計算グリッドの透過的利用をサポートするポータルシステム

福留　貴俊†　　本多　弘樹†　　弓場　敏嗣†

†電気通信大学　大学院情報システム学研究科

グリッドポータルシステム上のアプリケーションは、計算資源によって提供されるライブラリ等の下位グリッドアプリケーションと、それらをいくつか組み合わせて構成される上位グリッドアプリケーションからなっている。本稿では、以下の 3 つの特徴を持つ開発環境に重点を置いたグリッドポータルシステムの設計とプロトタイプ実装について述べる。①下位グリッドアプリケーション提供者が下位グリッドアプリケーションを登録しやすい。②上位グリッドアプリケーション開発者に対して充実した上位グリッドアプリケーション開発環境を提供する。③上位グリッドアプリケーション利用者に対して位置透過、アクセス透過なインタフェースを持つ上位グリッドアプリケーションを提供する。

1. はじめに

広域ネットワーク上の計算資源を利用することで、高い性能を得るためのグリッドが注目を集めている。

グリッドポータルシステムは、Web ブラウザのような汎用的なクライアントアプリケーションをフロントエンドに用いて、各種計算資源を提供するグリッドサービスを利用するためのシステムである。これによって、ユーザはクライアントアプリケーションのインストールを行うことなく、インターネット上からグリッドサービスを利用できる。Globus Toolkit[1)]等により利用基盤が整い、ユーザの増加が期待される今、ユーザがグリッドサービスを手軽に利用できるようにするためのグリッドポータルシステムの果たす役割は大きくなってきている。

グリッドポータルシステム上で開発されるグリッドアプリケーションは以下の 2 つに分類することができる。

① **下位グリッドアプリケーション：**計算資源上で提供されるライブラリなどのグリッドアプリケーション

② **上位グリッドアプリケーション：**下位グリッドアプリケーションをいくつか組み合わせて構成され、エンドユーザに直接利用されるグリッドアプリケーション

また、グリッドポータルシステムのユーザをグリッドアプリケーションとの関係によって以下の 3 つに分類する。

① **下位グリッドアプリケーション提供者：**計算資源を持ち下位グリッドアプリケーションを提供するユーザ

② **上位グリッドアプリケーション開発者：**上位グリッドアプリケーションを開発するユーザ

③ **上位グリッドアプリケーション利用者：**開発された上位アプリケーションで計算を実行するエンドユーザ

上位グリッドアプリケーション開発者は多くの場合、下位グリッドアプリケーションを利用することになる。しかし、下位グリッドアプリケーション提供者でなければ、下位グリッドアプリケーションを利用するための情報を正確に把握することは難しい。この関係は上位グリッドアプリケーション開発者と上位グリッドアプリケーション利用者の間にも当てはまる。

本稿で述べるグリッドポータルシステムは、容易なグリッドアプリケーションの開発と利用を目指したものである。さらに本グリッドポータルシステムは開発環境に重点を置き、以下の特徴を有する上位グリッドアプリケーションのフレームワークを提供する。

① **容易な登録：**下位グリッドアプリケーション提供者に対して、下位グリッドアプ

Portal system supporting transparent use of computational Grid

Takatoshi Fukudome†
Hiroki Honda†
Toshitsugu Yuba†
†The Graduate School of Information Systems, University of Electro-Communications.

リケーションの登録を容易に行えるようにする。

② **充実した開発環境：**上位グリッドアプリケーション開発者に対して、充実した開発環境を提供する。

③ **透過的利用：**上位グリッドアプリケーション利用者に対して、位置透過、アクセス透過な上位グリッドアプリケーションの利用を可能にする。

2. 設計方針

2.1 前提条件

本稿のグリッドポータルシステムは以下の条件を前提としている。

- Globus, Ninf[2)]などを用いて下位グリッドアプリケーションが提供されている。
- グリッドポータルシステム自体は計算資源の管理、監視を行わない。計算資源の管理はそれぞれの下位グリッドアプリケーション提供者が行う。
- 下位グリッドアプリケーションを利用するには、計算資源の位置、採用しているグリッドシステム、利用するためのプログラムの記述方法等の様々な情報を知らなければならない。従って、下位グリッドアプリケーション提供者ではない上位グリッドアプリケーション開発者が、下位グリッドアプリケーションの情報を正確に把握することは難しい。
- 上位グリッドアプリケーション開発者はセキュリティ管理、Web インタフェースのプログラミング等に労力を割きたくない。或いはそれらの専門知識を持たない。
- ユーザ間で上位グリッドアプリケーションを共有できない場合、上位グリッドアプリケーション利用者は自分の開発した上位グリッドアプリケーションしか利用できない。従って、ユーザ間で上位グリッドアプリケーションを共有する仕組みが必要である。
- 上位グリッドアプリケーションの利用において、操作方法が統一されていて、入力すべき情報と操作手順が自明であれば、アクセス透過性が高く、上位グリッドアプリケーション利用者にとって利用しやすい。

2.2 前提条件を踏まえた 3 つの設計方針

前提条件より、グリッドポータルシステムに求められるのは、アプリケーションの透過性、開発者の手間を極力減らす開発環境であると考える。以下の 3 つの設計方針により、この要求を満たすグリッドポータルシステムを開発する。

① 位置透過、アクセス透過な入出力インタフェース

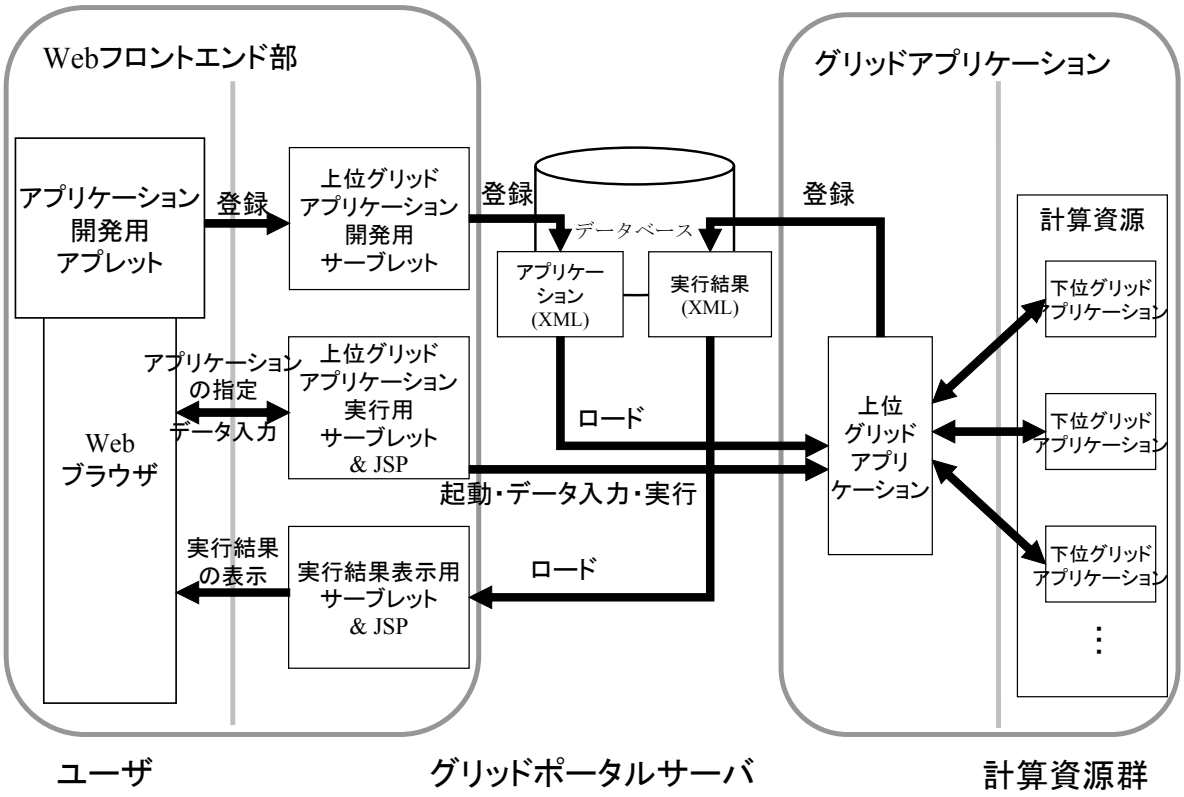
下位グリッドアプリケーションの利用が難しいのは、下位グリッドアプリケーションが位置透過、アクセス透過ではないからである。サーバの位置情報、採用しているグリッドシステムの違い、引数の順番や名前等、要因は様々である。本稿のグリッドポータルシステムでは、下位グリッドアプリケーションを上位グリッドアプリケーションでラップすることで、この問題を解決する。従って、上位グリッドアプリケーションは位置透過、アクセス透過に利用できる入出力インタフェースを持つ必要がある。

位置透過を実現するために、上位グリッドアプリケーションを可能な限り少ない情報で特定可能にする。アクセス透過を実現するために、上位グリッドアプリケーションの呼び出し方法を統一し、アプリケーション、各引数に対する解説ドキュメントを入出力インタフェースが持つ。これにより、呼び出し方、引数の名前や順番に迷うことなく、アクセス透過に利用できる。

② マルチプログラミングモデル

全ての処理を 1 つのプログラミングモデルで記述する場合、その記述は複雑なもの、或いは極めて機能の制限されたものとなる。それぞれのアプリケーションに最適なプログラミングモデルを、複数のプログラミングモデルから選択すること（マルチプログラミングモデル）が可能となれば、複雑さ、機能制限の問題を解決できる。例えば、いくつかのパラメータを入力するだけの単純なプログラミングモデルや、ビジュアルプログラミングを行うプログラミングモデル等の特定の用途に特化したプログラミングモデルも採用できる。しかし、それぞれのプログラミングモデルが異なる入出力インタフェースを持てば、アクセス透過性を失う。

複数のプログラミングモデルとアクセス透過性を両立するために、上位グリッドアプリケーションのアプリケーション本体と入出力インタフェースを分離する。入出力インタフェースを共通のコンポーネントとする。アプリケーション本体の記述を各プログラミング



モデルに任せることで、複数のプログラミングモデルの使い分けを可能とする。

③ GUI を用いたインタラクティブな開発環境

マルチプログラミングモデルの特徴を活かすには CUI では不十分であり、GUI を用いる必要があると考える。また、CUI よりも GUI の方が直感的な操作が行えるので、上位グリッドアプリケーション開発者の負担、ケアレスミスを軽減することができる。

2.3 その他の設計方針

前節の 3 つの設計方針を補強するためのいくつかの設計方針を示す。

- **ライブラリコール機能**

上位グリッドアプリケーションの透過性を利用して、上位グリッドアプリケーションから他の上位グリッドアプリケーションを呼び出せるようにする（ライブラリコール）。これによって、上位グリッドアプリケーション開発者は下位グリッドアプリケーションのことを意識せずにすむ。上位グリッドアプリケーションを通して、その下位グリッドアプリケーションを自分の上位グリッドアプリケーション開発に利用することができる。

- **グループ管理**

上位グリッドアプリケーション開発に他の上位グリッドアプリケーションを利用できたとしても、それを他のユーザから提供してもらう仕組みがなければ、結局自分の持つ下位グリッドアプリケーションしか利用できないことになる。そこで、ユーザ間でアプリケーションを共有するための仮想的なグループを設ける。グループ内で上位グリッドアプリケーションを共有することで、そのグループ内のアプリケーション開発権限を持つ下位グリッドアプリケーション提供者から下位グリッドアプリケーションの計算能力を提供してもらうことができる。

3. グリッドポータルシステムの概要

3.1 システムアーキテクチャ

本稿のグリッドポータルシステムは、Java と XML によって実装されている。

グリッドポータルシステムの核となるポータルサーバはサーバサイド Java のおける MVC (Model-View-Controller)アーキテクチャの基本に基づいている。ロジック部分に Java

Beans、出力に JSP(Java Server Pages)、入力にサーブレット、を用いて、グリッドポータルシステムを構成している。ロジック部分は、サービス全体の管理を行う中枢部と、各ユーザ、各グループへのサービスを担当するユーザサービス提供部、グループサービス提供部から構成されている。

ユーザ情報、上位グリッドアプリケーションの記述、計算結果等のデータは全て XML の形で管理される。全てのデータが XML であるため、データベースにはデータベースの構造に XML を利用している Xindice[3]を用いた。

本稿のグリッドポータルシステムの機能は主に以下の 3 つである。

① **上位グリッドアプリケーション開発機能：**上位グリッドアプリケーションの開発を行う。
② **上位グリッドアプリケーション実行機能：**上位グリッドアプリケーションで計算を行う。
③ **実行結果表示機能：**上位グリッドアプリケーションの実行結果を見る。

図 1 に 3 つの機能を中心にして見た場合の全体図を示す。

3.2 シングルサインオン

ログイン時に認証を受ければ、以後必要な場合は自動的に認証を行う、シングルサインオンの機能は各ユーザサービス提供部とそれを管理する中枢部によって実現されている。ユーザはユーザ名とパスワードを入力しログインすることで、認証を受ける。

なお、現在はユーザ情報を持つ XML ファイル上に平文でパスワードをおいている状態だが、今後 MyProxy[4] 等の認証システムを取り入れていく予定である。このようにセキュリティ関係はまだ脆弱であるため、認証システムを持つグリッドシステムには現在対応していない。従って、現在対応可能なグリッドシステムは、Ninf のような認証を必要としないグリッドシステムだけである。

3.3 ユーザサービス提供部

ユーザへのサービスは各ユーザサービス提供部によって提供される。ユーザサービス提供部は、シングルサインオンでログインした時に生成される。そして、ログオン中のユーザサービス提供部は、上位グリッドアプリケーション開発、上位グリッドアプリケーション実行、実行結果表示などの機能をユーザに提供する。

ユーザがログオフするときに、そのユーザのユーザサービス提供部も破棄される。しかし、ログオン中にユーザの起動した上位グリッドアプリケーションが実行中であれば、ユーザサービス提供部は破棄されない。全ての上位グリッドアプリケーションの実行が終了した時点で、ユーザサービス提供部は破棄される。

3.4 グループサービス提供部

グループにサービスを提供するグループサービス提供部も、ユーザサービス提供部とほぼ同じ機構を持っている。ただし、グループの持つ上位グリッドアプリケーションを常時利用可能にするために、グループサービス提供部の生成は自動で行われる。

グループには、グループに所属するユーザのグループに対する権限を設定する機能がある。権限には以下の 3 つがある。

① **管理者権限：**認証やグループ情報の操作権限
② **開発者権限：**グループ共有の上位グリッドアプリケーションを開発する権限
③ **利用者権限：**グループ共有の上位グリッドアプリケーションで計算を行う権限

上位グリッドアプリケーションの実行結果のデータは、実行した上位グリッドアプリケーション利用者によって管理される。よって、グループには実行結果のデータを管理する機能はない。

4. 上位グリッドアプリケーションの構造

上位グリッドアプリケーションは、全ての上位グリッドアプリケーションで共通のコンポーネントと、各プログラミングモデル独自のコンポーネントと、各上位グリッドアプリケーションの XML の記述、から成り立っている。上位グリッドアプリケーションの XML の例を図 2 に示す。以下の機構によって、上位グリッドアプリケーションは透過性を提供している。

- **引数情報**

arg タグは上位グリッドアプリケーションの引数を示している。arg タグの情報を基に、引数の型と配列構造からなるメタデータが作成される。メタデータを用いて、入力データの上位グリッドアプリケーション内で使える形への変換と、型と配列構造のチェックを行う。

このようにして出来た引数のデータを用いて、上位グリッドアプリケーションを実行する。

- **解説文**

  document タグは上位グリッドアプリケーションとその引数に対する解説文を意味する。この解説文により、上位グリッドアプリケーション実行時に、上位グリッドアプリケーションの機能とその引数の意味が自明なユーザインタフェース、入出力インタフェースの作成が可能になる。

- **ライブラリコール情報**

  call タグはライブラリコールのためのものである。グループ名、アプリケーション名の他に、簡単な引数情報を持っており、呼び出す上位グリッドアプリケーションの引数情報と同定を行う。

- **アプリケーション本体**

  上位グリッドアプリケーションの本体の XML の記述は、各プログラミングモデルに依存する。引数のメタデータ、ライブラリコールの情報、上位グリッドアプリケーションの本体の XML の記述、を利用して上位グリッドアプリケーションを構築する。そして、引数のデータに対して、上位グリッドアプリケーションを実行する。

5. 上位グリッドアプリケーションの開発と利用

上位グリッドアプリケーション開発、上位グリッドアプリケーション実行、実行結果表示、の 3 つの段階に分けてグリッドポータルシステムの動作について説明する。

5.1 上位グリッドアプリケーション開発

上位グリッドアプリケーションの開発は、Java アプレットを用いた GUI の開発環境を用いて行われる。

各上位グリッドアプリケーション共通のコンポーネントには以下の機能がある。

- 入出力インタフェース作成（図 3）
  - ➢ 上位グリッドアプリケーションの名前・ドキュメントの入力
  - ➢ 入出力される引数の名前・型・ドキュメントの入力
- ライブラリコールのための情報の取得
  - ➢ 既存の上位グリッドアプリケーションの入出力インタフェースの情報の取得
- 開発した上位グリッドアプリケーション

```
<?xml version="1.0"?>
<application name="Matrix_Multi" type="PAD"
 developer="fukudome">
<documentation>Matrix Multiplication</documentation>

<arg name="m" type="int" dim="0" inout="in" hasValue="false">
  <document>m times trial</document>
</arg >
<arg name="n" type="int" dim="0" inout="in" hasValue="false">
  <document>n * n Matrix</document>
</arg >
<arg name="a" type="double" dim="3" inout="in"
hasValue="false">
  <array dim="0" type="var" var="m"/>
  <array dim="1" type="var" var="n"/>
  <array dim="2" type="var" var="n"/>
  <document>input a[m][n][n]</document>
</arg >
<arg name="b" type="double" dim="3" inout="in"
hasValue="false">
  <array dim="0" type="var" var="m"/>
  <array dim="1" type="var" var="n"/>
  <array dim="2" type="var" var="n"/>
  <document>input b[m][n][n]</document>
</arg >
<arg name="c" type="double" dim="3" inout="out"
hasValue="false">
  <array dim="0" type="var" var="m"/>
  <array dim="1" type="var" var="n"/>
  <array dim="2" type="var" var="n"/>
  <document>output c[m][n][n]</document>
</arg >

<call name="g1:mmul" group="g1" application="mmul">
  <callarg name="n" type="int" dim="0" inout="in"/>
  <callarg name="a" type="double" dim="2" inout="in"/>
  <callarg name="b" type="double" dim="2" inout="in"/>
  <callarg name="c" type="double" dim="2" inout="out"/>
</call>
<PAD>
<var name="i" type="int" dim="0" inout="not io"
hasValue="false"/>
<fornode>
<forelements comp="l">
  <begin mode="value" type="int" value="0"/>
  <iterator mode="var" var="i" arraydim="0"/>
  <step mode="value" type="int" value="1"/>
  <end mode="var" var="m" arraydim="0"/>
</forelements>
<body>
  <callnode call="g1:mmul">
    <nodearg callarg="n" mode="var" var="n" arraydim="0"/>
    <nodearg callarg="a" mode="var" var="a" arraydim="1">
      <array dim="0" mode="var" var="i"/>
    </nodearg>
    <nodearg callarg="b" mode="var" var="b" arraydim="1">
      <array dim="0" mode="var" var="i"/>
    </nodearg>
    <nodearg callarg="c" mode="var" var="c" arraydim="1">
      <array dim="0" mode="var" var="i"/>
    </nodearg>
  </callnode>
</body>
</fornode>
</PAD>
</application>
```

引数情報

ライブラリコール情報

各プログラミングモデルによるアプリケーション本体の記述

図 2 上位グリッドアプリケーションの XML

のポータルサーバへの登録

プログラミングモデルを追加する際は、以下のものを追加すればよい。

- プログラミングモデルのインタフェース
- プログラムの XML 化メソッド
- XML の書式

プログラミングモデルには、以下のようなものが考えられる。

- 図 4 のように、パラメータを入れるだけで、下位グリッドアプリケーションを上位アプリケーション化するモデル
- 図 5 のようなビジュアルプログラミングモデル
- スクリプト言語を用いたプログラミングモデル

現在、図 4 のパラメータ入力モデルでは Ninf の利用が可能となっている。

開発された上位グリッドアプリケーションは XML ファイルに変換され、ポータルサーバへと送信される。ポータルサーバは送信された XML ファイルを、実行可能かどうか確認した後、上位グリッドアプリケーションとしてデータベースに登録する。

5.2 上位グリッドアプリケーション実行

上位グリッドアプリケーションの実行は、Web インタフェースを動的に生成して行う。上位グリッドアプリケーション利用者に提示される画面は、全て共通コンポーネントによって作成されたものである。利用している上位グリッドアプリケーションがどのようなプログラミングモデルで開発されたかを、上位グリッドアプリケーション利用者が意識することはない。プログラミングモデルを追加するときは、XML から実行用のインスタンスを生成、実行する部分を作成すればよい。以下に実行時の動作を示す。

(1) 上位グリッドアプリケーション利用者がグループ名、アプリケーション名を入力する。

(2) グループ名、アプリケーション名によって指定された上位グリッドアプリケーションを起動する。既に他の利用者によって起動されたインスタンスが残っている場合は、そのインスタンスを利用する。

(3) 上位グリッドアプリケーションの入出力インタフェースのメタデータから、動的に作成された入力フォームが返ってくる。入力フォームは上位グリッドアプリケーションと引数に関する解説文を含んでいる。図 6 に入力フォームを示す。

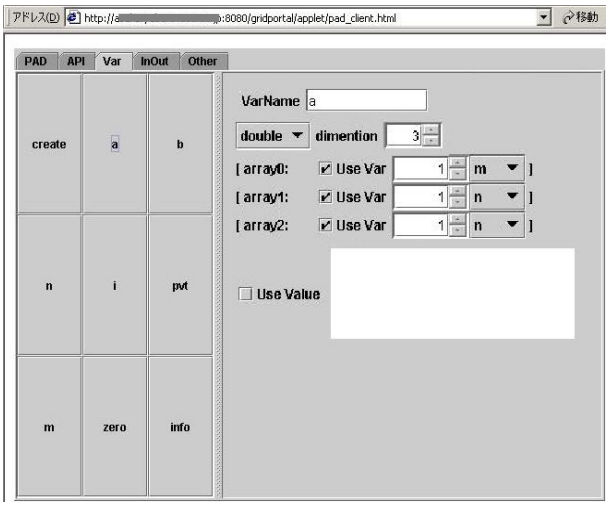


図 3 入出力インタフェース作成画面

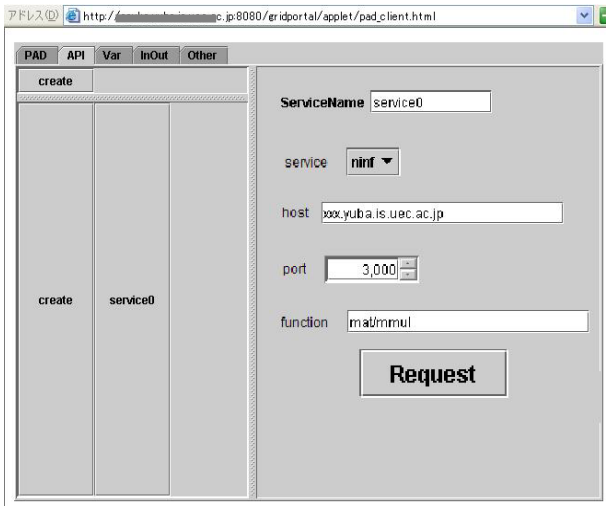


図 4 パラメータ入力モデル

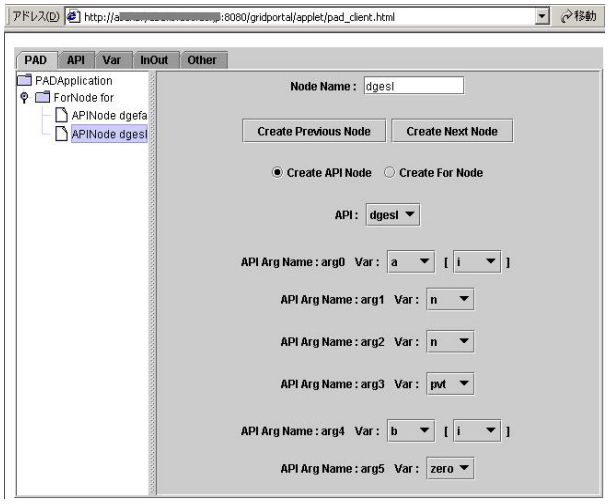


図 5 ビジュアルプログラミングモデル

(4) 入力フォームにデータを入力し、結果の取得に必要な ID を記入して、送信する。
(5) 上位グリッドアプリケーションの実行が開始される。上位グリッドアプリケーションの実行は利用者に対して隠蔽される。この時点で、利用者は他の作業に移ることができる。
(6) 上位グリッドアプリケーションが終了したら、上位グリッドアプリケーションの情報と実行結果の引数データを XML に変換して、データベースに登録する。

5.3 実行結果表示

実行結果は XML で表現されている。実行結果は入力フォームに入力した ID によって管理されている。この ID を入力することによって、上位グリッドアプリケーションの実行結果を閲覧する。上位グリッドアプリケーションが実行中、或いはエラーが発生した場合も通知する。図 7 に実行結果を表示している画面を示す。

6. 関連研究

既存の研究を見てみると、GridPort[5]では、GridPort を基盤とした Hot Page[6]、LAPK portal[7]等の複数のポータルサービスを、シングルサインオンで利用できる（マルチポータル）。本稿のマルチプログラミングモデルとの大きな違いは、グリッドアプリケーション実行時のインタフェースがそれぞれのポータルサービスで異なり、アクセス透過性がないことである。また、グリッドアプリケーション、計算資源等の位置透過性に関しても考慮されていない。

JiPANG[8]では、Jini で下位グリッドアプリケーションをラップすることで、ユーザに対して位置透過、アクセス透過なインタフェースを持つ下位グリッドアプリケーションを提供する。しかし、透過性を維持するために、計算資源のあるサイト内に計算資源を管理するサービスを稼動させなければならない。グリッドポータルシステムが、下位グリッドアプリケーション提供者の計算資源を管理するという構造は、グリッドポータルシステムの計算資源に対する介入が強く、下位グリッドアプリケーション提供者に受け入れられない可能性がある。また、下位グリッドアプリケーションを登録するためには XML を記述しなければならない。本ポータルでは、下位グリッドアプリケーション登録者は GUI を用いて、

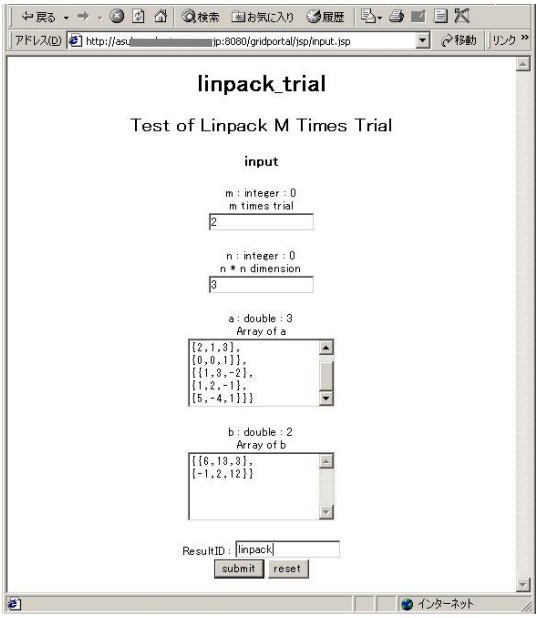


図 6 入力フォーム

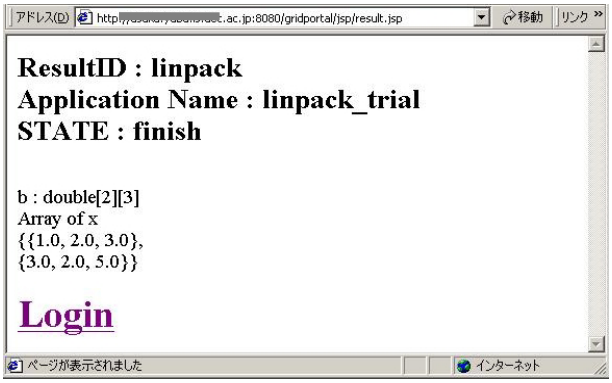


図 7 実行結果表示画面

下位グリッドアプリケーションを登録可能なので、直接 XML を記述する必要はない。

7. まとめと今後の課題

本稿では、上位グリッドアプリケーションの開発環境に重点を置いた、計算グリッドの透過的利用をサポートする、グリッドポータルシステムの設計とプロトタイプ実装について述べた。まだ実装の途中ではあるが、いくつかの簡単なプログラミングモデルの実装と、上位グリッドアプリケーションの開発、実行に成功している。

容易な登録と充実した開発環境は、マルチプログラミングモデルと GUI を用いたインタラクティブな開発環境により実現した。位置透過性は、グループ名とアプリケーション名による単純な名前空間により実現した。アク

セス透過性は、異なるプログラミングモデル間で入出力インタフェースを統一することで実現した。

以下に今後の課題について述べる。

- **セキュリティ**

早急に解決すべき課題としてセキュリティの確保が挙げられる。上位グリッドアプリケーションの開発、実行環境の構築が優先したため、セキュリティの実装が遅れている。取り分け MyProxy 等の認証システムの実装を急ぐ必要がある。実用的なグリッドシステムでは認証システムは必須であり、このままではインターネット環境での実用的なグリッドシステムに対応できない。また、現在の平文でパスワードを持っている状態は、セキュリティの面で問題がある。本グリッドポータルシステムは様々なグリッドシステムを計算資源の対象としている。いくつもの認証システムを柔軟に使い分けられる実装にする必要がある。

- **スケジューリング**

本稿の実装では、スケジューリングは行っていない。計算資源のグリッドシステムを限定していないので、ポータルサーバによる一元的な各計算資源間のスケジューリングは難しい。個々のプログラミングモデル、上位グリッドアプリケーションが、各計算資源間のスケジューリングを行う必要がある。

- **Globus への対応**

本稿の実装では、Globus は一切使われていない。今後、Globus も実装に取り込む予定である。その中でも最初に取り込むのが認証システムとなる。しかし、完全に Globus に頼った実装は、他のグリッドシステムを取り込むときや、Globus 自体に問題が発生したときに、問題になると思われる。あくまで取り込むグリッドシステムの一つと捉えている。

- **マルチプログラミングモデル**

共通コンポーネントのインタフェースにも改善の必要がある。また、プログラミングモデルの追加も必要である。WebFlow[9]のように GUI 上でグリッドのワークフローを記述するプログラミングモデルは、本稿のグリッドポータルシステムと親和性が高く、ぜひ提供したいと考えている。ある程度の数のプログラミングモデルがそろったら、マルチプログラミングモデルに対する評価を行う必要がある。

参 考 文 献

1) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit , *International Journal of Supercomputer Applications*, Vol. 11, No. 2, pp. 115-128 (1997). http://www.globus.org/.
2) Ninf Project http://ninf.apgrid.org/.
3) Xindice http://xml.apache.org/xindice/.
4) Novotny, J., Tuecke, S. and Welch, V.:An Online Cre-dential Repository for the Grid: MyProxy. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press (2001).
5) Thomas, M., Mock, S., Dahan, M., Mueller, K. and Sutton, D.: The GridPort Toolkit: a System for Building Grid Portals, *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press (2001).
6) Hot Page https://hotpage.npaci.edu/.
7) LAPK portal https://gridport.npaci.edu/LAPK/.
8) 鈴村豊太郎, 松岡聡, 中田秀基, 関口智嗣: Jini を用いた Computing Portals System の開発, 情報処理学会報告, 2000-HPC-81, pp.57-62 (2000).
9) Akarsu, E., Fox, G. C., Furmanski, W., Haupt, T.: WebFlow – High-Level Programming Environment and Visual Authoring Toolkit for High Performance Distributed Computing, *In Proceedings of Supercomputing '98* (1998).