Grid ポータル構築ツールキット Ninf-Portal

中田秀基[†,††] 齊藤真幸[††] 鈴村豊太郎[††]
田中良夫[†] 松岡聡[††,†††] 関口智嗣[†]

広域に分散した各種資源を集合的に使用して大規模計算を行う計算機構 Grid が注目を集めている．多様な管理主体に属する資源から構成される Grid 環境上のプログラムを多くのユーザに使用させるためには，ポータルと呼ばれる機構が必要である．このため，Grid へのポータルを構築するためのツールキットがいくつか提案されている．しかしこれらのツールキットを用いる場合でも，ポータル構築者がフロントエンドとなるポータルのユーザインタフェース部，バックエンドとなる実際の Grid プログラムの 2 つを記述しなければならず，ポータル構築者の負担が大きい．我々は，前者に対して XML ベースのユーザインタフェース生成系を，後者に対して Grid RPC システムである Ninf-G を使用することでプログラマの負荷を軽減するポータル開発キットを提案する．さらに，提案したシステムを用いて，実用的なプログラムをポータル化し有効性を確認した．

Grid Portal Toolkit Ninf-Portal

Hidemoto Nakada,[†,††] Masayuki Saito,[††]
Toyotaro Suzumura,[††] Yoshio Tanaka,[†] Satoshi Matsuoka[††,†††]
and Satoshi Sekiguchi[†]

As the Grid proliferates as the next-generation wide-area high-performance computing infrastructure, end-user Grid interfaces in the form of “Grid Portals” is becoming increasingly important, especially computational scientists and engineers. Although several Grid portal toolkits and proposals have been proposed, a Grid Portal creator must construct and deploy both the user interface and the application portions of the Grid Portal, resulting in considerable programming efforts. We aim to easen this burden by applying the state-of-the-art Web/XML interface generation technologies for the former, and the Ninf-G GridRPC system for easily “Gridifying” exisiting applications for the latter, and realizing their seamless integration. The resulting system which we call the “Ninf Portal” allowed concise description and easy deployment of a sample application on the Grid with very small programming efforts.

1. はじめに

高速なネットワークの普及によって，広域に分散した各種資源を集合的に使用して大規模計算を行う計算機構 Grid が現実的となった．Grid は一般に多様な管理主体に属した多数の各種計算資源によって構成されているため，使用には煩雑さがともない，一般の計算科学者などのユーザが直接使用することは難しい．このため，Grid の使用を容易にするための Grid ポータルと呼ばれる機構の必要性が認識されつつあり，NPACI の HOTPAGE[1)] をはじめとしてすでにいくつかのポータルが公開されている．

Grid ポータルとは，特別なソフトウェアサポートを持たないクライアントから Grid 上のアプリケーションを使用することを可能にするシステムである．特別なソフトウェアを使用しないという要請から，インタフェースとしては Web ブラウザを用い，プロトコルには http（もしくは https）を用いるのが一般的である．ユーザは，Web ブラウザでポータルにログインし，Grid アプリケーションと使用資源，使用データを指定して実行する．Grid アプリケーションの実行状態の監視と制御や，使用データや設定ファイルのアップロードや結果のダウンロードもできる．

Grid ポータルには，通常の Web アプリケーションの機能に加えて，Grid 特有の機能を実装しなければならない．これをサポートするために，Grid ポータ

† 産業技術総合研究所
National Institute of Industrial Science and Technology (AIST)
†† 東京工業大学
Tokyo Institute of Technology
††† 国立情報学研究所
National Institute of Informatics

ルを構築するためのツールキットがいくつか提案されている[2),3)]．これらのツールキットは，シングルサインオンや資源の検索などのポータルの基本的な機能を実現しているが，Grid アプリケーションのユーザインタフェースとなるデータ入力のページの記述や，実際に起動される Grid アプリケーションの記述に関しては，サポートを提供していない．このため，ポータル構築者がフロントエンドとなるポータルのユーザインタフェース部と，バックエンドとなる実際の Grid プログラムの 2 つを記述しなければならず，ポータル構築者の負担が大きい．

本稿では，フロントエンドの構築を XML ベースのユーザインタフェース生成でサポートし，バックエンドの構築を Grid RPC[4)] システムである Ninf-G[5)] でサポートすることでポータル構築者の負荷を軽減するポータル開発キット Ninf-Portal を提案した．さらに，提案したシステムを用いて，実用的なプログラムをポータル化し有効性を確認した．

本稿の構成は以下のとおりである．まず 2 章で，Ninf-Portal の設計について述べる．3 章では Ninf-Portal の重要なコンポーネントである Ninf-G について述べる．4 章でフロントエンド部の実装について詳説する．5 章で Ninf-Portal を用いたポータルの実例を示し，本システムの有効性を示す．6 章には関連する研究を示す．

2. Ninf-Portal の設計

2.1 Grid ポータルの要件

Grid ポータルには以下の機能が要請される．

認証と認可 認証とはユーザの身元を確認すること，認可とは身元に基づいて権限を与えることである．Grid ポータルにおける認証はシングルサインオンでなければならない．シングルサインオンとは Grid システム一般に要請される性質で，複数の資源に対して一度の操作で継続的な使用権を確立することである．Grid ポータルでは，この操作を http だけを用いて実現できなければならない．

データのアップロードと結果のダウンロード Grid アプリケーションの計算で使用するデータファイルや，Grid アプリケーションが結果として出力したファイルをユーザの計算機へダウンロードすることができなければならない．Grid ポータルでは，この機能は Web の HTML ページを用いて実現する必要がある．

プログラムの起動と実行制御 Grid 上の各計算資源上でプログラムを起動し，さらに起動したプログラムのモニタリングと停止処理などの制御をユーザに提供しなければならない．

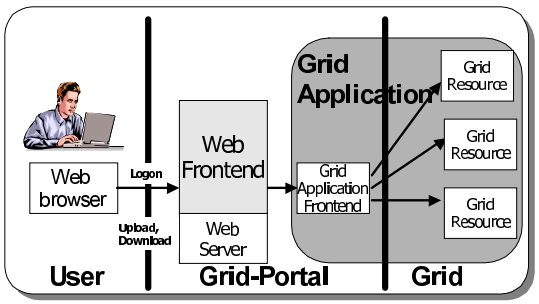

図 1 Grid ポータルの一般形
Fig. 1 General configuration of Grid portals.

使用資源の検索と指定 ユーザが，Grid 上の計算資源やデータ資源を検索して，どれを使用するかを決定することができなければならない．

2.2 一般的な Grid ポータルの構成

図 1 に Grid ポータルの一般的な構成を示す．Grid ポータルは大きく分けて，Web フロントエンド部と，バックエンドとなる Grid アプリケーション部から構成される．

Web フロントエンド部 ユーザインタフェースや認証などを行う部分．ユーザインタフェースは，使用資源の検索や指定を行ったり，Grid アプリケーションへの入力を指定したりするために用いる．この部分は，通常の HTML や，CGI やサーブレットなどの Web サーバと連動する動的コンテンツの記述フレームワークで記述される．

Grid アプリケーション部 実際の計算を行う部分．Web フロントエンド部で行われた認可によって取得した権限に基づいて，Grid 上の資源を活用して計算を行う．この部分は Globus などの Grid ツールキットを用いて記述される場合が多い．

既存のポータル構築ツールキットでは，Web フロントエンド部のフレームワークだけが提供されている．このためポータル構築者が，Web フロントエンドのユーザインタフェース部（具体的にはユーザインタフェースとなる HTML と CGI）と，Grid アプリケーション全体を記述しなければならない．

2.3 Ninf-Portal の概要

Ninf-Portal は，既存のポータル構築ツールキットの機能に加え，ユーザインタフェース部と Grid アプリケーション部の記述をサポートする機能を持つ．

ユーザインタフェース部に関しては，インタフェース情報から自動的にユーザインタフェースページを作成する機能を提供する．

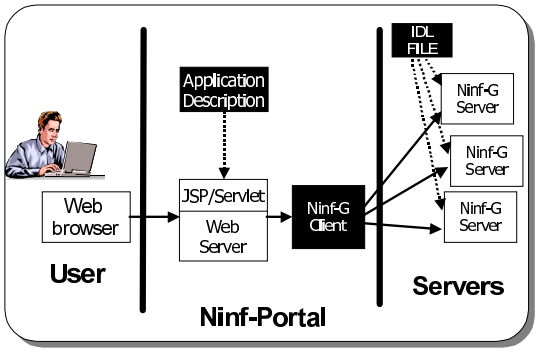

図 2 Ninf-Portal の概要
Fig. 2 Overview of the Ninf-Portal.

Grid アプリケーションに関しては，Grid RPC システム Ninf-G を提供することで，記述を容易にしている．Ninf-G に関しては 3 章で詳しく述べる．

Grid アプリケーションは，コマンドラインから起動されるプログラムとして記述する．Grid アプリケーションの入出力はコマンドラインの引数として指定する．入力情報としては，文字列と入力ファイルを指定することができる．出力情報としては，ファイルへの出力と標準出力への出力が使用できる．

Web フロントエンド部は，ユーザが Web インタフェースで入力した情報を使用して引数列を作り，Grid アプリケーションを起動する．Grid アプリケーションが出力した情報は，Web インタフェースを介してユーザに提供される（**図 2**）．

2.4 Web フロントエンド部の設計

フロントエンドは一般の Web アプリケーションとほとんど同じ構造となるので，一般の Web アプリケーション用フレームワークが使用できる．現在広く用いられているフレームワークとしては CGI（Common Gateway Interface）とサーブレット，JSP がある．

CGI は，Web サーバと外部プログラムの通信様式を規定したものである．外部プログラムの言語は規定しないが，Perl の CGI ライブラリが整備されているため，一般には Perl で書くことが多い．CGI には，事実上すべてのサーバで機能するという利点があるが，反面，リクエスト時にプロセス起動をするため実行コストが大きい，セッションの管理が煩雑といった欠点がある．

サーブレットは，Java で記述したプログラムで HTML を生成する機構である．サーブレットを実行する Java VM のプロセスはサーバと同時に起動しており，リクエスト時にはスレッドを割り当てるだけなので，実行コストが小さい．さらに，Java の提供する膨大な API を自由に使用してプログラミングできるため，拡張性に優れている．また，セッションが明示的にサポートされており，任意のデータオブジェクトをセッションに保持することができる．

JSP（Java Server Pages）は，HTML 中に Java コードを記述することを可能にする機構である．JSP はサーブレットに変換して実行されるため，サーブレットとの相性がよい．

本システムでは，サーブレットと JSP を使用して実装する．サーブレットを用いるのは，1) セッション情報を管理することが容易，2) Globus クライアントの Java 用 API である Java CoG キット[6]の整備が進んでおり，Grid 情報へのアクセスが容易，3) Java の広範な API 群により，他のコンポーネント（データベースなど）との連携が容易，だからである．

2.5 ユーザインタフェース部の自動生成

Web アプリケーションのユーザインタフェース部は一般に，HTML ページと，その HTML ページからサブミットされたデータを処理するサーブレット（もしくは CGI）で構成される．

Ninf-Portal では，HTML ページにはポータル構築者が記述した XML ファイルから自動的に生成した JSP を使用し，データ処理には，任意のサブミットデータを処理することのできる汎用のデータ処理サーブレットを使用する．ポータル構築者は，XML ファイルを記述するだけでよい．

2.6 シングルサインオンの実現

Ninf-Portal のバックエンド部となる Ninf-G では，Globus の認証機構である GSI（Grid Security Infrastructure）を用いて認証を行う．GSI は，ユーザの証明書によって署名された証明書（代理証明書）がユーザと同じアイデンティティを持つと見なす，証明書の委譲と呼ばれる機構によってシングルサインオンを実現している．したがって，何らかの方法でユーザの証明書を Grid ポータルに渡すことができれば，Grid ポータルに対するシングルサインオンが実現できる．

ユーザの代理証明書を Grid ポータルに渡す最も直感的な方法としては，HTTP でアップロードする方法が考えられる．しかしこの方法では，ログイン時にファイルを指定する必要があり操作が煩雑となるし，代理証明書内の秘密鍵が（https で暗号化されているとはいえ）ネットワークを通ることになり，安全上好ましくない．

この問題を解決するため，本システムでは MyProxy[7]を用いた．MyProxy は，代理証明書のレポジトリとなるサーバである．これを用いると，代理証明書を安全な第三者サーバに預けておき，他のホス

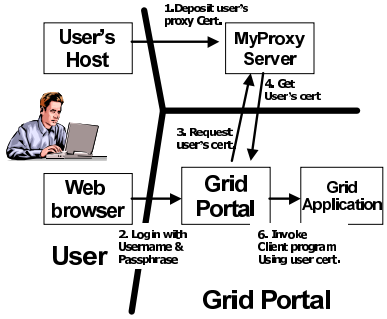

図 3 MyProxy の動作
Fig. 3 Overview of the MyProxy.

トからそれをユーザ名とパスフレーズで取り出すことができる．この過程は GSI で保護されているうえ，秘密鍵が転送されることがないように設計されているので安全である．

図 3 に MyProxy の動作の様子を示す．ユーザは事前に MyProxy サーバに代理証明書を預けておく．ユーザがポータルにログインする際に，ユーザ名とパスワードを入力すると，ポータルは MyProxy サーバにアクセスして，代理証明書を取り出す．それ以降はその代理証明書を用いて，Grid へアクセスする．

Java の CoG キットには MyProxy のクライアントが含まれており，JSP やサーブレットからも容易に使用することができる．

3. Ninf-G の概要

ここでは，Ninf-Portal の重要なコンポーネントである Ninf-G について詳しく述べる．

Ninf-G は Grid RPC システムである Ninf[8]を Globus ツールキット[9]を使用して再実装したものである．Globus ツールキットは，Grid 環境で必要となる機構のプロトコルと API を定めて実装したものであり，Grid 上でのデファクトスタンダードの地位を確立しつつある．Globus を利用することで，実装を容易にできるだけでなく，相互運用性が確保できる．

3.1 Grid RPC システム Ninf

Ninf は，サーバ側に存在するライブラリを，クライアント側のプログラムから簡単な操作で呼び出すことを可能にする，Grid RPC と呼ばれるシステムの 1 つである．**図 4** に，行列の乗算を行うクライアントプログラムの例を示す．通常の関数呼び出しを `Ninf_call` を用いて書き換えるだけで，計算をサーバ側で行うことができる．さらに非同期呼び出しのインタフェース `Ninf_call_async` を用いれば，複数のサーバ上の計算ルーチンを並列に呼び出すことができ，非常に容易に

```
double A[N*N], B[N*N], C[N*N];
....
Ninf_call("sample/mmul", N, A, B, C);
....
```

図 4 Ninf クライアントプログラムの例
Fig. 4 Ninf client program example.

```
Module sample;
Define mmul(IN  int    N,
    IN  double A[N*N],
            IN  double B[N*N],
            OUT double C[N*N])
Required "mmul_lib.o"
Calls "C" mmul(N, A, B, C);
```

図 5 Ninf IDL の例
Fig. 5 Ninf IDL example.

アプリケーションを並列化することができる．

サーバ側では，計算ルーチンを公開しなければならない．これには簡単な IDL を書くだけでよい．行列の乗算ルーチンを公開するための IDL 記述を**図 5** に示す．

IDL には，公開する関数の引数の型情報とモード情報とサイズ，実際の計算を行うライブラリ関数の名前とそれが収められているオブジェクトファイルを指定する．引数のサイズは，別の引数を用いた算術演算で指定することができ，さまざまな数値演算ライブラリに柔軟に対応することができる．

この IDL を IDL コンパイラでコンパイルすると，スタブ関数と make ファイルが生成される．この make ファイルを用いて make を実行すれば，サーバ側の実行ファイルが生成される．

3.2 Ninf-G と Globus

Ninf-G は Globus ツールキットの以下のライブラリを使用している．

GRAM GRAM（Globus Resource Allocation Manager）はサーバ側で実行ファイルの起動を安全に行うモジュールである．ベースに GSI を用いており，証明書を用いた認証とマップファイルによるユーザアカウントへのマッピングを行う，いわば安全な inetd である．Ninf-G では，このモジュールを用いてサーバ側実行ファイルを起動する．

MDS MDS（MetaComputing Directory Service）は資源の情報を蓄積するためのディレクトリサービスであり，プロジェクト全体の情報を管理する GIIS（Grid Index Information Service）と，サイトローカルな情報を管理する GRIS（Grid Resource Information Service）の 2 階層の LDAP

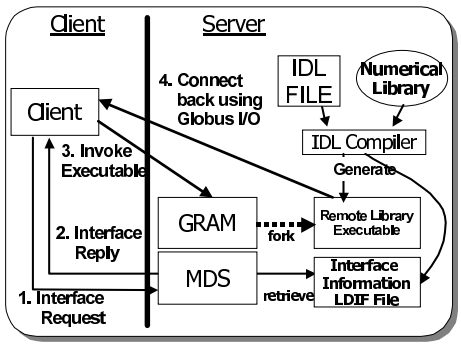

図 6 Ninf-G の概要
Fig. 6 Overview of the Ninf-G.

サーバで実装されている．

Ninf-G では，このモジュールを用いてサーバ側実行ファイルの場所やインタフェース情報の公開と取得を行う．

Globus-I/O Globus-I/O は，GSI を用いた安全な通信を実現するモジュールである．API としては，通常の TCP/IP に近い read/write モデルであるが，Globus の提供する擬似的なスレッドを使用する都合上，コールバックを用いたノンブロッキング I/O が提供されているのが特徴である．

Ninf-G では，このモジュールを用いてクライアントとサーバ側実行ファイルとの通信を行う．

Ninf-G の概要を**図 6** に示す．Ninf-G の動作は，サーバ側のリモートライブラリの登録と，クライアント側からの呼び出しに大きく分けられる．次節以降ではこれらについて詳しく述べる．

3.3 リモートライブラリの構築および登録

Ninf-G でリモートライブラリを作成するには，

(1) ライブラリ関数の呼び出し情報を Ninf IDL と呼ばれる言語で記述する，
(2) Ninf IDL コンパイラで IDL ファイルをコンパイルし，スタブルーチンを作成する，
(3) スタブルーチンと関数本体をリンクし，リモートライブラリを作成する，
(4) MDS（GRIS）に，作成されたリモートライブラリを登録する，

といった作業が必要になる．(3)，(4) の動作は，IDL コンパイラが生成する make ファイルで自動化されている．MDS には，リモートライブラリのインタフェース情報と，サーバ上でのパスを登録する．MDS に情報を登録するためには，LDAP サーバの要請する LDIF（LDAP Data Interchange Format）と呼ばれる形式で情報を出力するプログラムを作成し，MDS の設定ファイルに記述する必要がある．Ninf-G では，特定のディレクトリ（`${GLOBUS_LOCATION}/var/gridrpc/`）に LDIF フォーマットのファイルを置き，それに簡単な置換を行うフィルタを MDS の設定ファイルに追加している．

3.3.1 LDIF ファイルの作成

Ninf-G の IDL コンパイラはスタブソースファイルと make ファイルのほかに，LDIF ファイルを作成するために用いる引数情報ソースファイルを生成する．生成された make ファイルを用いて make コマンドを実行すると，スタブソースファイルをコンパイルしてリモートライブラリを生成するほかに，引数情報ソースファイルをコンパイル，実行して引数情報を XML 形式で表したファイルを生成する．そして，さらにその XML 形式のファイルをもとに，リモートライブラリに関する情報（リモートライブラリ名，パスおよび引数情報など）を GRIS に登録するために必要となる LDIF 形式のファイルを生成する．LDIF は簡単なテキスト（ASCII）形式のフォーマットで，属性値がテキストで表現できない場合やサイズが大きすぎるような場合には，base64 encoding を行う．図 5 に示した行列の乗算を行うリモートライブラリの IDL を，IDL コンパイラおよび make ファイルで処理した結果得られる LDIF ファイルを**図 7** に示す．1 行目に“`__ROOT_DN__`”と書かれているが，この部分は前述したように MDS に登録されたフィルタプログラムによって，適切な DN（distinguished name）に置換される．

3.3.2 MDS への登録

上述の手順で生成される LDIF ファイルは，make を実行したディレクトリに生成される．MDS に登録するためには，このファイルを前述の指定ディレクトリに設置する必要がある．これも make ファイルに記述されているので，`make install` コマンドを実行するだけで自動的に（`${GLOBUS_LOCATION}/var/gridrpc/`）にコピーされる．これで，MDS への登録が完了したことになる．

3.4 リモートライブラリの実行

Ninf-G のリモートライブラリ実行は図 6 に番号で示した過程で行われる．

(1) クライアントが MDS に引数情報とパス情報をリクエスト．
(2) 引数情報とパス情報を取得．
(3) パス情報を用いて GRAM にリモートライブラリの起動をリクエスト．
(4) リモートライブラリからクライアントに Globus I/O を用いて接続．

```
dn: GridRPC-Funcname=sample/mmul, Mds-Software-deployment=GridRPC-Ninf-G, __ROOT_DN__
objectClass: GlobusSoftware
objectClass: MdsSoftware
objectClass: GridRPCEntry
Mds-Software-deployment: GridRPC-Ninf-G
GridRPC-Funcname: sample/mmul
GridRPC-Module: sample
GridRPC-Entry: mmul
GridRPC-Path: /home/ninf/tests/sample/_stub_mmul
GridRPC-Stub:: PGZ1bmN0aW9uICB2ZXJzaW9uPSIyMjEuMDAwMDAwIiA+PGZ1bmN0aW9uX25hbWUgbW9kdWxl
 PSJwZXJmIiBlbnRyeT0icGluZ3BvbmciIC8+IDxhcmcgZGF0YV90eXBlPSJpbnQiIG1vZGVf
  ...（以下省略）
```

図 7 mmul の LDIF ファイル
Fig. 7 LDIF file for mmul.

3.4.1 引数情報とパス情報の取得

Ninf-G ではリモートライブラリのパスと引数情報は MDS に登録されている．クライアントはライブラリ名を鍵として MDS を検索してこれらの情報を取得する．この MDS に対する検索結果はクライアント側でキャッシュされ，可能な限り再利用される．これによって比較的大きい MDS 検索のコストを低減している．

3.4.2 リモートライブラリの起動

リモートライブラリの起動は GRAM を用いて行う．この際に，MDS から取得したリモートライブラリのパス情報を使用する．また，引数として次の段階でリモートライブラリからのコールバックを受けるための受信用のポートを指定する．このポートはあらかじめオープンしておく必要がある．

3.4.3 リモートライブラリからクライアントへのコールバック

リモートライブラリは起動すると，引数からクライアントのアドレスとポートを取得する．そして，Globus I/O を用いてこのアドレスとポートに接続する．その後，クライアントとリモートライブラリはこのポートを用いて，互いに通信する．

このとき，クライアント側でオープンされているポートは認証と認可に制約を設け，自分自身の証明書を持っているプログラム以外からは接続できないようになっている．したがって，コールバックを装った第三者が接続してしまうことはない．

4. Web フロントエンド部の実装

4.1 Web フロントエンドの機能

Grid ポータルの Web フロントエンドでは，Grid アプリケーションに与える入力データの指定と，出力データの提供を行う．

入力データの指定方法としては，1) 直接文字列を与える，2) ローカルマシンに存在するファイルをアップロードする，3) テキストフィールドに記述した内容をファイルとしてアップロードする，の 3 つの方法が考えられる．

出力データの提供方法としては，1) 直接ブラウザ画面に表示する，2) ファイルとしてダウンロードできるようにリンクを表示する，の 2 つの方法が考えられる．

Ninf-Portal ではこれらの方法を，Grid アプリケーション IDL で記述することができる．

```
<!ELEMENT Application (Information*,
          ArgumentFormat, Argument*)>

<!ELEMENT Information (name?,location?,
          manufacturer?,description?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT manufacturer (#PCDATA)>
<!ELEMENT description (#PCDATA)>

<!ELEMENT ArgumentFormat (#PCDATA)>

<!ELEMENT Argument (type,info,
                    method?,comment?)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT info (#PCDATA)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

図 8 Grid アプリケーション IDL の DTD
Fig. 8 DTD of the Grid application IDL.

4.2 Grid アプリケーション IDL

Grid アプリケーションの記述は，XML を用いた Grid アプリケーション IDL で行う．XML をベースとすることで，コンパイラに既存の XML パーザを使用することができるため，実装が容易になった．本システムでは，コンパイラは Java 言語の DOM レベルの XML パーザを用いて記述した．Grid アプリケーション IDL の DTD を**図 8** に示す．

location で Grid アプリケーションのバイナリファイルのパスを指定する．ArgumentFormat はバックエンドの Grid アプリケーションを呼び出す際の引数を指定する．Argument で各フィールドの名前，型，処理方法を指定する．複数の Argument を並べる際の順番は任意である．

Argument の type でフィールドの型を指定する．型

としては直接フィールドに値を入れる `string`，`int`，`float`，ファイルのアップロード，ダウンロードを指定するための `inputfile`，`outputfile` が用意されている．

Argument の method 要素で，入出力ファイルの指定・処理方法を記述する．入力ファイルの場合は `upload` と `field` のいずれかを指定する．`upload` であれば，ローカルなファイルシステムからアップロードする．`field` とすれば，テキストフィールドに書いた内容をファイルとしてアップロードする．出力ファイルの場合は，`display` と `link` のいずれかを指定する．`display` では，Grid アプリケーション終了時の画面に直接ファイルの内容が表示される．`link` とすると，終了時画面にはリンクだけが形成される．ユーザはブラウザの機能を用いてファイルをダウンロードすればよい．

生成される JSP ページは，大きく 2 つの部分から構成される．1 つは，ユーザの入力を促すフォームフィールドを含む HTML 部である．もう 1 つは，サブミットされた情報の処理方法を定めるメタデータをセッションに収めるための Java コード部である．JSP を用いるので，この 2 つの内容を単一のファイルに収めることができるうえ，Java コード部をコンパイルする必要がない．

フォームフィールドの入力部では，JavaScript による簡単な型チェックが行われる．たとえば整数型と宣言されているフィールドに小数点数が書かれているとサブミット時に変更を促す．これによって，早期のエラー回避が可能になる．

4.3 JSP ページとサーブレットの連携

データ処理サーブレットが，サブミットされたデータを処理するためには，個々のデータに関するメタデータが必要になる．このメタデータには，データフィールドの名前や，そのデータが単なる値なのかアップロードデータなのか，などの情報が収められる．メタデータは Java コードの形で JSP ページに埋め込まれる．JSP ページは，埋め込まれたメタデータをセッションに保存する．ユーザが，JSP で生成された HTML ページのフィールドにデータ入力してサブミットすると，リクエストはサーブレットに引き継がれる．サーブレット部では，セッションからメタデータを取り出し，その情報に基づいてサブミットされてきたデータを解釈し，Ninf-G で作成されたバックエンドプログラムを起動する．この様子を**図 9** に示す．

4.4 汎用データ処理サーブレット

汎用データ処理サーブレットは以下の機能を持つ．

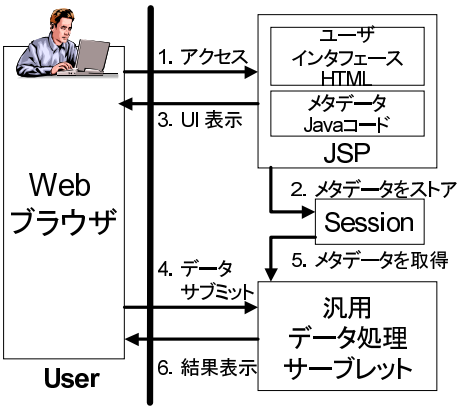

図 9 JSP とサーブレットの連携
Fig. 9 JSP and servlet cooperation.

- アップロードデータの読み出し
- Grid アプリケーションの実行
- 結果ページの作成

Grid アプリケーションの実行時には，ログイン時に MyProxy から取得したユーザの証明書を使用しなければならない．データ処理サーブレットは，証明書をテンポラリディレクトリに書き出し適切なパーミッションを設定する．次に，環境変数に証明書ファイルのパスを設定してから Grid アプリケーションを起動する．

5. 実アプリケーションを用いた Ninf-Portal の評価

ここでは Ninf-Portal を用いた実際の Grid ポータルを構築を通じて Ninf-Portal の有用性を評価する．アプリケーションとしては BMI 固有値問題[10]を用いる．BMI 固有値問題は数値最適化問題の 1 つであり，計算量が大きく並列計算に適しているため Grid のアプリケーションの 1 つとして期待されている．

Ninf-Portal による Grid ポータルの構築には Ninf-G を用いた Grid アプリケーション自身の記述と Grid アプリケーション IDL による Web インタフェースの実現が必要となる．

5.1 Grid アプリケーションの記述

Ninf-G による Grid アプリケーションの記述は，1) クライアント，2) リモートライブラリ，3) リモートライブラリ IDL，の 3 つの記述によって行われる．既存のアプリケーションを Grid アプリケーション化する場合には，アプリケーションをクライアント部とリモートライブラリ部に分離し，リモートライブラリ IDL を記述するだけでよい．

Web インタフェース部と Grid アプリケーションは

```
for(i = 0; i < nServer; i++){
  sprintf(ninfURL[i],
      "ninf://%s/BMI/BBRootCalc",
      serverList[0]);
}
  ..... 略 .....
for(i = 0; i < nServer; i++){
  ..... 略 .....
  pid[i] =
    Ninf_call_async(
      ninfURL[i],
      xDIM, yDIM, nBlock, blockStruct, Sparse,
      NonzeroElement, nData, Data, nodeData[i],
      zValue, nBranch, nDepth, epsilon, nLeaf,
      &nRemNodes[i], resultData[i], &zUpdate[i],
      zValueData[i]);
  ..... 略 .....
}
```

図 10 BMI コードの一部
Fig. 10 Program fragment of the BMI client code.

```
Define BBRootCalc (
  IN int xDIM,
  IN int yDIM,
  IN int nBlock,
  IN int blockStruct[nBlock],
  IN int Sparse,
  IN int
   NonzeroElement[xDIM + yDIM + xDIM * yDIM + 1]
                 [nBlock],
  IN int nData,
  IN double Data[nData],
  IN double Bound[xDIM * 2 + yDIM * 2],
  IN int nBranch,
  IN int nDepth,
  IN double epsilon,
  IN int nLeaf,
  OUT int nRemNodes[],
  OUT double
   resultData[nLeaf][xDIM * 3 + yDIM * 3 + 2],
  OUT int error[],
  OUT double zValueData[xDIM + yDIM + 1]
)
Required "BBRootCalc.o libBMI.a sdpa.a
          meschach.a"
Calls "C++" BBRootCalc(
    xDIM, yDIM, nBlock, blockStruct, Sparse,
    NonzeroElement, nData, Data, Bound,
    nBranch, nDepth, epsilon, nLeaf,
    nRemNodes, resultData, error, zValueData);
```

図 11 BMI の IDL の一部
Fig. 11 IDL file for the BMI.

引数を通じて情報を交換する．BMI 固有値問題の場合には，入力として計算のタイプを示す文字列と，計算対象となるデータファイルを与えなければならない．出力は標準出力にプリントアウトする．

図 10 にクライアントプログラムのリモートライブラリを呼び出している部分を抜き出したものを示す．このプログラムでは `Ninf_call_async` を用いて複数のサーバで同時にリモートライブラリを実行することで，並列実行を行っている．一般に並列実行には複数の通信路を同時に非同期に制御することが必要となり，プログラムが煩雑になる．Ninf-G ではシステムがこの複雑さを隠蔽するため，プログラマは容易に分散並列プログラムを記述することができる．

```
<?xml version="1.0"
      encoding="shift_jis"?>
<!DOCTYPE application SYSTEM
                "JSPGenerator.dtd">
<Application>
  <!-- Application Information -->
  <Information>
    <name> BMI Application </name>
    <location>
      /home/saito/work/BMIClientC/BMISolver
    </location>
    <manufacturer>
      Kento Aida
    </manufacturer>
    <appdescription>
      BMI Application Portal
    </appdescription>
  </Information>

  <ArgumentFormat>
    -t $type $uploadfile
  </ArgumentFormat>

  <!-- Arguments Information -->
  <!-- first Argument -->
  <Argument>
    <argname> uploadfile </argname>
    <type>inputfile</type>
    <info> InputFile </info>
    <method> upload </method>
    <description>
      inputfile of data
    </description>
  </Argument>

  <!-- second Argument -->
  <Argument>
    <argname> type </argname>
    <type> string </type>
    <info> type </info>
    <description>
      type of program
    </description>
  </Argument>
</Application>
```

図 12 Grid アプリケーション IDL 記述の例
Fig. 12 Grid application IDL example.

図 11 に BMI で用いたリモートライブラリの IDL を示す．前半はリモートライブラリインタフェースの定義である．後半ではリモートライブラリが実際に行う関数呼び出しを定義している．8 行目の `NonzeroElement` などは 2 次元の配列であるが，そのサイズは先行する引数に対する四則演算で定義されている．このように転送するデータサイズを動的に決定しなければならない場合でも，容易に記述することができる．

5.2 Grid アプリケーション IDL の記述

上述したように，BMI 固有値問題では，問題のタイプを示す文字列と，設定情報を収めたファイルの 2 つを入力とし，出力は標準出力に出力される．標準出力はデフォルトでユーザに提示されるため，IDL に記述する必要はない．したがって IDL には，2 つの入力に関する情報を記述すればよい．

```
<%@ page import="bmi.*" %>

<html>
<head>
  <title>BMI Application portal</title>
</head>
<body>

<%
  int    argnumber = 2;
  String argformat = "-t $type $uploadfile";
  String executablepath =
     "/home/saito/work/BMIClientC/BMISolver";
  String args[]       = {"inputfile","string"};
  String namerow[]    = {"uploadfile","type"};
  String filemethod[] = {"upload","null"};
  PortalApplicationDescription obj =
      new PortalApplicationDescription(
             argnumber,
             argformat,
             executablepath,
             args,
             namerow,
             filemethod);
  session.setAttribute("inputs",obj);
%>
<br>
      Welcome to BMI Application Portal!
<br>
<form action="/bmi/servlet/Portal"
      name="MyForm"
      method=post
      ENCTYPE="multipart/form-data"
      onSubmit="return checkData(this)">
 <table border = 3 align = center>
   <tr>
     <td>InputFile</td>
     <td><input type = file name = arg0></td>
   </tr>
   <tr>
     <td>type</td>
     <td><input type = text name = arg1></td>
   </tr>
 </table>
 <center>
   <input type  = submit
          align = center
          value = "submit">
 </center>
 </form>
</body>
</html>
```

図 13 自動生成された JSP の例
Fig. 13 Automatically generated JSP code.

BMI 固有値問題の Grid アプリケーション IDL ファイルを**図 12** に示す．

このアプリケーション IDL ファイルをコンパイラで処理すると，ユーザからの入力を受け取るための JSP ファイルが生成される．この JSP ファイルを Web サーバの適切なパスに置くだけで，Grid ポータルのユーザインタフェース部が実現できる．

生成されたファイルを**図 13** に示す．「<%」と「%>」で囲まれた部分が Java コード部である．1 行目の Java コード部で，bmi の他のパッケージをロードしている．8 行目から始まる Java コード部で，この Grid アプリケーションに関するメタデータを

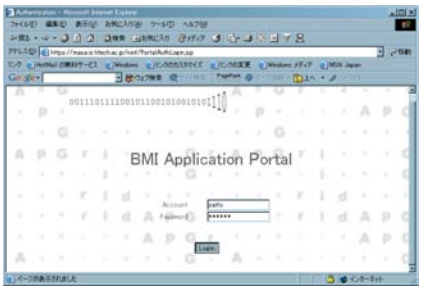

図 14 login 画面
Fig. 14 Screenshot of a login page.

PortalApplicationDescription というオブジェクトのインスタンスとして作成し，セッションの属性として設定している．Ninf-G で記述された Grid アプリケーションの起動や出力データの提示は，メタデータに収められた情報に基づいて，汎用データ処理サーブレットによって行われる．実際の入力インタフェース部となるのは，後半の form 文である☆．フォームのタグとして，IDL の info 要素で指定した情報が使用されている．

また，input の name 属性として，arg0，arg1 といった値が指定されている．汎用データ処理サーブレットは，この名前を用いて Web ブラウザからアップロードされたデータを処理する．

この生成された JSP からも分かるように，汎用データ処理サーブレットと連携するためには，メタデータを記述したり，規約を満たした属性名を使用したりしなければならず，これをポータル提供者が直接記述するのは煩雑である．Grid アプリケーション IDL から JSP を自動生成することでポータル提供者の負担が軽減されている．

5.3 実 行 例

実行の様子を**図 14**，**15**，**16** に示す．図 14 はユーザの認証を行うログイン画面である．図 15 は，ユーザにプログラムの実行に必要なデータを入力してもらうための画面である．この画面は Grid アプリケーション IDL をコンパイルしてできた JSP が生成している．Grid アプリケーション IDL で Argument 要素の method 要素で upload を指定しているので，ファイルをアップロードするためのフィールドができている．

結果を出力する画面を図 16 に示す．この例では Grid アプリケーションの標準出力を表示しているが，出力

☆ この例では，入力データが文字列型とファイル名型であるため JavaScript による型チェックは行われていないが，数値型の入力データを定義すると，チェック用の JavaScript コードが挿入される．

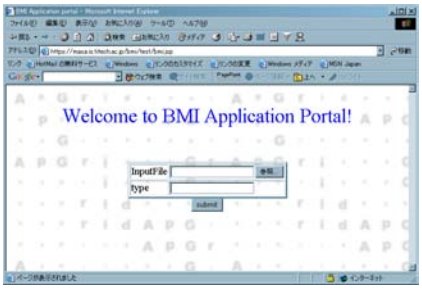

図 15 データ入力画面
Fig. 15 Screenshot of a data input form.

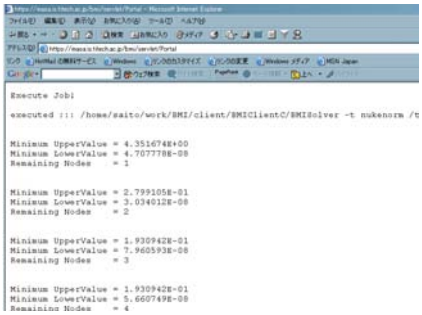

図 16 結果出力画面
Fig. 16 Screenshot of a result page.

ファイルをダウンロードするように指定することも可能である．

6. 関連研究

Grid ポータルのコンストラクションキットとしては，NPACI の GridPort [2] や，NLANR の Grid Portal Development Kit [3]，Indiana の XCAT Science Portal [11] などがある．GridPort は HotPage など実働している Grid ポータルの作成に広く用いられているツールキットである．大部分が Perl の CGI で書かれており，新しい Grid アプリケーションを登録する際には Perl でプログラミングしなければならない．バックエンドプログラムの記述には globus の API を直接使用する．

Grid Portal Development Kit は，Java で記述されたポータルである．バックエンドの Globus へのアクセス部を Java Bean の形で提供し，JSP の中から直接 Grid へアクセスすることを許す．しかし JSP という比較的短期間に終了することが期待されるページ生成系の中に，長時間の実行が予想される Grid アクセスを記述させることがよいことなのかは疑問である．

XCAT Science Portal は，ノートパッドと呼ばれるスクリプトの記述をユーザに許すポータルである．ノートパッドは Jython（Python を JavaVM で実行するもの）で CoG キットを用いて記述する．ユーザが複雑なコードを記述し，アップロードして実行することができる．しかし，Web サーバと Web アプリケーション実行部をクライアント側に置くため，クライアントのインストールコストが非常に高い．また，クライアントがさまざまなプロトコルで Grid に直接アクセスするため，クライアント側の資源とクライアントと Grid 間のネットワーク資源への要件が厳しくなり，使用できる環境が限定される恐れがある．

HTTP を利用した Grid 関連の動きとしては，Web Services のテクノロジと Grid テクノロジを融合させる OGSA [12] がある．OGSA は Web Services をベースに Grid のセキュリティを導入したものであり，一種のコンポーネントテクノロジである．OGSA は HTTP を利用するとはいえ，人間が直接使用するためのものではなく，通常のブラウザから使用することはできない．

7. まとめと今後の課題

本稿では，ポータル構築者の負荷を軽減するポータルシステム Ninf-Portal を提案した．フロントエンド部では XML を JSP に変換することで，ユーザインターフェイスを自動生成し，バックエンドでは，Grid RPC システムである Ninf-G を利用することで，最小限の手間での Grid プログラミングを可能にしている．このシステムを実装し，実用的なアプリケーションについて Grid ポータルを試作したところ，容易にポータルが実現できることが分かり，有効性が確認できた．

今後の課題としては以下があげられる．

- **情報提供機構の組み込み**
  Ninf-Portal 現在の Ninf-Portal は，プログラムの起動の簡素化に重点を置いており，ポータルの重要な機能である，Grid 環境の情報提供を行っていない．NWS などの情報収集プログラムやディレクトリサービスから情報を取得し，ユーザに提供する機能を組み込む必要がある．
- **Ninf-G クライアントの Java 化**
  現在は Ninf-G クライアントの API が C のみで提供されているため，Grid アプリケーションコードを C で記述する必要がある．このため，サーブレットコードから Grid アプリケーションを外部プログラムとして起動している．Ninf-G の Java API が提供されれば，Grid アプリケーションを Java で記述し，サーブレットの中に統合することができる．これによって，システムが単純にな

り，ポータルのインストールが容易になるだけでなく，ユーザの代理証明書を外部ファイルに書き出す必要がなくなり，脆弱性が低減できる．

- **他のバックエンドへの対応**
  現在の Ninf-Portal は Ninf-G をバックエンドとしているが，認証機構として GSI を使用しているものであれば，他のシステムをバックエンドとすることも可能である．6 章で述べた OGSA などのコンポーネントをバックエンドとすることも検討していく．
- **スクリプト機構の組み込み**
  現在のシステムでは，Grid アプリケーションを組み込むためには，ポータルサービスの提供者が Grid アプリケーションと，Grid アプリケーションインタフェースの記述を行わなければならない．また，ユーザはポータルサービス提供者が提供した Grid アプリケーションを使うだけあり，自由に Grid アプリケーションを組み合わせて使用することはできない．これに対処するために，XCAT のようにポータルにスクリプト実行機能を実装することが考えられる．Ninf-G の API をスクリプト言語に提供すれば，ユーザはポータル経由で自由に Grid アプリケーションを組み合わせて実行することが可能になる．この場合任意のコードをポータルで実行することになり，セキュリティが大きな問題となることが予想されるので，実現手法を吟味しなければならない．

参 考 文 献

1) NPACI HOTPAGE. https://hotpage.npaci.edu/
2) Thomas, M., Mock, S. and Boisseau, J.: Development of Web Toolkits for Computational Science Portals: The NPACI HotPage, *Proc. HPDC 9*, pp.308–309 (2000).
3) The Grid Portal Development kit. http://dast.nlanr.net/Projects/GridPortal/
4) 中田秀基，田中良夫，松岡　聡，関口智嗣：Grid RPC システムの API の提案，情報処理学会研究報告 HPC，Vol.2001, No.78, pp.37–42 (2001).
5) 田中良夫，中田秀基，平野基孝，佐藤三久，関口智嗣：Globus による Grid RPC システムの実装と評価，情報処理学会システムハイパフォーマンスコンピューティング研究会，No.77 (2001).
6) von Laszewski, G., Foster, I. and Gawor, J.: CoG Kits: A Bridge Between Commodity Distributed Computing and High-Performance Grids, A Java Commodity Grid Kit, *ACM 2000 Java Grande Conference* (2000).
7) Novotny, J., Tuecke, S. and Welch, V.: Initial Experiences with an Online Certificate Repository for the Grid: MyProxy, *Proc. 10th International Symposium on High Performance Distributed Computing* (*HPDC-10*), IEEE Press (2001).
8) Nakada, H., Sato, M. and Sekiguchi, S.: Design and Implementations of Ninf: towards a Global Computing Infrastructure, *Future Generation Computing Systems, Metacomputing Issue*, Vol.15, No.5–6, pp.649–658 (1999).
9) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications* (1997).
10) 合田憲人，二方克昌，原　辰次：並列分散計算システム上での BMI 固有値問題解法，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.42, No.SIG12(HPS4), pp.132–141 (2001).
11) Krishnan, S., Bramley, R., Gannon, D., Govindaraju, M., Indurkar, R., Slominski, A. and Temko, B.: The XCAT Science Portal, *Supercomputing 2001* (2001).
12) Foster, I., Kesselman, C. and J.M.N.S.T.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers.html#OGSA

(平成 14 年 1 月 29 日受付)
(平成 14 年 5 月 24 日採録)

中田　秀基（正会員）
昭和 42 年生．平成 2 年東京大学工学部精密機械工学科卒業．平成 7 年同大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．同年電子技術総合研究所研究官．平成 13 年独立行政法人産業技術総合研究所に改組．現在同所グリッド研究センター主任研究官．平成 13 年より東京工業大学客員助教授を兼務．グローバルコンピューティング，並列実行環境に関する研究に従事．

齊藤　真幸
昭和 52 年生．平成 14 年東京工業大学卒業．同年日本 アイ・ビー・エム 入社．在学中にグリッドのポータルシステムの研究に従事．

鈴村豊太郎

昭和 50 年生．平成 11 年東京工業大学理学部情報科学科卒業．平成 13 年同大学大学院情報理工学研究科・数理計算科学専攻修士課程修了．同大学院同専攻博士課程在学中．日本学術振興会特別研究員．グリッドのポータルシステムの研究に従事．

田中　良夫（正会員）

昭和 40 年生．平成 7 年慶應義塾大学大学院理工学研究科後期博士課程単位取得退学．平成 8 年技術研究組合新情報処理開発機構入所．平成 12 年通産省電子技術総合研究所入所．平成 13 年 4 月より独立行政法人産業技術総合研究所．現在同所グリッド研究センター基盤ソフトチーム長．博士（工学）．グリッドにおけるプログラミングミドルウェア，計算ポータル，およびテストベッド構築に関する研究に従事．IC'99 論文賞．ACM 会員．

松岡　　聡（正会員）

昭和 38 年生．昭和 61 年東京大学理学部情報科学科卒業，平成元年同大学大学院博士課程から，学情報科学科助手に採用，同大学情報工学専攻講師を経て，平成 8 年に東京工業大学情報理工学研究科数理・計算科学専攻助教授．平成 13 年 4 月に東京工業大学学術国際情報センター教授，平成 14 年より国立情報学研究所の客員教授を併任．博士（理学）（東京大学）．高性能システム，並列処理，グリッド計算，クラスタ計算機，高性能・並列オブジェクト指向言語処理系，などの研究に従事．ソフトウェアの技術開発によりコモディティ技術の大幅な活用で従来の 100 倍の計算パワーを計算科学に広域に提供することを目指す．現在進行中のプロジェクトは，(1) 産業技術総合研究所などと共同の種々のグリッド計算のプロジェクト（Ninf，GFarm プロジェクトなど），(2) 大規模コモディティPC クラスタ構築プロジェクト（Presto クラスタ群），ならびに (3) 計算環境に適合・最適化を目指す Java 言語の開放型 Just-In-Time コンパイラ OpenJIT，JDSM 等．平成 8 年度情報処理学会論文賞，平成 11 年情報処理学会坂井記念賞受賞．国際学会 ISOTAS'96，ECOOP'97，ISCOPE'99 のプログラム委員長や Reflection2001 の大会委員長などを務め，平成 14 年に ACM OOPSLA'2002，平成 15 年には IEEE CCGrid のプログラム委員長．また，Global Grid Forum の Steering Group 委員ならびに Area Director を務める．ソフトウェア科学会，ACM，IEEE-CS 各会員．

関口　智嗣（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 59 年筑波大学大学院理工学研究科修了．同年電子技術総合研究所入所．情報アーキテクチャ部主任研究官．以来，データ駆動型スーパーコンピュータ SIGMA-1 の開発等の研究に従事．平成 13 年独立行政法人産業技術総合研究所に改組．平成 14 年 1 月より同所グリッド研究センターセンター長．並列数値アルゴリズム，計算機性能評価技術，グリッドコンピューティングに興味を持つ．市村賞受賞．日本応用数理学会，ソフトウェア科学会，SIAM，IEEE 各会員．