社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－HPC－96 (13)
2003／10／16

GridRPCシステムにおけるリモートプログラムシッピング機構

渡邊 啓正† 本多 弘樹† 弓場 敏嗣†
田中 良夫†† 佐藤 三久†††

グリッドにおける計算ミドルウェアとして、グリッドに対応したRPCシステムであるGridRPCシステムが注目されている。GridRPCでは、ユーザが呼び出したい関数がサーバ上にインストールされていない場合、ユーザがサーバ上へソースファイルを転送し、サーバ上でコンパイルして、それをGridRPCシステムの利用手順に従って登録（リモートプログラムシッピング）しなければならない。我々はこの作業におけるユーザの負担を軽減する機構を開発した。また、本機構は、非均質なグリッド計算資源に対してGlobus Toolkitを用いて接続し、自動的に計算資源を無駄無く使用してリモートプログラムシッピングできるという特徴をもつ。

Remote Program Shipping System for GridRPC Systems

Hiromasa Watanabe†, Hiroki Honda†, Toshitsugu Yuba†,
Yoshio Tanaka††, and Mitsuhisa Sato†††

As a Grid middleware for scientific computing, GridRPC systems are watched with keen interest. However, before starting computation with GridRPC, the user must perform operations in installation of the computational libraries used on the GridRPC servers, which may be complex and take a big deal of labor. We have developed Remote Program Shipping System, which reduces the user's trouble at the installation. Additionally, this system uses heterogeneous computational resources effectively that can be accessed by the Globus Toolkit. This paper describes the design and implementation of the Remote Program Shipping System.

1. はじめに

近年、広域ネットワーク環境において、地理的に分散した多数の計算資源を容易に効率良く利用することを目的としたグリッドが注目され、グリッドに関する研究が世界中で盛んに行われている。現在では、ユーザ認証・資源管理機構・通信ライブラリといったグリッドの要素技術を提供するグリッド構築基盤ミドルウェアとして、Globus Toolkit[1]が事実上標準で多くのグリッドの大学研究機関において利用されている。

また、グリッド上での高性能並列分散計算プログラムを容易に開発するためのミドルウェアとして、従来のRPCを容易にグリッド上で実現できるGridRPC[3]がある。GridRPCでは、手元の計算機から遠隔地のサーバ上のライブラリを呼び出すことにより並列分散計算を行う。最適化問題等のタスク並列プログラムの実装に対して有効なプログラミングモデルとして注目されており、Global Grid Forum[2]においても、GridRPC APIの標準化に関して議論が重ねられている。

GridRPCシステムとしては、米テネシー大学のNetSolve[8]、産業技術総合研究所のNinf[4]と、NinfをGlobus Toolkit上で動作できるようにしたNinf-G[4,5]が開発されている。近年では、クラスタ計算機の普及に伴い、複数のサイトに分散配置されたクラスタを利用して、マスターワーカー型の大規模な並列分散計算を実行するといった用途に使われている。

しかしながら、GridRPCを用いたプログラムを実行する場合、予めサーバ上にライブラリがインストールされていることが前提となっており、GridRPCシステムのユーザは、ライブラリのインストールのために、利用する全てのサーバについて、次の作業を行う必要がある。すなわち、サーバにログインして、ライブラリのソースファイルをコンパイルし、GridRPCシステムが提供するスタブルーチンをリンクして、作成された実行形式をGridRPCサーバに登録するという作業が必要である（以降、この一連の作業をリモートプログラムシッピングと呼ぶ）。

数多くのサーバを使用する状況や、デバッグ等の理由でライブラリを頻繁に更新する状況では、ユーザが全てのサーバに逐一ログインしてライブラリのインストール作業を行うのは、ユーザにとって大きな負担となる。また、異なる組織間で計算資源を共有するグリッドにおいては、物理的なログインができない、あるいはグリッドにおける標準の認証技術を用いたファイル転送コマンドや遠隔コマンド実行コマンドだけが利用できる、という場合がある。そのような環境では、ライブラリの遠隔インストール作業のために、ユーザは非常に複雑な手順を踏まなければならなくなる。

† 電気通信大学 大学院情報システム学研究科
The Graduate School of Information Systems, University of Electro-Communications
†† 産業技術総合研究所 グリッド研究センター
Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology
††† 筑波大学 電子・情報学系 計算物理学研究センター
Institute of Information Science and Electronics / Center for Computational Physics, University of Tsukuba

そこで我々は、この問題を解決するため、ユーザがサーバにログインせずに、サーバ上にライブラリを容易にインストールできる機構を設計・実装した。本機構を用いることで、ユーザは、リモートプログラムシッピングを行う環境に関する情報を記述したファイルを作成するだけで、複数のサーバ上にライブラリを自動的にインストールすることができる。

さらに、本機構ではグリッドにおけるリモートプログラムシッピングに関する他の課題も解決している。すなわち、冗長なコンパイルの回避やライブラリのバージョン管理の導入を考慮して、ライブラリをコンパイルするホストを限定する機能がある。また、グリッドが持つ資源の非均質性や、サーバ毎に異なる設定に対応するべく、configure を自動生成する機能や、ライブラリのインストールに関する資源の運用環境情報を反映する機能を備えている。

本稿では、第２章で GridRPC システムである Ninf-G におけるリモートプログラムシッピング作業の詳細について述べ、第３章で本機構の設計及び実装について述べる。第４章で本機構の全体動作をまとめ、第５章で本機構の動作試験について述べる。第６章で関連研究、第７章で今後の展開について言及し、第８章でまとめる。

2. GridRPC システム：Ninf-G

Ninf-G は、旧実装である Ninf をもとに、通信層や計算資源管理部分等を Globus Toolkit を利用して再設計実装した GridRPC システムである。本機構開発時点で Globus Toolkit 上で動作する唯一の GridRPC システムであり、ユーザ認証に対応した唯一の GridRPC システムである。また、本機構との連携使用を見込む OpenGR コンパイラ [6]（第６章で後述）が Ninf-G を実装対象としているという理由もあり、本機構は Ninf-G を実装対象の GridRPC システムとした。他の GridRPC システムへの対応については第７章で触れる。

次に、Ninf-G において、被呼び出し関数を計算サーバ上で GridRPC を用いて呼び出せるライブラリとするまでに、ユーザが行わなければならない作業の手順を示す（図１を参照）。

① ユーザが、サーバに実行させたい関数を含むソースファイル群（ライブラリの引数情報等を記述した Interface Description Language(IDL)ファイルや、Makefile も含む）を作成する。
② ソースファイル群をサーバ上へ転送する。
③ サーバ上で被呼び出し関数のソースファイルをコンパイルする。
④ サーバ上で Ninf-G によって提供される IDL コンパイラを用いて IDL ファイルをコンパイルし、スタブモジュールを作成する。
⑤ サーバ上で make を実行して、計算ルーチンとスタブをリンクした実行形式（ライブラリ）を作成する。
⑥ サーバ上で make install を実行することにより、作成されたライブラリを計算サーバ上の GridRPC ライブラリデータベース（Globus Toolkit の MDS）に登録する。

3. 機構の設計及び実装

グリッドにおけるリモートプログラムシッピングの実装に必要な機能は次の通りである。

リモートプログラムシッピング機能
グリッドでの利用に適したツールを用いて、ユーザがサーバにログインせずに、ライブラリを自動的にインストールできるようにする。

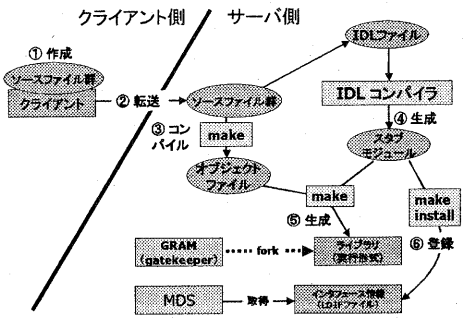

図１：Ninf-G におけるリモートプログラムシッピング作業

configure 自動作成機能
非均質なサーバ上で、サーバのシステムに応じたライブラリのコンパイルを容易に実現するために、ライブラリのソースファイルに対する configure スクリプトを自動的に作成する。

コンパイルサーバ機能
均質なサーバ群にリモートプログラムシッピングする際、冗長なコンパイルを避けるため、ライブラリをコンパイルするホストを限定する。

資源の運用環境情報の反映機能
ライブラリのインストールに関するサーバ毎に異なる運用設定を遵守するために、サーバの運用環境情報を自動的に反映してリモートプログラムシッピングを行う。

上記の各機能の本機構における設計及び実装について、以下に述べる。

3.1 リモートプログラムシッピング機能

3.1.1 シッピングに使用するツール

リモートプログラムシッピングを行うには、クライアント側からサーバ側へライブラリのソースファイルを転送する機能と、サーバ上でライブラリのコンパイル等のインストール作業を行うために遠隔にコマンド実行を行う機能が最低限必要である。また、グリッドの計算機上での動作を可能とするには、シッピングに使用するツールは次の要件を満たさなければならない。

① グリッドの多くの計算機上でツールが利用可能であること
② 許可されないユーザからのライブラリインストールを拒否するために、ユーザ認証機能を有すること
③ ユーザ認証を内部的に頻繁に行うため、シングルサインオン（パスワードを一度入力するだけで、複数の計算資源へアクセスできる機能）が可能であること

これらの要件を満たすツールとして、本機構では Globus Toolkit を選択した。これは次の理由に基づく。Globus Toolkit には、高性能ファイル転送機構である GridFTP と、遠隔コマンド実行が可能な資源管理機構である GRAM（Grid Resource Allocation Manager）がある。また、本機構が Ninf-G を実装対象の GridRPC システムとしているため、Ninf-G の基盤システムとして Globus Toolkit が利用できることを

想定できる（①）。さらに、Globus Toolkit には公開鍵暗号、X.509 証明書等の技術に基づくセキュリティインフラ（Grid Security Infrastructure：GSI）が組み込まれている（②）。加えて、GSI はシングルサインオンが可能なように設計・実装されている（③）。

3.1.2 機構全体の実装

本機構の全体的な実装に用いる環境を決める上で、次の点を重視した。

（1）グリッドの非均質な計算機上で動作させるために、高い可搬性を有すること
（2）複数のサーバへ並行してリモートプログラムシッピングを行うために、並列処理が容易に実装可能であること
（3）本機構の開発における手間が少なく、バグの混入の危険性が少ないこと

C 言語や Java 言語による実装の場合、Globus Toolkit が提供する API を用いて開発を行うことになるが、(1)が何らかの方法で解決できたとしても、(2)や(3)の点で開発コストが高くなり不適切である。

そこで、本機構は（Bourne）シェルスクリプトによって実装することにした。なぜなら、シェルスクリプトは上記の(1)や(2)の点を容易に満足するからである。(3)については、Globus Toolkit が標準で提供する GridFTP や GRAM のコマンドインターフェース（globus-url-copy、globus-job-run コマンド）を用いることで、開発コストを低く抑えられる。

シェルスクリプトに類似したスクリプト言語である Perl も候補として考えられるが、本機構との連携使用を見込む OpenGR コンパイラが Perl を前提に開発されていないため、本機構で Perl を開発環境として用いることは避けた。

3.1.3 シッピング設定ファイル

リモートプログラムシッピングを行う環境に関する情報であるシッピング設定情報は、シッピング環境が変化したときに設定項目の再編集が容易であるという理由から、テキストファイルとして作成しておき、リモートプログラムシッピングスクリプトを呼び出す際の引数として指定することにした。そのため、ユーザはこのシッピング設定ファイルをシッピング環境に応じて作成する必要がある。

シッピング設定ファイルには、XML 形式を採用している。これは、シッピング設定ファイルの可読性を向上させるためには XML で利用できる木構造が便利であり、また、入手しやすく高速な XML パーサライブラリ[10,11,12]が確立されている、という理由に基づく。シッピング設定ファイルの例を図２に示す。

また、本機構のシッピングスクリプトは XML 形式のシッピング設定ファイルを直接理解できない。そのため、XML 形式のシッピング設定ファイルからシッピング設定情報を抽出してシェルスクリプト形式で書き出すプログラムを、expat[10]ライブラリを用いて C 言語で開発した。このプログラムは、クライアントホスト上でシッピングスクリプト実行時に内部的に呼び出される。

さらに、シッピング設定ファイルでは、主にクラスタのノード群に対するシッピング環境情報の記述を容易にすることを目的として、次の機能を実装している。すなわち、ホスト名が連番になっているサーバ群にシッピングする場合には、node0[0-7].abc.com といった正規表現でホスト名を指定でき、それらに対して一括して同じ設定項目を記述することができる機能

```
<shipping-environment>
  <debug>no</debug>
 (中略)
</shipping-environment>
<client>
  <hostname>localhost</hostname>
  <username>watanabe</username>
  <module>
    <name>mmul</name>
    <source-dir>/home/watanabe/rps/mmul/server
</source-dir>
    <build-option>
      <create-configure>yes</create-configure>
      <update-configure>no</update-configure>
    </build-option>
  </module>
</client>
<module-repositories>
  <module-repository>
    <hostname>gex1.yuba.is.uec.ac.jp</hostname>
    <access-type>globus</access-type>
 (中略)
  </module-repository>
</module-repositories>
<servers>
<server-group>
    <hostname>koume.hpcc.jp,koume0[0-3].hpcc.jp
</hostname>
    <access-type>globus</access-type>
 (中略)
    <mr-hostname>gex1.yuba.is.uec.ac.jp</mr-hos
tname>
 (中略)
  </server-group>
</servers>
```

図２　シッピング設定ファイル

である（一括設定項目の展開は前述の XML 解析プログラムが行う）。

また、シッピングスクリプトは、複数のシッピング設定ファイルを受け付けることができるように設計・実装した。これは、同じサーバ群に対して複数の異なるライブラリをインストールしたい場合等に、共通する設定項目（この場合はサーバ群にアクセスするための情報）の記述量を最小限に抑えるためである。

3.2 configure 自動作成機能

グリッドでは、一般にシッピング先の計算サーバ群が非均質で、計算サーバに応じた個別のライブラリコンパイル方法が必要になる可能性がある（特定のシステムにおいて特定のコンパイルオプションが必要になる等）。この問題に対し、インストールするソフトウェアの Makefile やヘッダファイルをコンパイル環境に応じて自動的に変更する configure スクリプトを使う手法が、従来から用いられてきた。

本機構でもこの手法に従い、ライブラリのソースファイル毎に configure スクリプトを作成することで、この問題を解決する。しかしながら、ユーザがライブラリのソースファイル毎に configure スクリプトを作成するのは、一般に困難で多大な負担となる。そこで、本機構では、このユーザの負担を軽減するために、configure スクリプトを自動的に作成する機能を提供する。

ただし、configure スクリプト中のシステム情報調査処理には既に確立された手法が多数存在するため、本機構がそれらを再開発することは不適切である。そのため、本機構では、特定の入力ファイルから configure スクリプトを自動生成する GNU autoconf[13]を用いて configure スクリプトを作成している。従って、本機構にはライブラリのソースファイルから autoconf の入力ファイルを自動作成するこ

とが要求される。これについてはソースファイル中でインクルードしているライブラリヘッダ名から経験的に推測することで実現している。

まとめると、本機構では、configure スクリプトを次の手順によって自動的に作成している（図３を参照）。

1. gcc を用いて、ライブラリのソースファイルがインクルードしているライブラリヘッダ名の列を取得する。
2. ライブラリヘッダ名から、必要なシステム情報調査処理を推測し、それに基づいて autoconf の入力ファイルを出力する。
3. autoconf を実行し、configure スクリプトを作成する。

3.3 コンパイルサーバ機能

近年では、クラスタ計算機の普及に伴い、複数のサイトに分散配置されたクラスタを利用して大規模な並列分散計算を実行するという用途で GridRPC システムが使われることが多い。そのように計算で使用するサーバ群が部分的に均質な状況では、ソースファイルをコンパイルして生成されたライブラリファイルを、他のサーバ上に複製して利用できることが多い。このとき、全てのサーバ上でライブラリのコンパイルを行うのは、資源の無駄遣いである。

そこで本機構では、そのような状況において、ライブラリのコンパイルを行うホスト（コンパイルサーバと呼ぶ）を限定し、コンパイルサーバ上で作成されたライブラリを他のサーバ（計算サーバと呼ぶ）へ複製・転送して、登録するということを自動的に行う（図４を参照）。なお、計算サーバに対するコンパイルサーバの指定は、計算サーバ管理者が計算サーバの運用環境情報データベース（Globus の MDS）に記述しておく形となっている（3.4 節を参照）。

また、作成されたライブラリをコンパイルサーバ上から他の計算サーバ上へ転送するために、広域ネットワークを介して接続可能な、ライブラリを保管するためのホスト（モジュールレポジトリ）が必要になる。モジュールレポジトリでは、複数のリモートプログラムシッピング作業が同時に行われる可能性があるため、次の条件で各ライブラリをモジュールレポジトリ内で区別して保存・管理している。

- シッピングユーザ名
- ライブラリ名
- ライブラリバージョン番号
- シッピング日時
- クライアントホスト名
- コンパイルサーバホスト名

3.4 資源の運用環境情報の反映機能

ユーザが計算に使用するグリッド上の計算資源は、多くの場合、ユーザ以外の管理者によって管理・運用される。本機構では、ユーザがそのような他人の計算資源を用いる際、計算資源の管理者の定めた運用環境情報に従って、リモートプログラムシッピングを行うようにするべきであると考える。

そこで、本機構では、各資源のリモートプログラムシッピングに対する運用環境情報を、各資源上の Globus Toolkit の情報提供サービスである MDS（LDAP データベース）に登録しておく。そして、リモートプログラムシッピングスクリプトが実行時にユーザの権限で MDS からそれらを読み出して、その運用環境情報に従ってリモートプログラムシッピングを行う。現時点では、本機構は次の運用環境情報を反映できる。

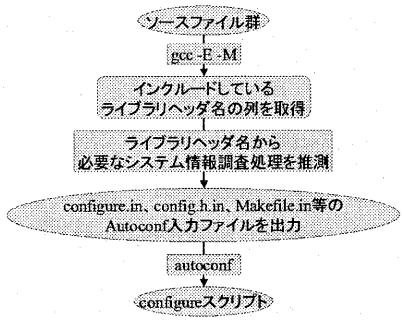

図３　configure 自動作成機能

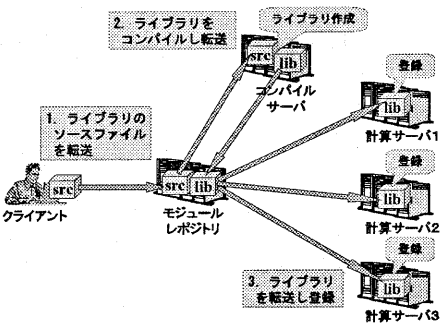

図４　コンパイルサーバ機能

- 計算サーバ上でライブラリをインストールするフォルダ
- 計算サーバに対するコンパイルサーバへ接続するための情報

計算資源の管理者が MDS にリモートプログラムシッピングに関する運用環境情報を追加するためには、まず、運用環境情報を LDAP Data Interchange Format（LDIF）形式ファイルで記述する必要がある。次に、作成した LDIF ファイルを、MDS の情報ツリーへ登録することになるが、それには煩雑なコマンド入力作業が必要となる（作業の詳細については省略する）。そこで本機構では、この作業を計算資源の管理者が容易に行えるようにするため、計算資源上で LDIF ファイルを MDS の情報ツリーへ自動的に登録するシェルスクリプトを提供している。

4. 機構の全体動作

第３章で述べた本機構の機能が全体としてどのように動作するのかを示す。

まず、事前に、各計算資源の管理者が各計算資源の運用環境情報を各計算資源の MDS に登録しておく必要がある。この手順は 3.4 節で述べた通りである。

そしてユーザが行う作業は次の通りである。

- ユーザが、計算サーバ上で実行したい関数をコンパイルするためのソースファイルや Makefile を作成する。
- リモートプログラムシッピングスクリプトに渡す

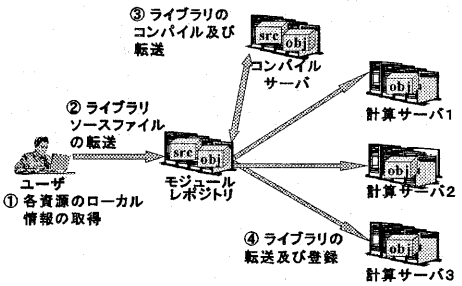

図5 本機構の全体動作

シッピング設定ファイルを作成する。

- Globus Toolkit の grid-proxy-init コマンドを用いてユーザのプロキシ証明書を作成し、クライアントホスト上で Globus Toolkit のコマンドを利用できるようにしておく。
- リモートプログラムシッピングスクリプトを起動する。

シッピングスクリプト内部で実行される処理は次の通りである。

- クライアントホスト上で、シッピング情報ファイル及び各資源の MDS から各資源の運用環境情報を取得する。
- クライアントホスト上から、ライブラリのソースファイルや Makefile をモジュールレポジトリ上に転送する。
- コンパイルサーバに対して、ライブラリのソースファイルをコンパイルし、作成されたライブラリファイルをモジュールレポジトリ上に転送するように、遠隔指示する。
- 各計算サーバに対して、モジュールレポジトリからライブラリファイルを各計算サーバ上に転送し、ライブラリファイルを登録するように遠隔指示する。

5. 動作試験及び評価

使用する計算サーバ群が不均質なグリッドを再現するため、開発したリモートプログラムシッピングスクリプトの動作試験を、表1の計算サーバ群を用いて下記の環境で行った。

- サイト1の1ホスト上で1回シッピングスクリプトを起動した。
- サイト1内にモジュールレポジトリホストを1つ指定した。
- ライブラリのソースファイルにおいて、コンパイル方法に機種依存性がある socket ライブラリをインクルードし、configure 自動作成機能を使用した。
- コンパイルサーバ機能を計算サーバ上のオブジェクトファイルに互換性がある場合において使用した。
- サイト1の計算サーバ4台と、サイト2の計算サーバ4台については、計算機のホスト名が連番であったため、シッピング設定ファイルでは正規表現を用いてホスト名を簡略表記し、一括して設定を行った。

サイト1 (電気通信大学) Globus Toolkit 2.0,Ninf-G 1.0, RedHat Linux 7.3 ×5台 Globus Toolkit 2.0,Ninf-G 1.0, Solaris 8 ×2台
サイト2 (産業技術総合研究所) Globus Toolkit 2.2,Ninf-G 1.0, RedHat Linux 7.3 ×5台

表1 動作試験に用いた計算サーバ群

動作試験の結果、上記のシッピング設定の通りに、全ての計算サーバに対してリモートプログラムシッピングが行われたことを確認した。

また、本機構では、ユーザはシッピング設定ファイルを作成しなければならない。この作業に要するユーザの手間について考察する。

ユーザはインストールしたいライブラリ毎にシッピング設定ファイルを基本的に1つだけ書く。ユーザが入手しなければならない情報やそのために必要な権限については、基本的に従来と変わらないが、シッピングに用いるモジュールレポジトリにアクセスするための情報や権限のみ、新たに必要となることがある（必須ではなく、無くてもシッピングできる）。シッピング設定ファイルの行数は最小で 60 行程度であり、シッピング環境が異なるサーバやモジュールレポジトリが増えた場合のみ、1サイトあたり約 12 行ずつ増加する(この理由は 3.1.3 節を参照)。

グリッドにおいて、使用するサーバの台数が多い、あるいは頻繁にライブラリを更新するという状況が予想される。その状況において、ユーザがサーバにログインしてライブラリインストールのためのコマンド列を入力する従来の方法では、ユーザはサーバ毎にログインして同じようなコマンドを何度も入力しなければならない。それと比較して、本機構では、ユーザは、サーバ毎に異なるシッピング環境情報のみをテキストファイルに記述しておき（最小で 60 行程度）、サーバにログインせずにシェルスクリプトを一度実行するだけで、全てのライブラリインストール作業を行うことができる。このことから、本機構がライブラリインストールにおけるユーザのコマンド入力作業の負担を大きく軽減することができると言えるだろう。

6. 関連研究

6.1 OpenGR コンパイラ

OpenGR コンパイラは、既存のプログラムにコンパイラ指示文（プラグマ）を挿入することによって、GridRPC を使用した並列分散プログラムを容易に開発できるようにする OpenGR 指示文機構とその処理系である。

OpenGR コンパイラはライブラリのソースファイルの作成を容易にし、本機構はそのファイルのサーバへのインストール作業を容易にするという、相補的な役割を担っている。双方を連携使用し、GridRPC を用いたプログラムの開発作業全体を容易に行えるようにする予定である。

6.2 既存のプログラム自動インストーラ

既存のクラスタ向けプログラム自動インストーラと異なり、本機構は、グリッドにおけるリモートプログラムシッピングで解決すべき問題を2つ解決している。1つはサーバが不均質であるために、実装に用いる基盤ミドルウェアがグリッドに適していなければならず、また configure が必要となるという点である。もう1つの問題は、サーバが他人のものであるため、サーバ

管理者のローカル管理方針を遵守しなければならない点である。また、共通点としては、均質なクラスタに対して本機構のコンパイルサーバ機能が有効であることが挙げられる。

グリッドの計算サーバ群にプログラムをインストールする機能を持つグリッドポータル構築ツールキットとして、白砂らによる PCT4G[7]がある。PCT4G と本機構との相違点を以下にまとめる。

- PCT4G ではシッピング先の計算機群が他人の管理下にあることを想定していない。そのため、本機構で実装している、遠隔計算機群の運用環境情報を反映してライブラリのインストールを行う機能が欠けている。
- PCT4G の、データの定期的な配布・更新機能は本機構には存在しない。これは、ライブラリインストールにおけるユーザの負担の軽減を第一の目的とする本機構のコンセプトとして含まれなかったからである。
- シッピング先の計算機数が増大した場合におけるシッピング設定ファイル中のノード群に対する設定記述は、本機構では一括指定を用いて容易に行うことができる。

7. 今後の展開

7.1 SSH、SCP を基盤とした動作

Globus Toolkit の各種コマンドが使えない状況を想定し、遠隔コマンド実行やファイル転送に、SSH や SCP を用いることができるように、現在拡張を行っている。新たに解決が必要な問題として、本機構の資源の運用環境情報を反映する機能で使用している Globus Toolkit の MDS に該当する情報提供サービスが存在しないため、MDS に代わる情報提供手段が必要となることが挙げられる。この代替の情報提供手段について、現在調査・検討を行っている段階である。

7.2 他のグリッド計算ミドルウェアへの対応

本機構は Ninf-G を実装対象としているが、他のグリッド計算ミドルウェア（Ninf や MPICH-G2[9] 等）を含めた、グリッドにおける汎用的な遠隔プログラムインストーラとして拡張していく予定である。そのためには、グリッド計算ミドルウェア毎に異なると推測されるプログラムのインストール手順（インストールにおけるワークフロー）の記述仕様を統一し、共通した方法で扱えるように本機構を再設計しなければならない。現在、ワークフローの統一的な記述方法について、検討を行っている段階である。

また、今後のリモートプログラムシッピング機構の改善として、次を予定している。

- 計算サーバ上のライブラリのバージョンに応じたリモートプログラムシッピング
- リモートプログラムシッピングの自動実行
- ユーザがクライアントホストから直接コンパイルサーバへ接続できない状況におけるコンパイルサーバ機能
- グリッドポータルへの適用

8. まとめ

GridRPC システムにおいて、従来ユーザがサーバにログインして行っていたライブラリインストール作業を、ユーザがサーバにログインせずに容易に行える機構を設計・実装した。本機構を用いることで、ユーザは、ライブラリをインストールする環境に関する情報を記述したファイルを作成し、本機構のシェルスクリプトを一度呼び出すだけで、複数のサーバに対しライブラリを自動的にインストールすることができる。

さらに、本機構ではグリッドにおけるリモートプログラムシッピングに関する他のいくつかの課題も解決している。すなわち、冗長なコンパイルの回避やライブラリのバージョン管理の導入を考慮して、ライブラリをコンパイルするホストを限定する機能を有する。また、グリッドが持つ資源の非均質性や、サーバ毎に異なる設定に対応するべく、configure を自動生成する機能や、ライブラリのインストールに関する資源の運用環境情報を反映する機能を備えている。

本機構は、GridRPC を始めとするグリッドミドルウェアを用いたプログラムの実行に際し高い利便性を提供するものであり、次世代の高性能計算基盤として注目されているグリッドを、より容易に、かつ効率良く利用することを可能とする。

謝辞 本研究を進めるにあたり、数多くの御助言、御協力を頂いた、産業技術総合研究所 グリッド研究センター長の関口智嗣氏、（株）SRA の平野基孝氏に深く感謝致します。

参考文献

1) The Globus Project, http://www.globus.org/.
2) Global Grid Forum, http://www.globalgridforum.org/.
3) K.Seimour, H.Nakada, S.Matsuoka, Jack Dongarra, C.Lee and H.Casanova, "Overview of GridRPC: A Remote Procedure Call API for Grid Computing," Proceedings of Grid Computing - Grid 2002, pp.274-278, 2002.
4) The Ninf Project, http://ninf.apgrid.org/.
5) Y.Tanaka, H.Nakada, S.Sekiguchi, T. Suzumura, and S.Matsuoka, "Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing," Journal of Grid Computing, Vol.1, No.1, pp.41-51, 2003.
6) 平野基孝, 佐藤三久, 田中良夫, 関口智嗣, "OpenGR コンパイラの設計および開発," 情報処理学会研究報告 ハイパフォーマンスコンピューティング研究会, Vol.2002, No.90, pp.55-60, 2002.
7) 白砂 哲, 鈴村 豊太郎, 中田 秀基, 松岡 聡：アプリケーションのインストール、データの配布、更新をサポートするグリッドポータル構築ツールキット（PCT4G）の開発, 情報処理学会研究報告, 2003-HPC-95, pp.173-178, 2003.
8) NetSolve, http://icl.cs.utk.edu/netsolve/.
9) MPICH-G2, http://www.globus.org/mpi/.
10) expat, http://expat.sourceforge.net/.
11) Libxml2, http://www.xmlsoft.org/.
12) Xerces, http://xml.apache.org/xerces-c/.
13) GNU autoconf, http://www.gnu.org/software/autoconf/.