2003－HPC－95　(14)
2003／8／4

Grid PSE Builder: グリッドにおける汎用PSE構築ツールの開発

平野基孝[†]　山本直孝[††]　田中良夫[††]
伊藤智[††]　関口智嗣[††]

我々は Grid 環境に対応した Web ベースの分散 PSE を構築するためのフレームワークの提供を目的として, Grid PSE Builder を開発している. Grid PSE Builder フレームワークにより, Grid PSE 開発者には Web セキュリティ, Web プログラミングに関する高度な知識を必要とせずとも, 既存の大規模科学技術計算アプリケーションやシミュレーションエンジン等を問題解決コンポーネントとする, Grid 環境と Web 環境を統合した Grid PSE の構築手法, PSE 利用者には Web インターフェースのみを介した高セキュリティかつ高利便性を持つ Grid PSE へのアクセス機構が, それぞれ提供される. 我々は実装途上の Grid PSE Builder の Grid ポータルサーバとしての機能を用いた実アプリケーションの Grid ポータル化を行い, 僅かな工数での既存アプリケーションの PSE コンポーネント化が可能であるという結果を得た.

Grid PSE Builder: Design and implementation of Grid-enabled problem solving environment builder

Motonori Hirano,[†] Naotaka Yamamoto,[††] Yoshio Tanaka,[††]
Satoshi Itoh[††] and Satoshi Sekiguchi [††]

We have designed and implemented Grid PSE Builder to provide a developing framework for Grid-enabled distributed PSE (Problem Solving Environment). For a PSE implementer, the framework provides a method to build a Grid PSE which consists of existing solving engines, such as large-scale scientific-engineering applications and simulation engines, as problem solving components without any particular knowledge about the Web securities and the Web programming. For a PSE user, it provides an access method to the Grid PSE with highly secured authentication mechanism and user-friendliness. By using Grid PSE Builder's function as a Grid portal kit, we have made some Grid portals with existing applications and it results that the Grid PSE Builder makes existing applications Grid PSE components with small amount of works.

1. はじめに

PSE (Problem Solving Environment) は, 計算機科学や計算機環境に関して特別な知識を持たない専門家 (科学者, 技術者) が, それらの知識を必要とせずとも, 彼らの専門分野の問題を解決するための総合的な環境として用いることが可能なコンピュータシステムであり, 例えば文献[1] 等に多数の例を見ることができる. これらの例は主に, Grid 環境上で動作する様々なアプリケーションを問題解決用コンポーネントとして分散 PSE を構築するための研究であり, Grid 環境を用いた分散 PSE は, Grid PSE と呼ばれ始めている.

Web 環境が発達し, 一つの確立したアプリケーションプラットフォームとして普及した現在, 分散 PSE のユーザインターフェースとして Web を用いることは, ユーザの利便性を鑑みて大変有意義である. Grid 環境に存在する既存のアプリケーションを Web インターフェースを介して使用可能にする Grid アプリケーション Web ポータル構築用フレームワーク (以下 Grid ポータルキット) の研究も行われており, GridPort[5], GPDK[4] (Grid Portal Development Kit) 等の実装が知られている. しかし, これらを使用して Grid PSE を構築しようとする場合, 実装方法によってはセキュリティ脆弱性を含んでしまうこと, PSE 開発者に Web セキュリティの知識を含む, 多くの Web プログラミング知識を要求するなど, 幾つかの問題点が考えられる.

我々はこれらの問題を考慮した上で, Grid PSE 構築のためのフレームワークの提供を目的とする Grid PSE Builder を開発している. Grid PSE Builder は, PSE 利用者には高セキュリティな Web インターフェースを介した PSE へのアクセスを提供し, PSE 提供者には Web プログラミングに関する高度な知識を必要とせずとも, 既存のアプリケーションを PSE コンポーネント, あるいは PSE そのものとして Web アクセス可能にする手段を提供する.

Grid PSE Builder の現段階までの実装は, 特に Web インターフェースと PSE コンポーネントの結合機構の実現に注力されている. したがって, 現状の Grid PSE Builder は, いわゆる Web 関連技術用語で呼称されるところの, Web アプリケーションポータルサーバの機能を有しているといえる. 本稿では主に, この Web - PSE コンポーネント結合手法について解説する.

本稿では, まず**第 2 節**で, 我々の目指す Grid PSE

† 株式会社 SRA
Software Research Associates, Inc.
†† 産業技術総合研究所 グリッド研究センタ
Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology

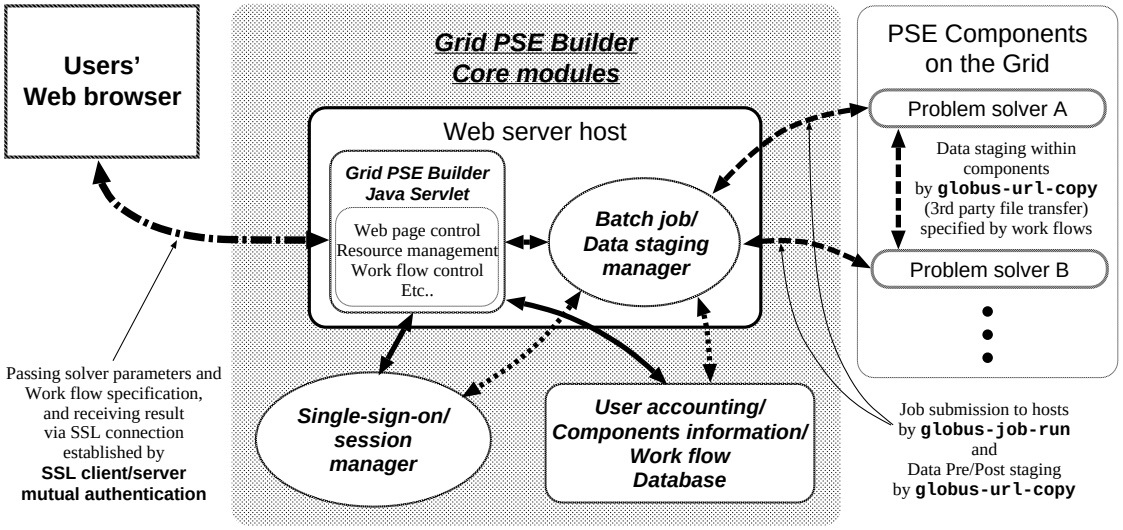

図 1: Grid PSE Builder を用いた Grid PSE の概要

構築フレームワークの要求仕様を明らかにした上で, Grid PSE 構築のために既存の Grid ポータルキットを使用する場合の問題点およびその解決手法について述べる. 次に**第 3 節** で Grid PSE Builder の概要と, Grid PSE Builder で構築された Grid PSE の動作概要を述べ, **第 4 節**で Grid PSE Builder の適用実例を紹介する. 次に**第 5 節**で関連研究について述べる. 最後に**第 6 節** でまとめと Grid PSE Builder の現状および今後の予定について述べる.

2. Grid PSE 構築フレームワークの要求仕様

我々が目標とする Gird PSE およびその構築フレームワークの要求仕様を以下に示す.

(1) セキュリティに十分配慮すること.
(2) PSE 使用者の操作に Web インターフェースを用いることでユーザの利便性を高く保つこと. これには, 基本的に PSE 操作の全てが Web ブラウザのみにて行えることが含まれる.
(3) セキュリティ管理, Web 画面プログラミング等を極力隠蔽し, PSE 開発者がそれらの高度な専門知識を必要とせずとも, 最小の労力で Grid PSE の構築が行えるためのメカニズムが提供されること.
(4) PSE コンポーネント間の入出力依存関係を記述するための, ワークフロー記述メカニズムが提供されること.

次に, 既存の Grid ポータルキットを用いて Grid PSE を構築する際の問題点を考察する. 特にセキュリティ機構の考察のため, まず現状の Grid 環境構築のためのデファクトスタンダードとなっている, Globus toolkit の使用を念頭におき, そのセキュリティ機構を説明する.

2.1 Globus toolkit のセキュリティ機構

Globus toolkit は PKI ベースのサーバ・クライアント間相互認証を行うことで高セキュリティ性を確保している. 通常の PKI 認証 (SSL/TSL) では, 秘密鍵と, それから生成される公開鍵とその他認証情報, いわゆるデジタル ID 証明書を用いるが, Globus toolkit ではさらに, 秘密鍵とデジタル ID 証明書から, proxy 証明書と呼ばれる有効期間の短い一時証明書 (とその元になる一時秘密鍵) を生成し, これをクライアント側のデジタル ID 証明書として使用する. 以下, proxy 証明書作成の元になったデジタル ID 証明書を primary 証明書と呼称する. PKI ではさらに, シングルサインオン機構をセキュアに提供する手段として, credential delegation の概念を用意している. Globus での credential delegation は, proxy 証明書を認証ケーパビリティの入れ物として, ケーパビリティを委譲したい相手 (peer プロセス, ホスト) との間で, 現在持っている proxy 証明書を primary 証明書として, さらに proxy 証明書を生成・委譲することで実現されている. この機構を Globus では GSI (Grid Security Infrastructure) と呼んでおり OpenSSL の上で実装されている. GSI の上位層に, セキュリティ API として IETF 標準となっている GSSAPI☆を実装し, Globus の通信層 API である Globus I/O が GSSAPI 層を介すことで, P2P でのセキュアな通信が可能となる.

2.2 Grid PSE 構築における既存 Grid ポータルキットの使用時の問題点

2.2.1 ユーザ認証機構の問題点

Web 側では, 前述の proxy 証明書および credential delegation の機構は直接サポートされていないので, proxy 証明書を用いた認証を行うためには, どこかで生成された proxy 証明書を, Grid 環境にセキュアに委譲する手段が必要となる. Globus を用いた Gird 環境への Web 環境を介した proxy 証明書の委譲手法として, MyProxy[3] が実装されている. MyProxy は, 任意の場所にあり, かつ Globus が完備されたマシン上に存在するユーザの primary 証明書を利用して, MyProxy の提供する `myproxy-init` コマンドにより作成された proxy 証明書を, ユーザ名と任意のパスフレーズをキーにして MyProxy サーバに登録

☆ この方式の Globus での GSSAPI 実装は, GSSAPI-GSI と呼ばれている.

しておき, これを基に proxy 証明書を実際に必要とする Grid/Web アプリケーションポータルサーバが MyProxy サーバよりそのユーザの proxy 証明書を得る, という方式で動作する. GridPort, GPDK ともに, MyProxy をユーザ認証機構に用いている. また GridPort は, primary 証明書とその秘密鍵を Grid/Web アプリケーションポータルサーバに置いておき、Web ブラウザ上からユーザ名と秘密鍵のパスフレーズを入力させることで proxy 証明書をサーバ上で作成するという認証方式も提供しているが、この手法は PKI の、秘密鍵がユーザの手元にしかないので安全であるという前提にそもそも違反しており、セキュアなユーザ認証方式とは到底呼べない。したがって、本研究ではこの手法を完全に無視する。

MyProxy を利用する場合, Grid/Web アプリケーションポータルサーバが任意のユーザ proxy 証明書を MyProxy サーバから取得するための, ユーザ名とパスフレーズの入力のための (ユーザ認証) 機構を PSE 開発者自身で実装する必要があり, ここで Web ベーシック認証等, 背後で動作する Globus の PKI ベースの認証に比べて攻撃に対し著しく脆弱な手法が使用される可能性があり, 攻撃者の格好の攻撃ポイントとなり得る. たとえ SSL 接続が使用されたとしても, 構築された Grid/Web アプリケーションポータルサーバ自身にクロスサイトスクリプティング脆弱性[7] があれば, ユーザ名, パスフレーズ, セッション情報等の漏洩が起こりうる.

さらに, PSE 使用者が, Globus および MyProxy の使用できる環境を用意しなければならず, かつ, Web ブラウザ外での操作をユーザに行わせる必要があるため, Web ブラウザさえあれば Grid PSE を使用できるというユーザ利便性が確保できない.

このように, MyProxy を利用する既存の Grid ポータルキットでは, 我々の要求仕様のうちの (1), (2) を満たすことができない. そこで我々はユーザ認証に Web SSL サーバ・クライアント相互認証機構を用いるとともに, セッション管理用デーモンプログラムを用意し, ブラウザ cookie を使用しないセッション管理手法を実現した. これにより, この手法以外の脆弱なメカニズムによるユーザ認証機構を PSE 開発者が用意しなくて済み, 要求仕様 (1) が満たされる. また, Web SSL サーバ・クライアント相互認証は現在使用されているほとんど全ての Web ブラウザ, Web サーバで使用できる認証方式であることから, Web インターフェースのみで実現でき, 要求仕様 (2) も同時に満たすことができる.

2.2.2 PSE 開発者の労力

GridPort, GPDK ともに, 既存のアプリケーションを Web から呼び出すための機構を十分提供しているとは言い難く, これらを使用して Web ベースの Grid PSE を構築する場合, Perl CGI, Java script などを用いたいわゆる Web プログラミングの知識が必要となり, さらに前節で挙げたセキュリティ脆弱性の解決のためには, かなり高度な Web セキュリティの知識が必要とされる. しかし PSE を使うユーザ側も, PSE を提供する側も, 特定分野の専門家であっても Web 環境, Grid 環境の専門家で無いことが多く, 既存の専門分野アプリケーションを Web アクセス可能な PSE コンポーネント化したり, そのコンポーネントを呼び出すためのセキュアな Web 環境を構築すること自体に多くの労力が割かれることが想像される. これも, 我々の要求仕様のうちの (3) を満たさない.

そこで我々は, PSE コンポーネントとなる既存のソルバ, アプリケーションと Web 画面の結合のための, XML ベースの簡易言語 (以下コンポーネント XML) を提供することで, Web 認証機構の実装, 煩雑なスクリプトベースの HTML 記述, サーバサイド Web/CGI プログラム等から PSE 開発者を解放することにした. PSE 開発者は, 基本的にソルバアプリケーションプログラムへのパラメタ群を, どのような形式で Web ページから入力させるかを指定するだけで, 当該プログラムを Web アクセス可能にできる.

2.2.3 ワークフロー記述

現状, Grid 環境におけるワークフロー記述のために一般化された手法が存在しない. 我々は要求仕様 (4) を満たすための手法として, 一般化された手法を用いるべきだと考えており, 候補としては, BPEL4WS 等があがっている. いずれにしろ, 言語としては XML を用いることになると思われ, 本研究ではこのワークフロー記述のための言語を仮にワークフロー XML と呼称している. また, PSE 利用者, PSE 開発者が, 直接ワークフロー XML を用いてワークフローの記述を行うのは煩雑であることが予想されるので, UML による記述からワークフロー XML を生成する機構を提供することも考えている.

3. Grid PSE Builder 概要

Gird PSE Builder を用いた Grid PSE の概要を**図 1** に示す. **図 1** 中, 網掛けで囲まれた部分が Grid PSE Builder が提供する機能を実現するためのモジュール群であり, Grid PSE Builder Core Modules と呼ばれる.

Gird PSE Builder は, シミュレーションエンジンやアプリケーション自身が独立した一つの CUI プログラムしてと記述でき, バッチ処理可能であるシステムの PSE コンポーネント化を対象としており, 例えば何らかのウィンドウシステムで動作する GUI 部分が完全にシミュレーションエンジンと不可分なシステムや, 出力がリアルタイムストリーム形式 (動画, 音声等) でしか表現できないシステム☆, インタラクティブな操作が中心となるシステム (例えば 3D CAD) の PSE コンポーネント化は現状では対象としていない.

以下, 現状の Grid PSE Builder Core Modules の動作概要を, 主に PSE 利用者から見たフェーズを追って説明する.

3.1 サインオンフェーズ

まず, ユーザからの Web アクセスは, 任意の Web サーバプログラムの提供する servlet コンテナ内で動作する Grid PSE Builder Java servlet (以下 Java servlet) で処理される. クライアント Web ブラウザには, あらかじめ任意の認証局から取得したクライアントデジタル ID 証明書がインポートされているものとする. 任意の認証局を利用できない場合のため, Grid PSE Builder には, 簡易認証局キットが含まれており, PSE 提供者が容易に認証局を立ち上げ, PSE 利用者用デジタル ID 証明書を発行するための手段を

☆ もし動画等を最終結果とするならば, 任意の動画フォーマットで結果ファイルを生成するバッチ処理プログラムにすればよい.

提供している☆.

Web サーバプログラムとクライアント側 Web ブラウザは, SSL サーバ・クライアント相互認証により, PKI ベースの認証を得たうえで SSL 接続を確立する. SSL サーバ・クライアント相互認証とは, 通常の Web SSL 接続では, クライアントのみがサーバ側から提示されたサーバデジタル ID 証明書によってサーバの身分を確認するのに対し, さらにクライアント側が自分の持つクライアントデジタル ID 証明書をサーバ側に提示することで, サーバ側がクライアントの身分を確認するという認証方式を指す.

Java servlet は, 前述の SSL 接続認証時にクライアント側から提示されたクライアントデジタル ID 証明書内の DN (Distinguished Name) を元に User accounting DB を検索し, そのユーザのパスフレーズの入力を促すための Web 画面をクライアントブラウザに返す. 次に入力されたパスフレーズの検査により, 正当な利用者であることを確認する. 利用者確認が成功した場合, 利用許可がそのユーザに与えられるとともに, Single-sign-on/session manager にセッション情報が記録され, セッションの制限時間管理が開始される. 前述したように, 通常の Web アプリケーションではセッション情報をブラウザ cookie としてクライアントブラウザ側に持たせることが多いが, Grid PSE Builder ではこれをサーバ側で管理することで, クライアント側の何らかのセキュリティ問題でブラウザ cookie 内のセッション情報が悪意ある第三者に漏出するする可能性を皆無にしている.

セッション制限時間はデフォルトでは 12 時間であるが, PSE 提供者側で, 任意の時間制限が可能である. 一度利用許可が与えられると, 制限時間内であれば, サインオンフェーズを経ることなく, 次のコンポーネント選択フェーズに移行する. 制限時間を越える前に, 任意の時点でユーザ自身が利用許可を解除すること (サインオフ) も可能である.

3.2 コンポーネント選択フェーズ

使用許可の与えられたユーザのブラウザには, Java servlet が PSE コンポーネント選択用の Web 画面を提示する. ユーザはこれより自分の行いたい処理コンポーネントを選択する.

3.3 パラメタ入力フェーズ

ユーザによりコンポーネントが選択されると, Java servlet は Component Information DB より選択されたコンポーネントに関する,

(1) コンポーネントへの入力パラメタ (コマンドライン引数) 情報
(2) 入力パラメタをどのように Web 画面上のフィールドとして表現するかを与える Web 画面構成情報

を記述したコンポーネント XML を得る. Java servlet はこの XML をパースし, Web 画面情報より適切な HTML を生成し, クライアントブラウザに提示する. ユーザはこの画面により, 必要な入力パラメタを入力, 選択することができる. 入力パラメタとしては, 任意の文字列, 数字のほか, クライアントブラウザが動作するコンピュータ上のローカルファイルのファイルを指定することも可能である. この場合, クライアントブラウザが自動的にファイル選択ダイアログを表示する.

3.4 ジョブサブミットフェーズ

必要なパラメタ入力が終了した後, ユーザが web 画面上のサブミットボタンをクリックすることで, 指定されたパラメタは, HTTP post メソッドにより Java servlet に転送される. ファイルを指定した場合, Web サーバマシン上のデータスプールに一時的にアップロードされる. 次に Java servlet は, コンポーネント XML 内の入力パラメタ情報を元に実行すべきコマンドラインを生成し, Component Information DB よりそのコンポーネントを提供する Grid 環境上のマシンを取得した後, そのマシン上で実際にそのコンポーネントを実行するために, Batch job/Data staging manager にジョブをサブミットする. サブミットされたジョブにはユニークな Job ID が振られる. 次に Java servlet は, その Job ID をユーザに通知するための Web 画面を提示し, ユーザにジョブサブミットの成功を通知する. 以降, ユーザは Grid PSE Builder で作成された Grid PSE のトップページから辿れるジョブステータス画面に任意の時点でアクセスし, サブミットしたジョブの実行状況を確認することができる. この実行状況は, User Accounting DB に更新・蓄積される.

3.5 コンポーネント実行フェーズ

Batch job/Data staging manager にサブミットされたジョブは, `globus-job-run` コマンドにより, 当該マシン上で実行が開始される. 各々のジョブは, `globus-job-run` でリモート実行されるプロセスのデフォルトのカレントワーキングディレクトリ直下に, そのジョブ専用の一時スプーリングディレクトリを作成し, そのディレクトリに移動してから実際のジョブコマンド (コンポーネントプログラム) を実行する. もしそのジョブのコンポーネントの入力パラメタにファイルが含まれる場合, 実行に先立って, `globus-url-copy` コマンドによってそのファイルが (前述の一時ディレクトリに) プレステージングされる.

実行が終了したら, その実行によって生成された標準出力, 標準エラー出力, その他実行前には存在しなかったファイル全てを Batch Job/Data staging manager のジョブ結果スプールディレクトリにポストステージングする. もしワークフロー記述がある場合, 次に実行されるコンポーネントの実行マシンに, そのコンポーネントの入力パラメタとして必要とされている出力ファイルがプレステージングされる☆☆. 最後に, User Accounting DB 内のジョブ実行状況を終了状態に更新する.

このフェーズでの `globus-*` コマンド実行に必要とされる proxy 証明書には, Batch job/Data staging manager のプロセス所有者のものが基本的に使用される. Batch Job/Data staging manager は, いわゆるデーモンプロセスとして動作しており, 必要な時に proxy 証明書を生成している☆☆☆. したがって, リモートホストでのコンポーネント実行には, 高々一つのユーザアカウントしか必要としない. これは, Grid 環境内の全ての PSE コンポーネントサーバにおける PSE

☆ この場合, 簡易認証局自身の root デジタル ID 証明書もクライアント Web ブラウザにインポートされていなければならない. そうでない場合, 通常 Web ブラウザがセキュリティ警告を表示する.

☆☆ 現在実装中.

☆☆☆ SSL Web サーバと同様に, プロセス起動時に秘密鍵のパスフレーズの入力を必要とする.

コンポーネントの提供者の管理負荷の低減と, セキュリティレベルの向上を目的としたものである.

3.6 ジョブ結果表示フェーズ

ジョブ実行が終了したジョブに関して, ユーザはジョブステータス画面から当該ジョブの Job ID を指定することで, ジョブ結果表示画面が Java servlet により提示される. ジョブ結果表示画面にはそのジョブの標準出力, 標準エラー出力がテキストで表示されるとともに, 生成された出力ファイルへのハイパーリンクが, 全出力ファイルに対して張られている. ユーザは必要に応じてそれらを選択し, 使用しているマシンへダウンロードすることができる.

4. 適 用 事 例

現状の Grid PSE Builder の機能を利用した web ベースアプリケーションポータルサーバの構築事例として, 首藤らの S-model による長期気象予報ポータル (以下気象予報グリッドポータル)[8] がある. また, 市販の熱流体解析エンジン PHOENICS を PSE コンポーネントとするポータルシステムが山本により開発中である.

既存のプログラムを Grid PSE Builder を用いた Grid PSE のための PSE コンポーネントにするためには, 基本的にコンポーネント XML を記述するだけでよい. **図 2** に, 気象予報グリッドポータルで使用しているコンポーネント XML を示す. `<argspec>` タグで指定された部分が, 実際に起動されるコマンドラインを与える. "`%`" ではさまれた識別子はコマンドライン引数の仮引数名となり, `<args>` タグ内で "`name=`" の形式で参照可能である. また `<args>` タグ内では, その引数の入力をどのように Web 画面から行わせるかの指定を行う. **図 2** でのコンポーネント XML 例はおおよそ 60 行であり, Grid PSE Builder をベースとした Web サーバ環境の構築等の初期設定の手間を勘定に入れたとしても, スクラッチで同等機能を有する Web アプリケーションを開発する場合に比べて十分小さい労力でのアプリケーションの PSE コンポーネント化が可能であると考える.

また西川らは, Grid PSE Builder のサインオンおよびユーザ認証機構のみを用い, GridPort による web アプリケーションの高セキュリティ化を行うという手法により, Gaussian Portal Phase2[9] を構築した. 我々はこの手法の実現のため, Grid PSE Builder のコンポーネント XML の`<argspec>` タグに URL を指定可能にする機能を追加した. ユーザ認証は, この URL に CGI 引数として Grid PSE Builder 用クライアントデジタル ID 証明書の DN 内のユーザ名が渡され, それを受け取った CGI がそのユーザのサインオン状況を Single-sign-on/session manager に SSL 通信☆で問い合わせる, という手法で実現されている.

5. 関連研究と関連技術

PSE を Grid 環境で構築するための研究としては, GrADS, TOOLSHED プロジェクトが挙げられる. 対応する Grid 環境として Globus toolkit を使用する点

☆ クライアント側がサーバの片側 SSL 認証を行い, サーバへのアクセスコントロールは Peer アドレスの deny ベース制限で行う.

```
<application
<appname>weather simulation</appname>
<appid>21</appid>
<appcomment>
Climate Simulation based on varotropic S-model. Set parameters.
[start date] and [end date] format is yyyy/mm/dd
</appcomment>
<argspec>
/work/smodelG/results/weather_tsukuba.pl %fromdate%/%fromh%
%todate%/%toh% %simulation%
</argspec>

<arglist>
<args use="required">
<title>start date</title>
<text name="fromdate" size="10"
 maxlength="10" value="2001/01/01">
  <constraint>
    <type value="string" />
  </constraint>
</text>
</args>

<args use="required">
<title>start time</title>
<select name="fromh" size="1" multiple="false">
<default value="0" />
<option value="0">0</option>
<option value="6">6</option>
<option value="12">12</option>
<option value="18">18</option>
</select>
</args>

<args use="required">
<title>end date</title>
<text name="todate" size="10" maxlength="10" value="2001/01/11">
  <constraint>
    <type value="string" />
  </constraint>
</text>
</args>

<args use="required">
<title>end time</title>
<select name="toh" size="1" multiple="false">
<default value="0" />
<option value="0">0</option>
<option value="6">6</option>
<option value="12">12</option>
<option value="18">18</option>
</select>
</args>

<args use="required">
<title>number of simulation</title>
<text name="simulation" size="3" maxlength="3" value="10">
  <constraint>
    <type value="integer" />
    <minminInclusive value="1" />
    <maxInclusive value="200" />
  </constraint>
</text>
</args>

</arglist>
</application>
```

図 2: コンポーネント XML 例

は両者と同様だが, 本研究ではさらにユーザインターフェースとしての Web 環境の必須性に着目している.

また, Grid 環境と Web 環境を統合するための手法研究としては, 冒頭で述べた GridPort, GPDK が挙げられるが, これらを用いる場合の問題点に関しては**第 2 節**で述べたとおりであり, 我々の目標とする Grid PSE 構築には適さない. MyProxy に関しては, 特に Windows プラットフォーム上での Globus toolkit の普及が進んだ上で, 更に Java Web start 等の技術と組み合わせることで, ユーザの利便性を高く保つような実装をすることが可能であると考えるが, そのような実装をフレームワークとして提供している例は見受けられない.

Web SSL サーバ・クライアント相互認証機構を用いた Grid ポータル構築環境として, ITBL[2] がある. ITBL は Grid 環境として Globus を用いていないため credential delegation の機構がなく, ユーザが起動したプロセス同士が安全に通信を行うには, STAMPI もしくは TME を用いるか, VPN を構成する必要がある. プロセス間通信を全て STAMPI, TME で行うことは性能的に不利となり, 様々なサイトが仮想的に一組織を構成する Grid 環境において VPN を構成することは, 現実的でないと考える.

次世代 Grid 技術の標準仕様である OGSA (Open Grid Service Architecture) の Web Service の枠組みを利用することで, Web ベースの Grid PSE を構築するための考察も Walker らによって始められているが[6], PSE 提供者に対して OGSA ならびに Web Service に適合したアプリケーションの容易な開発手法を与えるフレームワークの実装開発までには至っていない. 更に, OGSA ならびに Web Service での Grid PSE 構築が可能になったとしても,

(1) Web Service は SOAP で実装され, SOAP RPC ベースのパラメタ授受には XML フォーマットが用いられるので, 多大なオーバヘッドを生じることが白砂[10] らによって報告されている.
(2) 既存のシミュレーションエンジン等を使用したい場合, それらを SOAP から呼び出すために何らかの wrapping 機構が必要となる.
(3) そもそも, PSE 提供者に彼らの専門分野外の OGSA, Web Service 等の最新の Web, Grid 知識を要求している.

の問題点が挙げられる. それに対し本研究では,

(1) シミュレーションパラメタのフォーマットはソルバ, シミュレーションエンジン固有でよく, 通信オーバヘッドしか生じない.
(2) 既存のエンジン等を用いるための特別な wrapping 機構を必要とせず, 必要であれば, 既存の shell, perl などのスクリプト言語を用いればよい.
(3) 既存の Web 技術と Grid 技術のみを用いており, PSE 提供者に要求する知識は OGSA, Web Service を用いる場合より少ない.

という点で, それぞれ有利であると考える.

6. ま と め

我々は Grid PSE Builder を開発し, 全機能の実装前ではあるが, 実環境への適用を行い, わずかな工数での既存アプリケーションの PSE コンポーネント化が可能であるという結果を得るとともに, 次に示す知見も得た.

(1) ワークフローの自由度を向上するための, Grid 環境内に存在するデータストレージ, Grid PSE Builder Web サーバ, ユーザクライアントマシン, PSE コンポーネントサーバ間のシームレスなデータステージング機構の必要性
(2) PSE コンポーネントサーバ側での audit trail および課金管理のための, PSE コンポーネントの個別ユーザアカウントでの実行機構の必要性
(3) ジョブの経過状態のモニタリング機構の必要性

これらに関し, (1) は現設計の機能拡張, (2) は proxy 証明書のクライアントブラウザ上での生成および委譲機構の機能追加として, 現在設計開発が続いている. 特に (2) に関しては, Web SSL クライアント・サーバ相互認証に用いるクライアントデジタル ID 証明書から Globus 用 proxy 証明書を作成するという設計をおこなっており, Web 認証と Grid 認証および credential delegation 機構のシームレスな結合が可能になる予定である.

今後は, 未実装部分の実装完了, ワークフロー XML に関する設計実装等を行っていく予定である.

謝　　辞

本研究の一部は科学技術振興事業団計算科学技術活用型特定研究開発推進事業の一環として実施している「仮想スーパーコンピュータセンタ利用環境 GridLib の構築」によるものである.

最後に, Grid PSE Builder の設計開発関係者各位に謝意を表します.

参 考 文 献

1) CONCURRENCY AND COUMPUTATION PRACTICE & EXPERIENCE GRID COMPUTING ENVIRONMENTS SPECIAL ISSUE: Vol. 14, No. 13-15 (2002)
2) http://www.itbl.jp
3) Novotny, J. et al.: An Online Credential Repository for the Grid: MyProxy, *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press (2001).
4) Novotony, J.: The Grid Portal Development Kit, *CONCURRENCY AND COMPUTATION PRACTICE & EXPERIENCE*, Vol. 14, No. 13-15, pp. 1129–1144 (2002).
5) Thomas, M. et al.: The GridPort Toolkit Architecture for Building Grid Portals, *Proceedings of the 10th IEEE Intl. Symp. on High Perf. Dist. Comp* (2001).
6) Walker, D. W.: Web Services For Grid-Enabled Problem-Solving Environments. a presentation on the Access Grid on 15 March 2002.
7) http://www.ipa.go.jp/security/ciadr/20011023css.html.
8) 首藤一幸, 武宮博, 平野基孝, 田中良夫, 関口智嗣: 気象予報グリッドポータルの開発, 情報処理学会研究報告, 2003-HPC-93(-29), pp. 167–172 (2003).
9) 西川武志, 長嶋雲兵, 関口智嗣: Quantum Chemistry Grid/Gaussian Portal Phase2, 情報処理学会研究報告, 2002-HPC-92(-8), pp. 43–48 (2002).
10) 白砂哲, 中田秀基, 松岡聡, 関口智嗣: Web サービス技術を基盤とする GridRPC システムの評価, 情報処理学会研究報告 2002-HPC-91, 情報処理学会, pp. 197–202 (2002).