Relis-G：計算グリッドのための遠隔ライブラリインストール機構

渡邊啓正[†] 本多弘樹[†] 弓場敏嗣[†]
田中良夫[††] 佐藤三久[†††]

グリッドのユーザが GridRPC 等で使用されるライブラリをグリッド上の分散した複数のサーバへインストールする際，遠隔操作コマンドの入力の手間，不均質環境におけるインストール，冗長なコンパイルの回避，サーバの運用方針の遵守といった複雑で手間のかかる問題を解決する必要がある．本稿では，ユーザがこれらの問題を容易に解決できるようにするべく，グリッドで可搬性と汎用性の高い遠隔ライブラリインストール機構 Relis-G を提案する．本機構はライブラリの自動遠隔インストールのほかに，インストールパッケージの自動作成，冗長なコンパイルの自動的な回避，サーバの運用方針の自動的な遵守といった機能を提供する．また，動作試験により本機構が前述のユーザの負担を大きく軽減することを確認した．

Relis-G: Remote Library Install System for Computational Grids

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

When a grid user installs a library that is used by the GridRPC etc. to the servers distributed over a grid, the user needs to solve complicated and laborious problems, such as user's effort to input commands for remote operation, library installation in heterogeneous systems, avoidance of redundant compilation, and observance of administration policy of each server. In order to enable the user to solve these problems easily, we propose the remote library install system, Relis-G, which is highly portable and multipurpose in a grid. Besides the function of automatic remote library installation, this system offers functions, such as automatic creation of an installation package, automatic avoidance of redundant compilation, and automatic observance of administration policy of each server. Moreover, we confirmed that this system greatly mitigated the above-mentioned user's burden by operation tests.

1. はじめに

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効率良く利用するための基盤である，グリッドに関する研究が世界中でさかんに行われている．なかでも，グリッドの複数の計算資源を用いて大規模計算環境を構築することを目的とした計算グリッドは，高い計算性能を得る手段として期待され，様々な科学技術分野における次世代計算基盤として注目されている．

グリッドで動作するアプリケーションを開発する際，GridRPC[2] システムが用いられることが多い．これは従来の RPC（Remote Procedure Call）をグリッドに拡張したもので，グリッドの資源を利用する際の認証・資源管理・通信といった処理の複雑さを隠蔽するグリッドミドルウェアの 1 つである．この一例として，Ninf/Ninf-G[3] や NetSolve[4] がある．

GridRPC システムでは，計算依頼時にクライアントからサーバへ関数の引数や入力データが転送され，あらかじめサーバ上に登録されている関数が呼び出される（計算依頼時に関数のコードが転送されるシステムについては 5 章で触れる）．この方法の場合，呼び出される関数がサーバ上に登録されていない場合は，計算依頼に先立ってユーザがその関数を含むライブラリをサーバにインストールしなければならない．ライブラリのインストール作業は，ユーザがサーバ上でライブラリのソースファイルをコンパイル・リンクし，

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The 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・情報工学系計算科学研究センター
Institute of Information Sciences and Electronics/Center for Computational Sciences, University of Tsukuba

サーバ上の計算グリッド用サービスとして登録するという作業で構成される．また，これらの作業の詳細は使用されるグリッドミドルウェアによって決められる．

従来，ライブラリインストール作業は ssh や scp 等の遠隔操作ツールを用いてユーザの手作業で行われてきた．しかし，この方法には以下の問題がある．(1) 多数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，ユーザの作業の手間の量が多くなる．(2) ライブラリを開発するために，グリッドの不均質な計算機でコンパイル可能なインストールパッケージが作成される必要がある．(3) PC クラスタ等のグリッドの一部の均質な計算機群では，計算資源を効率良く使用するために，ライブラリの冗長なコンパイルは避けられるべきである．(4) 一般にサーバがユーザ以外の人に管理されるため，ライブラリインストール作業は各サーバの運用方針に沿って行われる必要がある．

ライブラリのインストールに関するこれらの問題は前述のグリッドミドルウェアでは解決されていない．また，既存の自動ソフトウェア配布システムはイントラネットでの利用を想定しており，グリッド環境での利用に向いていない（5 章を参照）．

本稿では，これらの問題を解決し，ライブラリのインストール作業におけるユーザの負担を軽減する機構 **Relis-G** の設計・実装について述べる．本機構は，ライブラリのインストール環境を示すファイルとともに，ユーザが 1 つのシェルスクリプトを実行するだけで，すべてのサーバにライブラリをインストールすることを可能とする．本機構の特徴は次のとおりである．(1) グリッドで標準的なセキュリティ技術に基づいて，ライブラリのインストール先のサーバが保護される．(2) グリッドの不均質な計算機上で動作できる．

本稿では，2 章で本機構の設計および実装について述べ，3 章で本機構の全体動作をまとめる．4 章で本機構の動作試験について述べ，5 章で関連研究に言及し，6 章で今後の展開を含めてまとめる．

2. 機構の設計と実装

本機構は次の機能で構成されている（**図 1** を参照）．

遠隔ライブラリインストール機能 遠隔サーバ群へのライブラリのインストールを自動的に行う．

インストールパッケージ自動作成機能 不均質なサーバ環境でコンパイル可能なライブラリのインストールパッケージを作成する．

コンパイルサーバ機能 均質なサーバ環境で冗長なコンパイルを自動的に回避する．

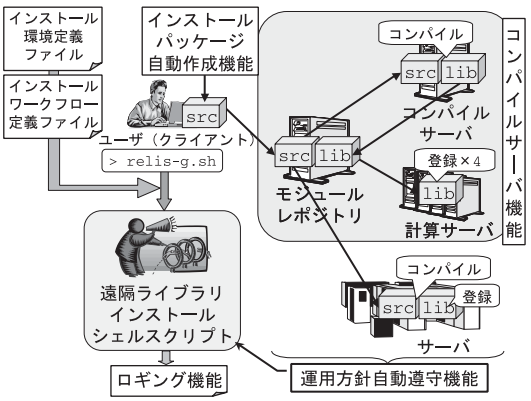

図 1 Relis-G の構成
Fig. 1 Structure of Relis-G.

運用方針自動遵守機能 サーバの運用方針にインストール方法を適合させる．

ロギング機能 遠隔ライブラリインストール中に出力されたメッセージのログをとる．

各機能の設計および実装について，以下に述べる．

2.1 遠隔ライブラリインストール機能

本機能は遠隔ライブラリインストールの基本的な機能性を提供する．

本機能は次の特徴を有していなければならない．

- 遠隔サーバアクセスのために，シングルサインオン機能を備えたユーザ認証機構を用いること．
- グリッドの不均質な計算機環境で動作可能であること．

これらの特徴を実現するべく，本機能は以下の方法で実現されている．

本機能は遠隔アクセスのために Globus Toolkit 2 [5] を用いている．このツールは以下の特徴を持つ．

- 多くのグリッド研究機関において事実上標準で利用されている．
- 公開鍵暗号や X.509 証明書に基づく，シングルサインオン機能を有するセキュリティインフラ（Grid Security Infrastructure：GSI）を含む．
- 遠隔にプロセスを起動する機能 GRAM（Globus Resource Allocation Manager）と，ファイルを転送する機能 GridFTP を有する．

また，本機能は Bourne シェルスクリプトで実装した．これは次の理由に基づく．

- グリッドの不均質な計算機において可搬性が高い．
- 複数のサーバへの並行インストールをプロセスの fork を用いて容易に実装できる．
- 遠隔アクセスツールのコマンド群を組み合わせることで本機能を実装できる．

本機能はライブラリのインストールを行う環境に関する情報を受け取るためにユーザ定義のテキストファイルを用いる．この方式は，多くの場合にシェルが端末として使用される，グリッドアプリケーションの開発プロセスに適している．

それらはインストール環境定義ファイル，およびインストールワークフロー定義ファイルである．ユーザはいったんこれらのファイルを作成すれば，同じインストール環境に対して同じライブラリインストール手順を容易に何度も実行できる．したがって，本機構は，ライブラリのデバッグ等，複数回のライブラリインストール作業を必要とする状況において大変有用である．

また，このファイルの分離は次の理由に基づく．すなわち，インストールワークフロー（インストールのためのコマンド列）の定義は，サーバ環境によって変化せず，また同じグリッドミドルウェアが使われている限り再利用可能である．

これらのファイルの詳細について，以下に述べる．

インストール環境定義ファイル

このファイルはライブラリのインストールが行われるサーバ環境についての情報を記述したものである．

このファイルは XML で記述されている．これは多数の設定項目をユーザが記述するために，XML の木構造が有する高い可読性が有用であるからである．

このファイルは大別すると，以下の 3 つの設定項目で構成されている（**図 2** を参照）．

- ライブラリのインストール手順に関する設定（約 11 行）
  3 行目以降の <client> タグ内にあたる．ライブラリのソースファイル群の存在するフォルダのパス，コンパイルサーバ機能の使用の有無，使用するインストールワークフロー定義ファイル（後述）等を指定する．サーバ環境にかかわらず，ユーザは必ずこの設定を指定する必要がある．
- モジュールレポジトリ（後述）に関する設定（1 ホストにつき約 6 行）
  15 行目以降の <modreps> タグ内にあたる．モジュールレポジトリ上の GRAM や GridFTP のサービスに遠隔にアクセスするための情報を指定する．ユーザは基本的にモジュールレポジトリごとにこの設定を記述する必要がある．
- サーバ群に関する設定（1 ホストにつき約 6 行）
  25 行目以降の <servers> タグ内にあたる．各サーバ上の GRAM や GridFTP のサービスに遠隔にアクセスするための情報を指定する．ユーザは基本的にサーバごとにこの設定を記述する必要がある．

```
<?xml version='1.0'?>
<install-env>
  <client>
    <host name="`hostname`"/>
    <user name="watanabe"/>
    <library name="mmul" version="1.0">
      <source dir="./mmul/server"/>
      <configure create="yes" update="no"
       staticlink="yes"/>
      <compile always="no"/>
      <workflow file="./wffile.xml"/>
      <mfheader package="globus_common"/>
    </library>
  </client>
  <modreps>
    <modrep-group>
      <host name="gex1.yuba.is.uec.ac.jp"/>
      <rsh type="globus-gram"/>
      <rsh type="ssh"/>
      <dbhost type="globus-mds"/>
      <dbhost type="https"
       file="/adminpolicy.tar.gz"/>
    </modrep-group>
  </modreps>
  <servers>
    <server-group>
      <host
       name="(gex1|grid0[0-3]).yuba.is.uec.ac.jp"/>
      <rsh type="globus-gram"/>
      <dbhost name="gex1.yuba.is.uec.ac.jp"
       type="globus-mds"/>
      <mrhost name="gex1.yuba.is.uec.ac.jp"/>
    </server-group>
    <server-group>
      <host name="asuka.yuba.is.uec.ac.jp"/>
      <rsh type="globus-gram"/>
      <dbhost type="globus-mds"/>
      <mrhost name="gex1.yuba.is.uec.ac.jp"/>
    </server-group>
    <server-group>
      <host name="latitude6.yuba.is.uec.ac.jp"/>
      <rsh type="ssh"/>
      <dbhost type="https"
       file="/adminpolicy.tar.gz"/>
      <mrhost name="gex1.yuba.is.uec.ac.jp"/>
    </server-group>
    <server-group>
      <host name="(koume|koume0[0-3]).hpcc.jp"/>
      <rsh type="ssh"/>
      <dbhost name="gex1.yuba.is.uec.ac.jp"
       type="globus-mds"/>
      <mrhost name="gex1.yuba.is.uec.ac.jp"/>
    </server-group>
    <use only="" never=""/>
  </servers>
</install-env>
```

図 2 インストール環境定義ファイル
Fig. 2 Installation environment definition file.

インストール環境定義ファイルでは，サーバ群のホスト名の記述に node[0-7].abc.com といった正規表現を用いることができる．これにより，クラスタ環境でディスク領域が NFS で共有されていない場合に，ユーザはノードごとのインストールを容易に行える．

このファイルを処理するために，インストール環境定義ファイルの内容をシェルスクリプト形式で書き出すプログラムを，軽量 XML パーサライブラリ Crimson [6] を用いて Java 言語で開発した．このプログラムは前述の正規表現の展開も行う．Java による実装はプラットフォーム独立性という点でグリッドの不均

```
<?xml version='1.0'?>
<process name="Ninf-G_Install">
  <require-env name="NS_DIR"/>
  <taskgroup type="oncompileserver">
    <sequence name="main1">
      <exec executable="./configure"/>
      <exec executable="make"/>
      <exec executable="${NS_DIR}/bin/ns_gen">
        <arg line="*.idl"/>
      </exec>
      <exec executable="make">
        <arg line="-f ${LIBRARY_NAME}.mak"/>
      </exec>
    </sequence>
  </taskgroup>
  <distfile file="${LIBRARY_NAME}.o"/>
  <taskgroup type="oncomputationserver">
    <sequence name="main2">
      <exec executable="make">
        <arg line="-f ${LIBRARY_NAME}.mak
install"/>
      </exec>
    </sequence>
  </taskgroup>
</process>
```

図 3 インストールワークフロー定義ファイル
Fig. 3 Installation workflow definition file.

質計算機環境に適している．また，クライアントホスト上での Java VM の要求は大きな障害にならない．

インストールワークフロー定義ファイル

このファイルは本研究で提案される，グリッドミドルウェア間で統一化されたインストールワークフロー（インストール用コマンド列）を XML で記述したものである．

これにより，本機構はグリッドにおける汎用的な自動ソフトウェア配布システムとして使用可能となる．本機構が任意のインストールワークフローを実行可能とするには，コマンド列がグリッドミドルウェア間で統一化された書式で記述される必要がある．このファイルの書式は次の要件を満たすように設計した．

- シェルに依存しない
  ワークフローが様々なワークフロー処理系に容易に流用されるようにするため，抽象的なプロセス表現を用いる必要がある．また，ユーザはシェルスクリプトについての知識を強要されるべきではない．
- インストール作業プロセスの記述に洗練された仕様を用いる
  ユーザにインストール作業プロセスを記述しやすくするため．

インストールワークフロー定義ファイルの内容は以下のとおりである（**図 3** を参照）．

- 各コマンドの記述（1 コマンドあたり約 3 行）
  6 行目以降の <exec> タグにあたる．Apache Ant[7] のビルドファイルで使用可能なタスク（Exec，Copy 等）を模した書式になっている．Ant のビルドファイルのタスクの書式はシェルに依存しないコマンドの表現として有用である．
- 複数のコマンド間の制御フローの記述（1 フローあたり約 2 行）
  前述のコマンド群を囲む 5 行目の <sequence> タグにあたる．ビジネスプロセス実装言語である BPEL4WS[8] の構造化アクティビティ（Sequence，While 等）を模した書式になっている．BPEL4WS はビジネスプロセスが自然に記述されるようにするため，とても洗練されている．
- タスク群の分割（約 4 行）
  前述の制御フロー群を囲む 4 行目の <taskgroup> タグにあたる．コンパイルサーバ機能（後述）のために，コンパイルサーバ上と計算サーバ上で実行されるフローに分けてタスク群を記述する．また，コンパイルサーバから計算サーバへ転送されるべきファイル群は 16 行目の <distfile> タグによって指定される．

このファイルを処理するために，インストールワークフロー定義ファイルを読み込んで相当するコマンド列をシェルスクリプト形式で書き出すプログラムを，インストール環境定義ファイルと同様に Crimson を用いて Java 言語で開発した．

インストールワークフローを実行する際，特定の環境変数にサーバ固有の値が定義されていなければならない場合がある（グリッドミドルウェアのインストールパス等）．この値を事前に取得する手段として，本機構では 2.4 節の運用方針自動遵守機能が利用できる．その際，ユーザが環境変数名をこのファイルの中で指定する（3 行目の <require-env> タグ）．

2.2 インストールパッケージ自動作成機能

本機能は，グリッドの不均質な計算機上でのライブラリのコンパイルのために，ビルド用ファイル群（configure スクリプト，Makefile 等）を自動作成する．

不均質なサーバ環境では，各サーバで個別のライブラリコンパイル手順が必要になる場合がある（特定のシステムで特定のコンパイルオプションが必要になる等）．その場合，各サーバに合ったコンパイル手順を実行するために，ビルド用ファイル群（特に configure スクリプト）が必要になる．

本機能では，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[9] を用いて configure スクリプトが作成される．autoconf の入力ファイルは本機能によってライブラリのソースファ

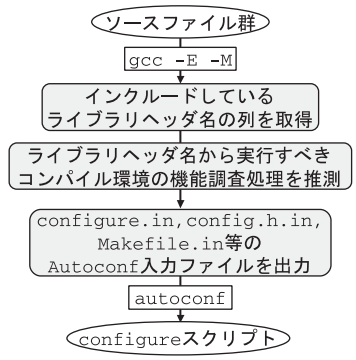

図 4 インストールパッケージ自動作成機能
Fig. 4 Installation package automatic creation function.

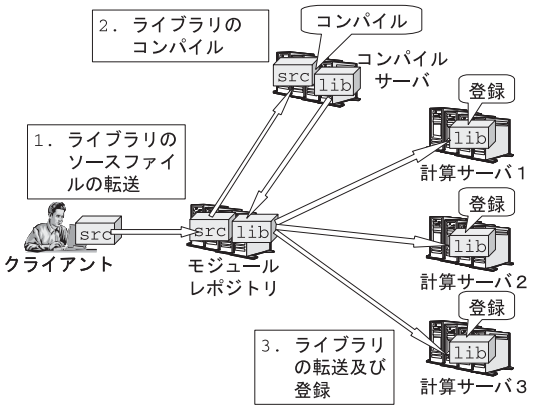

図 5 コンパイルサーバ機能
Fig. 5 Compile server function.

イルから自動作成される．その際，実行されるべきコンパイル環境の機能調査処理は，ソースファイルでインクルードされているライブラリヘッダ名をもとに割り出される（**図 4** を参照）．これは多くの機種依存性が外部ライブラリの使用に起因すると判断したためである．

2.3 コンパイルサーバ機能

本機能は，ライブラリのコンパイルをコンパイルサーバだけで行い，作成されたライブラリを他のサーバ（計算サーバ）へ転送して登録するということを自動的に行う（**図 5**）．これにより，ライブラリの冗長なコンパイルが回避可能となる．

近年の計算グリッドにおける複数のサイトに分散配置されたクラスタのように，均質なサーバ環境では，本機能の有用性は高い．

本機能を実現するには，次の問題を解決する必要がある．

- 計算サーバに対するコンパイルサーバの特定方法およびコンパイルサーバにアクセスするための情報の取得方法
- ファイル転送用サーバの配置方法

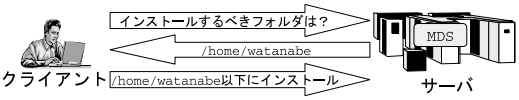

図 6 運用方針自動遵守機能
Fig. 6 Administration policy automatic observance function.

最初の問題について，本機能は，ライブラリのインストールに際し，各計算サーバの運用方針情報データベースに，そのサーバに対応するコンパイルサーバを問い合わせる（2.4 節を参照）．なお，この方法でコンパイルサーバへアクセスするための情報が得られなかった場合，本機能は従来どおりその計算サーバ上でライブラリのコンパイルと登録を行う．

2 つめの問題について，本機能は，サーバ間のライブラリファイルの転送の経由点として，GridFTP のファイル転送用サーバを最低 1 つ必要とする．このホストをモジュールレポジトリと呼ぶ．モジュールレポジトリはクライアントとサーバの両側からアクセスでき，ユーザの権限で任意のファイルがアップロード/ダウンロードされうる必要がある．

モジューレポジトリでは，複数の遠隔インストール作業が同時に行われる可能性がある．そのため，各ライブラリは次の情報に基づき区別して管理される．

- ユーザ名
- ライブラリ名
- ライブラリバージョン番号
- インストール作業実行日時
- コンパイルサーバホスト名

2.4 運用方針自動遵守機能

本機能は各計算資源の運用方針を自動的に取得し，それに従ってライブラリのインストールを行う．

計算資源の運用方針情報は Globus Toolkit に含まれる MDS という情報提供サービス（GSI を通してアクセスされる LDAP データベース）を利用したデータベースで管理される．そして，本機能は次の手順で実装されている（**図 6** を参照）．

- あらかじめ，計算資源の管理者がライブラリのインストールに対する運用方針情報を，資源上の MDS に登録しておく．登録作業には煩雑なコマンド入力作業が必要となる．そのため，その管理者は，本機能が提供する，その作業を自動的に行えるシェルスクリプトを利用できる．
- 本機構のシェルスクリプトは，インストール実行直前に，MDS 検索コマンド（grid-info-search）

を用いてユーザの権限で MDS から運用方針情報を読み出す．そして，それに従ってライブラリのインストールを行う．

計算サーバの運用方針情報は次のとおりであるが，今後必要に応じて拡張可能である．

- ライブラリをインストールするトップフォルダ（ライブラリはこの下にユーザ名・ライブラリ名・ライブラリバージョン番号で区別して置かれる）
- コンパイルサーバへ接続するための情報
- ライブラリのインストールに必要な環境変数の値

2.5 ロギング機能

本機能はインストール作業の結果をサーバごとのログファイルに出力する．

インストールが失敗した場合，どのホストにおいて，何が原因でインストールに失敗したのかがログファイルに出力される（遠隔ファイル転送の失敗等）．ライブラリのビルド時のエラーでは，インストールワークフローの実行時の標準出力と標準エラー出力の内容がサーバごとのファイルとして保存される．これらによってユーザはライブラリのインストールの結果を追跡することができる．なお，ユーザは本機構のシェルスクリプトを呼び出す際にログファイルのパスと，ビルドエラーの出力先フォルダを指定する．

2.6 適応範囲

本機構は，GridRPC のライブラリに限らず，MPI のアプリケーション等のインストールにも適用可能である．

サーバ間でインストールワークフローが異なる場合，インストール環境定義ファイルとインストールワークフロー定義ファイルは別途新たに作成される必要がある．これは，本機構でのインストールにおいて，1 つのサーバ群に対して 1 つしかインストールワークフローが適用できないためである．

ライセンス等が原因で特定のサーバのみで動作するライブラリをリンクする場合，コンパイルサーバ機能を使用するべきではない．その場合，インストール前にそのようなライブラリをリンクすることをユーザが把握して，コンパイルサーバ機能を使用しないようにインストール環境定義ファイルを作成する必要がある．

2.7 Globus Toolkit が利用できない場合への対応

本機構では遠隔アクセスツールとして Globus Toolkit の代わりに ssh が利用できる．シングルサインオン機能は ssh-agent を用いて実現される．また，運用方針自動遵守機能において，Globus Toolkit の MDS の代わりに https サーバが使用できる．MDS の場合と異なるのは以下の 2 点である．(1) ユーザがサーバの認証局（CA）のルート証明書のパスをインストール環境定義ファイルに記述する．(2) 本機構のシェルスクリプトが MDS 検索コマンドの代わりに https クライアントプログラム（本機構が提供）を用いる．

3. 機構の全体動作

2 章で述べた機能群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運用方針情報を各 MDS に登録しておく．そして，ユーザは次の作業を行う．

(1) ライブラリのソースファイル群を用意する．
(2) インストール環境定義ファイルとインストールワークフロー定義ファイルを作成する．
(3) クライアントホスト上で grid-proxy-init コマンドを実行し，プロキシ証明書を作成する．
(4) 本機構のシェルスクリプトを起動する．

本機構のシェルスクリプトの内部で実行される処理は次のとおりである（図 1 を適宜参照）．

(1) （以下の処理のログを随時書き出す）．
(2) インストール環境定義ファイルとインストールワークフロー定義ファイルを読み込む．
(3) 必要であればインストールパッケージを自動作成する．
(4) 各資源の運用方針情報を MDS から取得する．
(5) ライブラリのソースファイル群をモジュールレポジトリへ転送する．
(6) コンパイルサーバに対して，モジュールレポジトリからライブラリのソースファイル群をダウンロードし，ライブラリをコンパイルして，ライブラリファイルをモジュールレポジトリへアップロードするように，遠隔コマンド実行を行う．
(7) 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ダウンロードして登録するように，遠隔コマンド実行を行う．

4. 動作試験および評価

4.1 動作試験環境

サーバ群が不均質で，かつクラスタ環境のあるグリッドを再現するため，本機構の動作試験を，**表 1** の計算機群を用いて下記の設定で行った．

- サイト 1 の 1 ホストをクライアントホストとし，表 1 の 12 ホストすべてをインストール先サーバとした．また，サイト 1 にモジュールレポジトリを 1 つ指定した．
- インストールを行うライブラリは，Ninf-G を対

表 1 動作試験環境
Table 1 Operation test environment.

サイト 1（電気通信大学）
Globus Toolkit 2.4.3, Ninf-G Ver. 1.1.1, RedHat Linux 7.3 × 5 台
Globus Toolkit 2.4.3, Ninf-G Ver. 1.1.1, Solaris 8 × 1 台
OpenSSH 3.1p1, Ninf-G Ver. 1.1.1, RedHat Linux 7.3 × 1 台
サイト 2（産業技術総合研究所）
OpenSSH 3.6.1p2, Ninf-G Ver. 1.1.1, RedHat Linux 7.3 × 5 台

象とした行列積の関数 1 つだけを含むものとした．

- 動作試験環境においてコンパイル方法に明確な機種依存性がある，socket ライブラリをライブラリのソースファイルでインクルードし，インストールパッケージ自動作成機能を使用した．
- サイト 1 でライブラリのコンパイル回数を 4 回削減，サイト 2 でコンパイル回数を 4 回削減するように，運用方針自動遵守機能を用いて各サーバに対してコンパイルサーバを設定し，コンパイルサーバ機能を使用した．
- サイト 1 のサーバ 4 台とサイト 2 のサーバ 4 台ではホスト名が連番であったため，インストール環境定義ファイルでは正規表現を用いてホスト名を略記した．
- Globus Toolkit だけでなく，ssh，scp も遠隔アクセスツールとして使用した（2.7 節を参照）．
- ログファイルとエラー出力フォルダを指定し，ロギング機能を使用した．
- 本試験で用いたインストール環境定義ファイルとインストールワークフロー定義ファイルは，それぞれ図 2（全 50 行）および図 3（全 24 行）にあげたものである．

4.2 動作試験結果

動作試験の結果，出力されたログを参照し，上記の設定どおりに，すべてのサーバへ遠隔にライブラリのインストールが行われたことを確認した．また，本機構が自動作成したインストールパッケージが不均質サーバ環境で正しくコンパイルできたこと，均質サーバ環境においてライブラリのコンパイル回数が削減されたことを確認した．さらに，インストール環境定義ファイルでは正規表現を用いてサーバ群を指定することで 8 ホスト分（48 行）の設定記述の手間を省くことができた．

4.3 インストールに要する時間の評価

インストールに要する時間（以下，インストール時間）について従来手法と本機構を比較する．インストール時間は，コマンド入力等のユーザのキーボード入力に要する時間（以下，キーボード入力時間）と，指示された計算処理の実行するのに要する時間（計算機処理時間）の和として算出された．

表 2 動作試験環境における従来手法のインストール時間
Table 2 Installation time in the conventional method at the operation test environment.

正しく入力すべき文字数	2741
ミス修正文字数	182
総入力文字数	2923
キーボード入力時間	48 分 43 秒
総入力コマンド数	121
計算機処理時間	1 分 22 秒
インストール時間	50 分 5 秒

インストールを行うサーバ環境は前述の動作試験と同じとした．サーバの台数が 12 台を超える場合には表 1 のサーバ群が同じ構成比で複数群存在するものと仮定して算出を行った．キーボード入力時間は以下の前提に基づき算出された．

- 1 文字あたりの入力時間は 1 秒
- 入力ミスの頻度は 30 字に 1 回

計算機処理時間は，gettimeofday 関数を用いて測定した各コマンドに要する時間の和として算出された．

動作試験環境におけるインストール時間

表 2 に従来手法における動作試験環境（表 1）のサーバ群に対するインストール時間とその内訳を示す．従来手法では，複数のインストールは逐次的に実行されると仮定した．

表 3 に同環境の本機構におけるインストール時間とその内訳を示す．コンパイルサーバを使用せずオブジェクトファイルの使用に依存関係がない場合，本機構は自動的に並列にインストールを行う．

表 3 から，本機構のインストール時間の大部分を定義ファイルの作成のためのキーボード入力時間が占めていると分かる．

多数サーバにおけるインストール時間の比較

動作試験環境よりも多数のサーバを用いる場合の従来手法と本機構のインストール時間を，動作試験環境の測定データに基づき算出した．算出結果を**図 7** に示す．この結果から，(次の段落と比較して）定義ファイルを再利用せず，サーバ台数が 12 台以上の場合，従来手法よりも本機構の方がインストール時間が短いと分かる．

定義ファイルの再利用の効果

2 章で述べたように，本機構では，いったんインストール環境定義ファイル・インストールワークフロー

表 3 動作試験環境における本機構のインストール時間
Table 3 Installation time in Relis-G at the operation test environment.

コンパイルサーバ	使用	非使用
正しく入力すべきコマンド用文字数	180	180
インストール環境定義ファイルの文字数	1571	1572
インストールワークフロー定義ファイルの文字数	679	679
ミス修正文字数	162	162
総入力文字数	2592	2593
キーボード入力時間	43 分 12 秒	43 分 13 秒
総入力コマンド数	4	4
本機構以外の計算機処理時間	0.35 秒	0.35 秒
本機構のシェルスクリプトの処理時間の内訳		
初期化処理	0.57 秒	0.57 秒
インストール環境定義ファイルの変換・読み込み	0.55 秒	0.54 秒
インストールワークフロー定義ファイルの変換・読み込み	0.55 秒	0.55 秒
インストールパッケージ自動作成	1.13 秒	1.13 秒
運用方針自動取得	10.63 秒	4.83 秒
（並列）インストール	29.86 秒	15.98 秒
終了処理	0.01 秒	0.01 秒
計算機処理時間	43 秒	23 秒
インストール時間	43 分 55 秒	43 分 36 秒

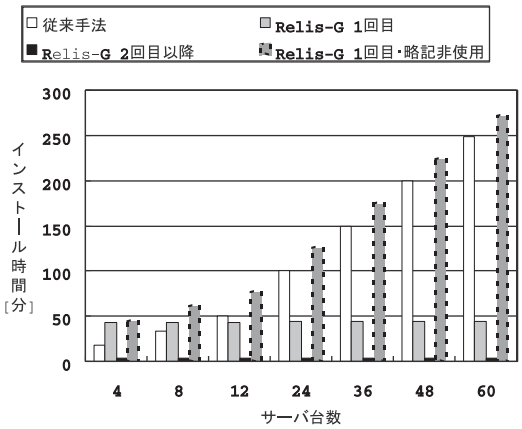

図 7 従来手法と本機構のインストール時間
Fig. 7 Installation time in the conventional method and Relis-G.

定義ファイルを作成すれば，同じインストール環境に対して再度インストールを行う際にそれらのファイルを再利用できる．図 7 に定義ファイルを再利用した場合のインストール時間を示す．この結果から，ユーザがデバッグ等の理由でライブラリを頻繁に更新する際，

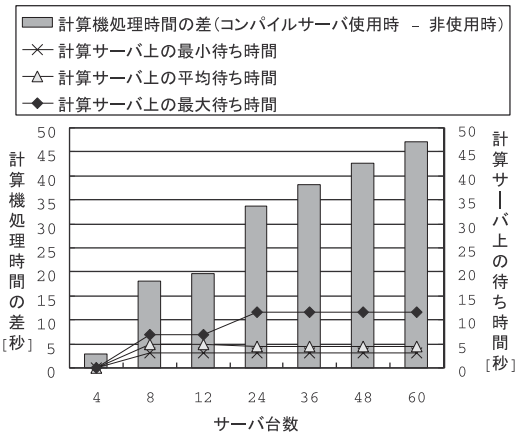

図 8 コンパイルサーバ機能の計算機処理時間への影響
Fig. 8 Impact of the compile server function to the computer processing time.

本機構は，2 回目以降のインストールにおけるユーザの手間を大幅に軽減すると分かる．

インストール環境定義ファイル中の略記の効果

インストール環境定義ファイルでは，複数のサーバに対して使用される遠隔アクセスツールやモジュールレポジトリ等の遠隔アクセス方法が同じ場合，サーバ群に対する設定を正規表現を用いて短く記述できる（2.1 節を参照）．インストール環境定義ファイル中の略記を使用しない場合の本機構のインストール時間を図 7 に示す．この結果から，インストール環境定義ファイル中の略記によってインストール時間が大幅に短縮されると分かる．

コンパイルサーバ機能の計算機処理時間への影響

コンパイルサーバ機能では，コンパイルサーバ上でライブラリファイルが作成されてモジュールレポジトリへ転送されるまで，計算サーバ上で動作しているインストール用シェルスクリプトは待機する．これらの処理による計算機処理時間の増加量と計算サーバ上の待ち時間を**図 8** に示す．この結果から次のことが分かる．(1) 3 秒から 48 秒程度計算機処理時間が長くなる．主な原因はコンパイルサーバ上で作成されたオブジェクトファイルの転送，および運用方針自動遵守機能による計算サーバに対応するコンパイルサーバの問合せである．(2) 計算サーバ上の待ち時間はサーバ台数によらず平均 5 秒程度である．

5. 関 連 研 究

計算機群へのプログラムのインストールを容易にするシステムとして，オフィス PC 群等のイントラネット内で利用される自動ソフトウェア配布システム[10)~18)]

がある．しかし，これらは以下のようにグリッドでの利用に向いていない．(1) グリッドで必要な SSL 等の技術に基づくセキュリティメカニズムがサーバ上に配備されていない．(2) 不均質計算機上で動作するように実装されてはいない．(3) コンパイル済みのソフトウェアの配布を想定しているため，不均質計算機上でのソフトウェアのコンパイルに対応できない．

InstallShield 等の既存のソフトウェアインストールパッケージ作成システム[19)~23)] には，本機構のように，グリッドの不均質な計算機上でコンパイル可能なインストールパッケージを自動作成するものはない．

白砂らの PCT4G[24)] はグリッドポータル構築ツールキットであり，ユーザがサーバ群に対してライブラリの遠隔インストールを行える．本機構と同様に Globus Toolkit とシェルスクリプトによって遠隔インストールを実現している．本機構ではインストールワークフローがシェルに依存しないインストールワークフロー定義ファイルとして記述されるが，PCT4G ではシェルスクリプトで記述される．また，PCT4G には，不均質環境向けのインストールパッケージの自動作成，冗長なコンパイルの自動的な回避，資源の運用方針の自動的な遵守といった機能が含まれていない．加えて，インストール環境定義ファイルのようにインストール環境の情報を略記できないため，サーバの台数が多い場合にインストール実行者の負担が大きい．

計算依頼時にサーバへプログラムコードを自動的に転送可能なグリッドミドルウェアがある[25),26)]．これらの特長として，ユーザが明示的なプログラムインストール作業をいっさい行う必要がなく，サーバ群へのプログラムの配備が容易に行えることがあげられる．しかし，Java 言語で書かれたプログラムの実行しか行えないといった問題をかかえている．また，GridFTP 等の高性能データ転送技術を用いてプログラムを効率良く配布することが難しい．

他のグリッドミドルウェア[4),27)~29)] にプログラムの自動転送機能を提供するものがあるが，いずれにおいてもグリッドの不均質環境におけるプログラムのコンパイル等に関する問題が解決されていない．

6. まとめと今後の課題

我々は遠隔ライブラリインストール機構 Relis-G を設計・実装した．グリッドアプリケーションの開発プロセスにおいて，本機構は，手間のかかるライブラリインストール作業を，従来手法のようなサーバ群へのログインをせずに，ユーザが容易に実行可能とする．本機構は，GridRPC をはじ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し，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．

本機構の今後の課題は以下のとおりである．

- グリッドポータル等のグリッドシステムへの組み込み，および API の作成
- スケーラビリティ等の性能評価
- ライブラリのテストラン，アンインストール機能の追加
- ライブラリ更新時における変更ファイルのみの転送による効率的なライブラリインストール
- 自動分散化コンパイラ OpenGR[30)] との併用
- 既存のライブラリの自動チューニング機構との併用
- インストール環境定義ファイルの作成支援
- グリッドサービスとしての実装による OGSA（Open Grid Service Architecture）への対応

謝辞 本研究を進めるにあたり，数多くのご助言，ご協力をいただいた，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，株式会社 SRA（Software Research Associates）の平野基孝氏に深く感謝いたします．

参 考 文 献

1) 渡邊啓正，本多弘樹，弓場敏嗣，田中良夫，佐藤三久：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，インターネットカンファレンス 2003 論文集，pp.27–34 (2003).
2) Seimour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C. and Casanova, H.: Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *Proc. Grid Computing — Grid 2002*, pp.274–278 (2002).
3) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T. and Matsuoka, S.: 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). http://ninf.apgrid.org/
4) NetSolve. http://icl.cs.utk.edu/netsolve/
5) Globus Toolkit. http://www.globus.org/
6) Crimson. http://xml.apache.org/crimson/
7) Apache Ant Project. http://ant.apache.org/
8) Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/
9) GNU autoconf. http://www.gnu.org/software/autoconf/

10) 富士通株式会社：Systemwalker. http://systemwalker.fujitsu.com/jp/
11) ノベル社：ZENworks 6. http://www.novell.co.jp/
12) 株式会社ハンモック：PALLET CONTROL. http://www.hammock.jp/pallet/
13) クオリティ株式会社：QND Plus. http://www.quality.co.jp/
14) 荒川修一，田中功一，武田光世，松井陽子：ソフトウェア配布システム RSU の開発，情報処理学会研究報告「マルチメディア通信と分散処理」，Vol.1994, No.065-021 (1994).
15) 高宮安仁，真鍋 篤，松岡 聡：Lucie: 大規模クラスタに適した高速セットアップ・管理ツール，情報処理学会論文誌：コンピューティングシステム，Vol.44, No.SIG11, pp.79–88 (2003).
16) 藤生正樹，箱崎勝也：Java 分散オブジェクトを利用したアプリケーション・インストーラの開発，情報処理学会研究報告「分散システム/インターネット運用技術」，No.015-005, pp.25–30 (1999).
17) 手塚 悟，木原健一，三宅 滋，古川 博，本林 繁，露木陽介：パソコン LAN システム構築支援ツール：Easy Installer, 情報処理学会論文誌，Vol.37, No.02-013, pp.300–311 (1996).
18) Digital Factory Inc.: Kondara MNU/Linux HPC 2.0. http://www.digitalfactory.co.jp/
19) InstallShield Software Corporation. http://www.installshield.com/
20) Zero G Software Inc.: InstallAnywhere. http://www.zerog.com/
21) 日本グレープシティ株式会社：InstallStudio. http://www.grapecity.com/Japan/
22) MindVision Software: Installer VISE. http://www.mindvision.com/
23) Wise Solutions Inc.: Wise Installation System. http://www.wise.com/
24) 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, SWoPP 2003, pp.173–178 (2003).
25) Suzumura, T., Matsuoka, S. and Nakada, H.: A Jini-based Computing Portal System, *Proc. SC2001* (2001).
26) 中田秀基，松岡 聡，関口智嗣：Java による階層型グリッド環境 Jojo の設計と実装，情報処理学会論文誌：コンピューティングシステム，Vol.44, No.SIG 11, pp.46–56 (2003).
27) Condor. http://www.cs.wisc.edu/condor/
28) Legion. http://www.cs.virginia.edu/~legion/
29) Nimrod. http://www.csse.monash.edu.au/~davida/nimrod.html/
30) Hirano, M., Sato, M. and Tanaka, Y.: OpenGR: A Directive-based Grid Programming Environment, *Proc. 5th International Symposium, ISHPC 2003*, International Workshop on OpenMP: Experiences and Implementations, pp.552–563 (2003).

(平成 16 年 1 月 31 日受付)
(平成 16 年 5 月 21 日採録)

渡邊 啓正（学生会員）

2003 年電気通信大学情報工学科卒業．同年より同大学大学院情報システム学研究科博士前期課程在学中．計算グリッドのプログラム開発支援環境に関する研究に従事．情報処理学会第 65 回全国大会にて学生奨励賞受賞．

本多 弘樹（正会員）

1984 年早稲田大学理工学部電気工学科卒業．1991 年同大学大学院理工学研究科博士課程修了．1987 年より同大学情報科学研究教育センター助手．1991 年より山梨大学工学部電子情報工学科専任講師．1992 年より同助教授．1997 年より電気通信大学大学院情報システム学研究科助教授．並列処理方式，並列化コンパイラ，並列計算機アーキテクチャ，グリッド等の研究に従事．工学博士．電子情報通信学会，IEEE-CS，ACM 各会員．平成 15 年度山下記念研究賞受賞．

弓場 敏嗣（正会員）

1966 年神戸大学大学院工学研究科修士課程修了．（株）野村総合研究所を経て，1967 年通商産業省工業技術院電子技術総合研究所（現在，独立行政法人産業技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機等の研究開発に従事．その間，計算機方式研究室長，知能システム部長，情報アーキテクチャ部長等を歴任．1993 年より，電気通信大学大学院情報システム学研究科教授．並列処理・分散処理の科学技術一般に興味を持つ．工学博士．情報処理学会，電子情報通信学会各フェロー．日本ソフトウェア科学会，日本ロボット学会，ACM，IEEE 各会員．

田中　良夫（正会員）

1965 年生．1995 年慶応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．1996 年技術研究組合新情報処理開発機構入所．2000 年通産省電子技術総合研究所入所．2001 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．

佐藤　三久（正会員）

1959 年生．1982 年東京大学理学部情報科学科卒業．1986 年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．1991 年通産省電子技術総合研究所入所．1996 年新情報処理開発機構並列分散システムパフォーマンス研究室室長．2001 年より，筑波大学電子情報工学系教授．同大学計算科学研究センター勤務．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術，グリッドコンピューティング等の研究に従事．IEEE，日本応用数理学会各会員．