でのフロアコントロールなど粒度の細かい制御が実現できない．

3.5.3 マルチユーザツールキット

Wahab(1996)[8] による **Collawt** は Java 言語によって実装されたマルチユーザツールキットである．**Collawt** の提供するウィジェット群はイベント共有を基礎とした描画状態共有機能を持つが，ウィジェットの動的なホスト間移動を実現していない．**Collawt** ではフロアコントロール機構をライブラリ内部に実現し，通常のシングルユーザツールキットとほぼ同様の書式で分散マルチユーザアプリケーションを記述できるが，アクセスコントロール機構を持たないためプログラマに対する負担となる．

4 設計

Mogul は，**ウィジェット部**，**ユーザマネージャ部**，**システムマネージャ部**および**コマンド部**の 4 サブシステムによって構成される．図 5に各サブシステム間の関係を示す．本節では，各サブシステムの設計と，提供する機能について概要を記述する．

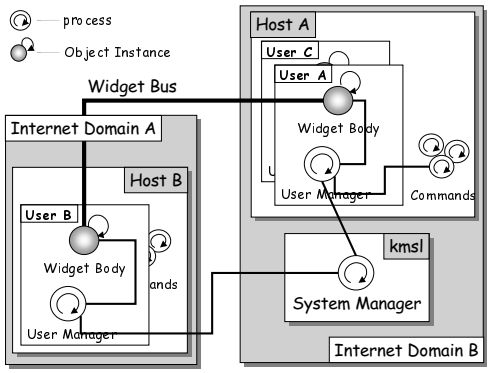

図 5: ソフトウエア構成概念図

4.1 ウィジェット部

ウィジェット部はウィジェット本体部とウィジェットバス部から構成される．

4.1.1 ウィジェット本体部

ウィジェット本体部は，プログラマから直接操作可能な Mogul ウィジェットをライブラリとして提供する．全てのウィジェットは，図 6に示す識別子を用いた名前空間において，一意に識別される．

```
//foo.bar.co.jp/userwho/sftest/0
```

図 6: ウィジェット識別子

図 6に示すウィジェット識別子は，ホスト *foo* 上の

ユーザ *jin* が所有している sftest ウィジェットの 0 番目のインスタンスに割り当てられる

4.1.2 ウィジェット バス部

ウィジェットバス部は，特定ウィジェットの全複製インスタンスに対して一つ存在し，当該ウィジェットの複製間で処理の一貫性を保つ．図 7はウィジェット本体部とウィジェットバス部の関係を示している．図 7 からわかるように，特定ウィジェットの全ての複製は一つのウィジェットバスを共有している．

共有イベントの伝播と複製ウィジェット間でのフロアコントロールはウィジェットバスを通して行われる．

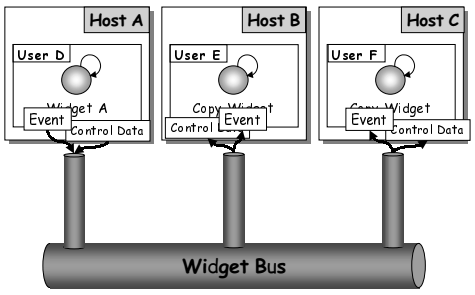

図 7: ウィジェットバス部概念図

4.2 ユーザマネージャ部

ユーザマネージャ部は Mogul ウィジェットに対して，ホスト間の動的な移動，複製機構を提供するサーバプログラムである．Mogul ウィジェットはインスタンス生成時に，ユーザマネージャの持つウィジェット管理テーブルへ登録され，ウィジェット識別子を得る．

ウィジェットのホスト間移動および複製は，ユーザマネージャ間の通信によって実現される．ユーザマネージャはウィジェットインスタンスに対して以下に示す 3 つの機能を提供する．

複製機能 ウィジェットインスタンスの複製をリモートホスト上に作成する機能

削除機能 リモートあるいはローカルホスト上に存在するウィジェットインスタンスを削除する機能

移動機能 ローカルホスト上に存在するウィジェットインスタンスをリモートホスト上に移動する機能

4.3 システムマネージャ部

システムマネージャ部は，Mogul で実現される位置透過的なウィジェット識別を実現するためのサーバプログラムである．ユーザマネージャとは異なり，各インターネットドメインに一つ動作する．Mogul を用いて構築されたアプリケーションは，システムマネージャが存在しなくても動作するが，この場合は位置透過性を保持できなくなる．

4.4 コマンド 部

コマンド部は，ユーザマネージャ部およびウィジェット部を，ユーザコマンドによって直接操作するための機構を提供する．コマンド部は，表 1に示す 5 つのコマンドによって構成される．全てのコマンドはユーザマネージャに対する通信を行い，それぞれの操作要求を行う．

表 1: コマンド一覧

コマンド	概　　要
`post`	ウィジェットインスタンスをリモートに複製 *post //miro.sfc.wide.ad.jp/jin/sftest/0* *moma@sfc.wide.ad.jp*
`fetch`	ウィジェットインスタンスをローカルに複製 *fetch //sfc.wide.ad.jp/jin/sftest/0*
`list`	ウィジェット管理テーブルから鍵を表示 *list jin@sfc.wide.ad.jp*
`delete`	指定したウィジェットインスタンスを消去 *delete //sfc.wide.ad.jp/jin/sftest/0*
`acl`	アクセスコントロールリストを更新する *acl deny jin@sfc.wide.ad.jp*

5 実装

Mogul の実装は Java 言語を用いて行った．JDK(Java Development Kit) のマルチプラットフォーム性から，ユーザは自分自身の使い慣れたオペレーティングシステムやユーザインタフェース上で，Mogul を用いて構築された分散マルチユーザアプリケーションを利用できる．

5.1 Mogul ウィジェット

ウィジェット本体部の提供する Mogul ウィジェットは，Java AWT におけるウィジェットのサブクラスとして実装した．既存の Java AWT ウィジェットに対して Mogul が実現している追加的機構は，ライブラリ内部に隠蔽実装されている．従ってプログラマは，既存の Java AWT ウィジェットの性質のみを考慮して，Mogul を用いた分散マルチユーザプログラミングが可能である．

Mogul ウィジェットのクラス名は，Java AWT ウィジェットのクラス名に，*Shared* という接頭辞を付加したものとなる．

5.2 ホスト 間移動・複製機構

Mogul ウィジェットのリモートホストに対する複製および移動は，ユーザマネージャ間の **Java RMI(Remote Method Invocation)**(以下 RMI と呼ぶ) [9] による通信により実装した．ウィジェット複製の場合，複製元ユーザマネージャの *postIt* メソッド中で，複製先ユーザマネージャの持つ *postIt* メソッドを RMI によって呼び出し，ウィジェット複製インスタンスを引数としてシリアライズすることによって実現される．図 8はユーザマネージャに実装された *postIt* メソッドである．

5.3 状態共有機構

同一ウィジェットの全複製インスタンス間で描画状態を共有する機構は，ウィジェットバスを用いてイベントを共有することによって実現した．ウィジェットバスは，同一ウィジェットの全複製インスタンスが共

```
public void postIt(SAWTComponent c,SAWTID org)
throws RemoteException {
  String managerID = c.getID().getManagerID();
  if(managerID.isMine()){
    RemoteSAWTManager dest = org.getManager();
    dest.postIt(c, ID);
    return;
  }
  else{
    if(c instanceof Container)
      processRecursively((Container)c);
    c.reset();
    SAWTEvent e;
    e = new SAWTEvent(c,org,SAWTEvent.POSTED);
    notifySAWTListener(e);
  }
}
```

図 8: *SAWTManager* クラスの *postIt* メソッド

有する仮想的な通信路であり，RMI を用いて実装されている．

共有されるイベントオブジェクトのインスタンスは，リモートウィジェットインスタンスの持つイベント処理メソッドに対する引数として，RMI 呼び出し時にシリアライズされる．図 9はウィジェットバス部に実装されているイベント配送メソッドである．

```
public void deliverEvent(SharedEvent e)
throws RemoteException{
  RemoteSAWTComponent org, dst;
  org = e.getComponentPeer();
  int size = peers.size();
  for(int i=0; i < size; i++){
    dst = (Remote...)peers.elementAt(i);
    if(!org.equals(dst))
      dst.processEvent(e);
  }
}

public void processEvent(SharedEvent e)
throws RemoteException{
  e.rebuild(body);
  if(!e.getOriginalID().isLocal())
    queue.postEvent(e);
}
```

図 9: *SAWTComponentPeer* クラスのイベント配送メソッド

5.4 フロアコントロール機構

ウィジェットに対する入力権限を制限するフロアコントロールは，コントロールのポリシをウィジェットバス部が管理し，当該ウィジェットの全複製に共有されることによって一貫性を保った処理を保証する．

実際のフロアコントロールは，フロアコントロールポリシ **FloorNumber** に，入力可能なウィジェットインスタンスの最大数を設定し，`floor` コマンドまたはフロアコントロールメソッドを用いて権限を取得，譲渡することによって行う．この時，**FloorNumber** を 1 に設定すれば，複製ウィジェット間で入力の競合が避けられ，データの一貫性を保証できる．

5.5 イベントフィルタ機構

Mogul の持つイベントフィルタ機構は，Java 言語によるシングルユーザアプリケーション中でイベントハンドラを登録する記述を流用して実現する．従って，登録されたイベントハンドラが扱うイベントのみを，複製ウィジェット間で共有する．

図 10に，イベントフィルタの概念図を示す．図中のイベントフィルタは，円柱状の形状で表される *ActionEvent* のみをネットワークに配送し，複製ウィジェット間で共有するよう設定されている．従って他の形状を持つイベントはフィルタを通過せず，結果としてネットワークへの負荷を軽減している．

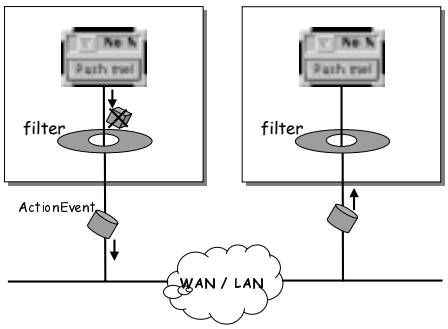

図 10: イベントフィルタ概念図

6 評価

本節では，本研究との関連性が特に高い **Collawt** と **Groupkit** を踏まえた上での機能面，基本性能面での評価について述べる．

6.1 機能評価

Collawt および **Groupkit** は双方ともマルチユーザツールキットとしての側面を有しており，本研究との関連が特に高い．これらのツールと Mogul との機能比較の結果を表 2に示す．

Collawt(表中 C) は，イベント共有と基本的なフロアコントロールを実現するマルチユーザツールキットである．Mogul と同様，Java 言語を用いて実装されているため，オブジェクト指向言語の機能拡張性を継承している．しかし **Collawt** は，Mogul が実現しているホスト位置透過的なウィジェット複製および移動の基本機能を持たない．また，Mogul を用いて実質的な追加行数およびプログラミング制約なしに分散マルチユーザアプリケーションを構築できるのに対して，**Collawt** におけるウィジェットライブラリは Java AWT ウィジェット群と名前衝突を起こしやすいため，プログラミング時の制約が高い．

Groupkit(表中 G) は Tcl/Tk を用いて実装されたマルチユーザツールキットである．ただし，マルチユーザツールキットとしての側面よりは，協調作業支援アプリケーション構築用 API としての性格が強い．**Groupkit** の提供するウィジェット群は Mogul ウィジェットと同様，描画状態共有が可能であるが，ウィジェットのホスト間移動や複製に関しては実現していない．

表 2: 他システムとの機能比較表

項　　目	C	G	Mogul
機能拡張性	易	易	易
ウィジェット複製	不可	不可	可
ウィジェット移動	不可	不可	可
イベント共有	可	可	可
フロアコントロール機構	有	有	有
アクセスコントロール機構	無	有	有
位置透過性	無	無	有
プログラミング	難	難	易

6.2　基本性能評価

基本性能特性の把握を目的として，Mogul の提供する各機能の所要時間を測定した．表 3に示す計算機を用いて，図 11に示す測定環境上で測定を行なった．なお，測定環境上で 2 台の計算機は同一のネットワークセグメント上に存在する．

表 3: 測定環境

項目	マシン 2(miro)	マシン 3(dali)
CPU	UltraSPARC 168MHz×2	UltraSPARC-II 248MHz×2
主記憶	64MB	512MB
OS	Solaris 2.5.1	Solaris2.5.1

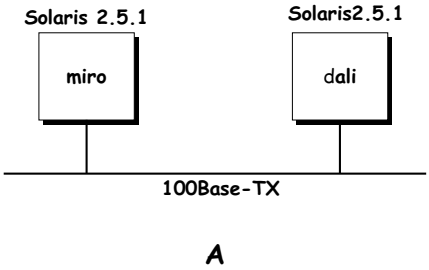

図 11: 測定環境

6.2.1　イベント共有性能

複製ウィジェット数とイベント共有性能との関連を明らかにするため，*SharedButton* ウィジェットの複製を 1 個から 4 個作成し，合計 2 個から 5 個のウィジェットインスタンス間でイベント共有を 100 回繰り返した．この時イベント共有のためにネットワークに送信されたデータは 1139 バイトであった．

図 12に，所要時間を項目別に色分け表示した測定結果を示す．RMI では IP マルチキャストを用いることができないため，複製ウィジェット数が増加すると所要時間も直線的に増加する．

図 12に示す項目の内，所要時間のおよそ 66%はイベントオブジェクトの直列化に費やされており，10%は RMI による遠隔メソッド呼び出しに費やされている．以上の結果から，実装方式の変更などによってスケーラビリティを確保する必要があると言える．

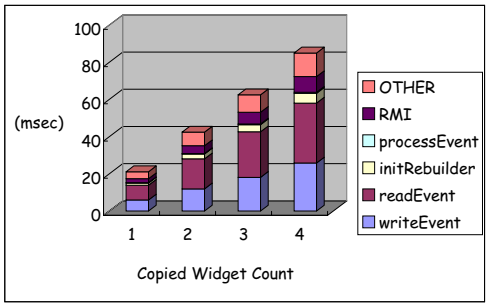

図 12: イベント共有所要時間詳細

6.2.2　ウィジェットのホスト間複製性能

ホスト *miro* 上に複製対象ウィジェットを生成し，post コマンドを用いて他ホストへ複製した時の所要時間を計測した．所要時間を項目別に色分け表示した測定結果を図 13に示す．

図 13 中で，所要時間の 65%程度を占めるのは，リモートホスト上でのウィジェットバス復元に関する項目である．ウィジェットバス復元は，各複製ウィジェット間の RMI による接続を復元する処理であり，ウィジェットバス復元所要時間のほとんどは RMI に起因して発生する．図 13中ではこの他に，描画に要する項目が全体所要時間の 15%を占めている．

以上の結果から，所要時間の多くを RMI に起因するオーバーヘッドが占めており，今後の実装最適化時にはプロセス間通信機構に関する考慮を要すると言える．

7　今後の課題

今後の課題として，IP Multicast への対応を含めたプロセス間通信機構の最適化が必要である．さらに，ホスト間移動に伴った，計算機環境への動的適応機構および QOS 制御機構の実現が必要である．特に，ホスト位置透過的なファイルアクセス機構，表示装置性能差に応じたメディア変換機構，および CPU やメモリ容量によって規定される計算機性能を基本とした QOS 制御機構の実現が挙げられる．

これらの諸課題を含めて現在，以下の機構の設計および実装作業が進んでいる．

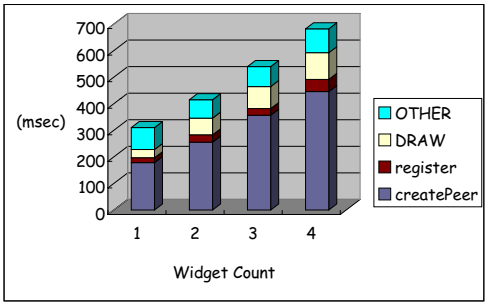

図 13: post-It 所要時間詳細

Mogul-FS
ホスト間を移動しながら特定ホスト上のファイルに継続的にアクセスするための動的分散ファイルシステムである．ビデオ回覧板などが容易に構築可能となる．

Mogul-MS
Mogul ウィジェットのホスト間移動に際して，ホスト間の表示装置性能差等に応じた画像および音声のフォーマット変換，色調変換などを行うためのアプリケーションレベルゲートウェイである．

Mogul-QOS
Mogul ウィジェットのホスト間移動に際して，計算機性能変化の検知を可能とするためのイベント処理機構である．

8 おわりに

本稿では，分散マルチユーザアプリケーションの特性を考察し，その構築ツールとして，位置透過型分散共有ツールキットライブラリ (**Mo**bile **G**raphical **U**ser Interface **L**ibrary: Mogul) を提案した．

Mogul を用いて構築したアプリケーションは，プログラマによる追加的コードなしに，動作中アプリケーションのウィンドウおよびウィジェットに対するリモートホストへの位置透過的な複製や移動，複製ウィジェット間でのイベント共有を可能とする．マルチユーザツールキットや共有ウィンドウを実現する既存のさまざまなツールと比較して，ウィンドウおよびウィジェットの移動をアプリケーションの動作中にも行えることが Mogul の第一の特徴である．さらに，これらの機能を特定の集合点ホストに対する登録などを経ずに，ホスト位置透過的に実現できることが第二の特徴として挙げられる．

定性評価においては，**Collawt** と **Groupkit** を取り上げ機能性の観点から比較評価した．**Collawt** および **Groupkit** は，Mogul が持つウィジェットの複製および移動機能を持たなかった．Mogul の提供する位置透過性や，シングルユーザアプリケーションに対する追加コード行数などの観点から，Mogul は分散マルチユーザアプリケーションのプログラマに対する負担をより軽減できることを実証した．

今後，ホスト間移動に伴った，計算機環境への動的適応機構および QOS 制御機構の実現が必要である．同機構の実現によって，分散マルチメディアアプリケーション構築時のプログラマに対する負担軽減，携帯情報端末 (PDA) を対象とした分散マルチユーザアプリケーション構築が可能となる．

謝辞

本研究を遂行するにあたり協力していただいた，慶應義塾大学 徳田・村井・楠本・中村研究会の諸氏に感謝致します．

参考文献

[1] Ellis C.A., Gibbs S.J., Rein G.L., “Groupware: Some issues and experiences”, Communications of the ACM 34, 1991.

[2] 中澤 仁, “位置透過型分散共有ツールキットライブラリの実装と評価”, 慶応義塾大学 徳田・村井・楠本・中村研究会卒業論文, 1997.

[3] Sun Microsystems Inc., “The JAVA Language Overview.”, Sun Microsystems, Inc., 1995.

[4] Prasun Dewan, “Tools for Implementing Multiuser User Interfaces”, Trends in Software: Issue on User Interface Software, Wiley, Volume 1, pages 149-172, 1993.

[5] Mark Stefik, Gregg Foster, Daniel G.Bobrow, Kenneth Kahn, Stan Lanning, Lucy Suchman, “Beyond the chalkboard: Computer support for collaboration and problem solving in meetings”, CSCM, 30(1):32-47, January 1987.

[6] Hussein Abdel-Wahab, Mark Feit, “XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration”, Proceedings of IEEE TriComm ’91: Communications for Distributed Applications & Systems, page159-167, April 1991.

[7] Abdel-Wahab, H., Jeffay K., “Issues, Problems and Solutions in Sharing X Clients on Multiple Displays”, Journal of Internetworking Research & Experience, pp1-15, Vol.5, No.1, March 1994.

[8] H. Abdel-Wahab, B. Kvande, S. Nanjangud, “Using Java for Multimedia Collaborative Applications”, Proceedings of the 3rd International Workshop on Protocols for

[9] Sun Microsystems, Inc., “Remote Method Invocation Specification”, 1996.