Globus による Grid RPC システムの実装と評価

田中良夫[†] 中田秀基[†] 平野基孝[††]
佐藤三久[†††] 関口智嗣[†]

我々は，世界に広く普及することを目指した Grid RPC システムである Ninf-G の設計および実装を進めている．本稿では，Ninf-G の設計，実装および簡単な性能評価について報告する．Ninf-G は Globus Toolkit を用いて実装されており，ユーザ認証や資源管理など様々な要素を Globus の機能に吸収させる事が可能となる．また，Globus Toolkit を用いて実装されている他のグリッドソフトウェアと同じようなインタフェースを提供することができる．LAN/WAN 環境のいずれにおいても Globus の機能を用いたリモートライブラリ呼び出しのオーバーヘッドは 1 秒程度であった．

Implementation and Evaluation of Grid RPC system based on the Globus Toolkit

Yoshio Tanaka,[†] Hidemoto Nakada,[†] Motonori Hirano,[††]
Mitsuhisa Sato[†††] and Satoshi Sekiguchi[†]

We have desgined and implemented the Grid RPC system called Ninf-G. This paper reports on the design, impelementation and preliminary performance evaluation of Ninf-G. Ninf-G is implemented based on the Globus Toolkit. Various low-level services such as user authentication and resource managements can be implemented and absorbed by the Globus Toolkit. Ninf-G provides the similar interface with other Grid softwares which are implemented using the Globus Toolkit.

1. はじめに

近年，広域ネットワーク環境において高性能計算機，データ，特殊な装置などの様々な資源を容易にかつ効率良く利用するためのインフラストラクチャおよびその技術である**グリッド**に関する研究が世界中で盛んに行なわれている[1]．グリッドは単体のスーパーコンピュータの計算性能を越える大規模計算環境，ペタバイト級の大規模データ処理，あるいは光学電子顕微鏡等特殊な装置の遠隔利用など様々な科学技術分野に貢献する．

Grid RPC はグリッドにおける遠隔手続き呼び出し (Remote Procedure Call, RPC) によるプログラミングモデルである[2]．Grid RPC システムとしては，Ninf[3]，NetSolve[4]，Nimrod/G[5] などクライアント / サーバモデルに基づくシステム (Network Enabled Server システム，NES システム) がいくつか研究，開発されている．Grid RPC はグリッドにおける標準的なプログラミングモデルのひとつとして有望であり，特にタスク並列なプログラムに対する有効なプログラミングモデルとして注目されている．現在 Global Grid Forum[6] の Advanced Programming Models Working Group においても Grid RPC におけるプロトコルの洗い出しや既存システムのサーベイなどの調査が行なわれている．

我々は，世界に広く普及することを目指した Grid RPC システムである Ninf-G の設計および実装を進めている．Ninf-G はグリッドにおけるソフトウェアの様々な構成要素 (セキュリティや資源管理など) を実装するためのツールキットとして世界に広く普及している Globus Toolkit[7] を用いて実装されている．Ninf-G は現在まだ開発中であるが，本稿では Ninf-G の設計および実装の方針を述べ，現時点での簡単な性能評価を行なう．次節では一般的な Grid RPC システムを構成する基本コンポーネントを示し，Ninf-G の設計および実装の方針を述べる．3節では Globus Toolkit の概要を説明し，4節では Globus Toolkit を用いた Ninf-G の実装方法を述べる．5節では Ninf-G の簡単な性能評価を行ない，最後にまとめと今後の展望を述べる．

† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†† (株)SRA
Software Research Associates, Inc.
††† 筑波大学
University of Tsukuba

2. Ninf-G の設計方針

Grid RPC システムの基本機能は，下記の 3 つのコンポーネントにより実装される．これらの 3 つのコンポーネントを基本コンポーネントと呼ぶ．

- 遠隔手続きを呼び出して計算を依頼するコンポーネント． NES システムではクライアント．本稿でも，このコンポーネントを**クライアント**と呼ぶ．
- クライアントからの呼び出し (計算依頼) を受け，実際に手続きを実行するコンポーネント． NES システムにおいてはサーバ．本稿でも，このコンポーネントの事を**サーバ**と呼ぶ．
- 実際の計算を行なうライブラリルーチンを組み込んだ，遠隔呼び出し可能な手続きである**リモートライブラリ**．一般にリモートライブラリはクライアントに通知するための引数情報を含む実行形式として実装され，クライアントと通信して計算を行なう．

実際のグリッドにおいて Grid RPC システムを用いて応用プログラムを実行することを考えた場合，上記のコンポーネントの他にもスケジューラやモニタなど様々なコンポーネントが必要となる．しかし，それらのコンポーネントはまだ研究段階にあるものが多く，実行環境や応用プログラムの性質に応じて最適な戦略 / アルゴリズムが変わるものが多い．そこで Ninf-G では，まず Grid RPC のコアとなる基本コンポーネントを実装，提供し，その他のコンポーネントは基本コンポーネントに対してプラグイン的に追加できるような設計とする．このように設計，実装することにより， Grid RPC の基本コンポーネントとして世界中で広く利用される事を目指す．

このような設計方針に基づいて， Ninf-G は米国で開発された Globus Toolkit を用いて実装することにした．詳細は次節で述べるが， Globus Toolkit はグリッドソフトウェアに必要とされる資源管理機構やユーザ認証システム，通信ライブラリなどの要素技術を実装するための API やコマンドを提供する低レベルなツールキットであり，グリッドのソフトウェアインフラストラクチャを構成する要素の事実上の標準になりつつある．Ninf-G の実装に Globus Toolkit を利用することにより，セキュリティ (ユーザ認証) や資源管理など様々な要素を Globus の機能を用いて実装する (Globus に任せてしまう) ことができる．これにより，より豊富な機能をより容易に実装することができるようになる．また，Globus Toolkit を用いて実装されている他のグリッドソフトウェアと類似したインタフェースを提供することができ，将来的に Globus Toolkit の付属パッケージとして Ninf-G を配布するという戦略も考えられる．

3. Globus Toolkit

Globus は米国の大規模なグリッドプロジェクトであり，グローバルコンピューティングのためのサービスの集まりである Globus Toolkit を提供している．Globus Toolkit は 1998 年 10 月にバージョン 1.0 がリリースされ， 2001 年 6 月の時点での最新バージョンは 1.1.4 である． Globus が提供するサービスを**表 1** に示す．Globus が提供するこれらのサービスは必要に応じて個別に利用することができるようになっており，既存アプリケーションへのインクリメンタルな導入が可能である．Globus は階層的な構造を持ち，高レベルの Globus サービスはローカルサービスの上に構築され，ローカルサービスの種類に依存しない均一なインタフェースを提供している．ここでは，今回の実装に際して特に重要である 3 つのサービス (Security， Resource Management, Information Infrastructure) および Globus における遠隔ジョブ実行の仕組みについて詳しく説明する．

3.1 Security

ユーザ認証や安全な (secure な) 通信を実現するため，Globus は **Grid Security Infrastructure (GSI)** を提供している． GSI は公開鍵による暗号化， X.509 証明書および Secure Socket Layer (SSL) 通信プロトコルに基づいて実装され，これに single sign-on(パスワードを 1 度だけ入力すれば良い) および委任 (delegation) の拡張がなされている．ユーザは自分の証明書を認証局に発行してもらう必要がある．サーバ上にはユーザの Globus ID(証明書に記載されている) とローカルユーザ ID(Unix のアカウントなど) との対応表があり，この表に従ったローカルユーザの権限でジョブが実行される．

3.2 Resource Management

Globus Resouce Allocation Manager (GRAM) は計算資源 (プロセッサ) 管理のためのサービスを提供する． GRAM は LSF， NQS， Condor など様々な (サイト固有の) 資源管理ツールに対する共通のインタフェースを提供する．GRAM のアーキテクチャにおいては， gatekeeper と jobmanager が主要なコンポーネントである． gatekeeper はクライアントからのジョブ要求を待ち受けるサーバであり， jobmanager は gatekeeper によって生成され，実際の資源管理方法に従ってジョブを生成するプロセスである． jobmanager はローカルな資源管理ツールに応じた型を持ち，例えば LSF 型の jobmanager の場合は LSF の *bsub* コマンドを使ってジョブをキューに投入し， LSF のスケジューリングに応じてホスト上でジョブが実行される．また，これらの資源要求に関する情報は RSL (Resource Specification Language) と呼ばれる言語を用いて記述され，各コンポーネント間で交換される．

3.3 Information Infrastructure

Globus は **Metacomputing Directory Service (MDS)** と呼ばれる LDAP(Light-weight Directory Access Protocol) を用いた情報サービスシステムを提供している． Globus の MDS は非集中管理方式

表 1　Globus が提供するサービス

Service	Name	Description
Security	GSI	authentication などのセキュリティサービス
Information Infrastructure	MDS/GRIS/GIIS	情報サービス
Resource Management	GRAM/RSL/DUROC	資源管理および資源要求等のサービス
Data Management	GASS/GSIFTP	リモートデータへのアクセスサービス
Communication	Nexus/Globus IO	通信サービス
Fault Detection	HBM	システムの状態および障害検知サービス
Portability		高い移植性を提供 (libc, pthread library など)

(pull モデル) であり☆，情報の問い合わせや情報を格納する名前空間の構築のために LDAP を用いている．MDS の主要なコンポーネントは **Grid Resource Information Service(GRIS)** と **Grid Index Information Service(GIIS)** である．GRIS は計算サーバ上で動作する LDAP のサーバであり，GRIS に問い合わせることによってその計算資源に関する情報を獲得することができる．GRIS が提供する情報はデフォルトでは globus のバージョン，計算機のハードウェア / ソフトウェア情報および gatekeeper/jobmanager に関する情報であるが，必要に応じて GRIS が提供する情報を追加 / 削除する事ができる．GRIS が単体の計算サーバの情報を扱うのに対し，GIIS は複数の GRIS の情報をまとめて扱う仕組みを提供し，それによって「複数のサイトにまたがった仮想的な組織」の情報を提供する MDS サーバとして機能する．

3.4 Globus におけるジョブ実行の仕組み

Globus を用いて遠隔資源上でジョブを実行する場合，以下のような流れになる．

(1) クライアントからユーザコマンドあるいは API を利用して gatekeeper にジョブ実行の要求を出す．実行すべきプログラムは RSL に従った記法でクライアントが指定する．
(2) gatekeeper がクライアントからの要求を受け取ると，クライアントと gatekeeper との間で相互認証が行なわれる．
(3) gatekeeper は適切な jobmanager を生成する．
(4) 生成された jobmanager はクライアントによって指定されたプログラムを実行するジョブプロセスを生成する．
(5) ジョブ実行の成功 / 失敗 / 終了 / 結果の通知やジョブの取り消し要求などのやりとりは，クライアントと jobmanager の間で行なわれる．ジョブプロセスの出力等は GASS を介してクライアント側の出力に送られる．

このように，Globus はクライアント / サーバモデルに基づく単純な実行モデルを提供すると同時に，ジョブプロセスの生成や状態確認，実行の取り消しといった資源管理システム固有の機能を jobmanager に任せることにより，各サイトのスケジューリングポリシーに依存しない柔軟な資源管理機構を実現している．

☆ バージョン 1.1.2 までは集中管理方式 (push モデル) であった

4. Ninf-G の実装

4.1 実装の概要

Ninf-G は実装中であり，本節では現在実装を終えている部分に関して説明する．今後の予定等については 6節で述べる．Ninf-G の実装の基本的なアイデアを以下に示す．

- リモートライブラリの呼び出しには，Globus の GRAM を利用する．つまり，Globus の gatekeeper をサーバとして利用し，クライアントは Globus の gatekeeper に対してジョブ要求を送る．リモートライブラリの実行中に障害が発生した場合は GRAM が検知する．
- リモートライブラリのパスの保持，検索に GRIS を用いる．具体的には，リモートライブラリの作成 (登録) 時に GRIS にパスの情報を登録する．
- 引数情報の保持，検索も GRIS を用いて行なう．具体的には，リモートライブラリ作成 (登録) 時に GRIS に引数情報を登録する．

Ninf-G では各コンポーネントおよびコンポーネント間のプロトコルがどのように実装されているかについて，以前我々が作成した NES システムである Ninf[8] の実装方法と比較してまとめたものを**表 2** に示す．

Ninf-G は Ninf をベースに実装されており，例えばリモートライブラリの構築に必要となる IDL(Interface Description Language) ファイルのコンパイラの主要部分等，使える部分はそのまま利用している．以下では，Globus Toolkit を用いて実装した Ninf-G のコンポーネント / 機能に関して，Ninf に対してどのような修正を加えたかという点に焦点を当てて説明する．

4.2 リモートライブラリの構築および登録

Ninf では，既存のライブラリを Ninf サーバ上でリモートライブラリとして作成する場合，

(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) Ninf サーバを動かし，作成されたリモートライ

表 2　Ninf-G と NS の実装方法

	Ninf-G	NS
サーバ	Globus の gatekeeper	独自のサーバ
ジョブ要求	Globus の API を使って gatekeeper に投げる	独自のプロトコルによりクライアントがサーバに要求
リモートライブラリのパス	GRIS に登録	サーバに登録
引数情報	GRIS に登録	サーバに登録

ブラリを登録する．

といった手順を踏む．Ninf サーバにリモートライブラリを登録することにより，リモートライブラリのパスは Ninf サーバが保持する事ができる．Ninf では Ninf サーバがクライアントからの要求を受け取ると，指定されたライブラリ名をキーとして Ninf サーバ自身が保持しているリモートライブラリのパスを検出し，実行する．そのため，クライアントが Ninf サーバにリモートライブラリの実行要求を出す場合，ライブラリ名だけを指定すれば良い．しかし，GRAM の API は仕様としてサーバ上で実行するプログラムのパスを指定する必要がある．Ninf-G では Globus の GRAM を用いてジョブ要求を行なっているため，クライアントがリモートライブラリのパスを知る必要が生じる．そのため，Ninf-G ではリモートライブラリのパスを GRIS に登録し，クライアントはライブラリ名をキーとして GRIS を検索し，リモートライブラリのパスを取得するという実装となっている．

4.3　引数情報の取得

Ninf では，リモートライブラリのパスと同様に，引数や返り値の情報 (引数情報) も Ninf サーバに登録されている．クライアントは Ninf サーバに問い合わせることによりリモートライブラリの引数情報を取得し，その情報に従って引数をサーバに送り，計算結果をサーバから受け取る．Ninf-G ではリモートライブラリのパスと同様に引数情報も GRIS に登録し，クライアントはライブラリ名を鍵として GRIS を検索し，引数情報を取得するという実装となっている．

4.4　LDIF ファイルの作成

Ninf-G の IDL コンパイラは Ninf と同様にスタブソースファイルと Makefile を生成するが，Ninf-G ではその他に引数情報を生成するためのソースファイル (引数情報ソースファイル) を生成する．生成された Makefile 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他に，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(LDAP Data Interchange Format) 形式のファイルを生成する．Ninf-G では，ホストの情報に関する LDIF ファイルと，リモートライブラリの各エントリ (関数) ごとの LDIF ファイルを生成する．LDIF は LDAP のエントリを簡単なテキスト (ASCII) 形式で表現するためフォーマットであり，*ldapadd* や *ldapsearch* などの LDAP ツールの入出力に利用される．LDIF エントリの基本的な形式を**図 1** に示す．属性値がテキストで表現できない場合やサイズが大

```
dn: <識別名>
<属性記述子>:  <属性値>
<属性記述子>:: <base64 でエンコードした属性値>
...
```

図 1　LDIF ファイルのフォーマット

きすぎるような場合には，base64 encoding を行なう．**図 2** にクライアントとサーバ間でデータを pingpong 転送するリモートライブラリの IDL ファイルを示す．また，この IDL ファイルをもとに，前述の手順に従って生成されたホストの情報に関する LDIF ファイルを**図 3** に，リモートライブラリの情報に関する LDIF ファイルを**図 4** に示す．　これらの LDIF ファイルは make を

```
Module perf;

Define pingpong(mode_in int n,
                mode_in int A[n],
                mode_out int B[n],
                mode_out double usec[1])
"n Bytes ping-pong transfer for
 performance evaluation."
Required "pingpong.o"
Calls "C" pingpong(n, A, B, usec);
```

図 2　pingpong ライブラリの IDL ファイル

```
dn: sw=GridRPC, hn=ninf.aist.go.jp, dc=gci,
 dc=jp, o=Grid
objectclass: GlobusSoftware
hn: ninf.aist.go.jp
cn: GridRPC
```

図 3　ホスト情報に関する LDIF ファイル

実行したディレクトリに生成され，`make install` コマンドを実行することにより，Globus が置かれているディレクトリ (deploy ディレクトリ) のサブディレクトリ (`${DEPLOY_DIR}/var/gridrpc/`) にコピーされる．

4.5　GRIS への登録

作成された LDIF は Ninf-G のリモートライブラリおよびサーバ計算機の情報を提供する．GRIS に対して情報を提供する情報プロバイダは，標準出力に LDIF オブジェクトを生成するインタフェースを備え

```
dn: rpcFuncname=perf/pingpong, sw=GridRPC, hn=ninf.aist.go.jp, dc=gci, dc=jp, o=Grid
objectclass: GridRPCEntry
hn: ninf.aist.go.jp
rpcFuncname: perf/pingpong
module: perf
entry: pingpong
path: /home/ninf/tests/pingpong/_stub_pingpong
stub:: PGZ1bmNOaW9uICB2ZXJzaW9uPSIyMjEuMDAwMDAwIiA+PGZ1bmNOaW9uX25hbWUgbW9kdWxl
 PSJwZXJmIiBlbnRyeTOicGluZ3BvbmciIC8+IDxhcmcgZGFOYV90eXBlPSJpbnQiIG1vZGVf
 ...（以下省略）
```

図 4 pingpong ライブラリの LDIF ファイル

ている必要があり，Ninf-G の場合は生成された LDIF ファイルを cat コマンドなどによって標準出力に生成する事により，リモートライブラリの情報を提供する情報プロバイダを作成することができる．GRIS がこれらの情報プロバイダからの情報を取得するようにするためには，GRIS の設定ファイルを修正すれば良い．GRIS を再起動する必要はない．GRIS の設定ファイルは `grid-info-resource.conf` である．**図 5** に `grid-info-resource.conf` ファイルの例を示す．この設定ファイルは 1 行に 1 つの情報プロバイダのエントリが記述されるフォーマットであり，最初の 3 つのエントリはデフォルトの情報プロバイダである．最後の行が Ninf-G が生成した LDIF ファイルを参照するためのエントリであり，

このように GRIS の設定ファイルを記述することにより，GRIS は `${DEPLOY_DIR}/var/gridrpc/` ディレクトリにあるすべての LDIF ファイルに書かれている情報を提供することができる．

4.6 クライアント API(*Ninf_Call* など)

Ninf はクライアントがリモートライブラリを呼び出すための API として，*Ninf_Call()* やその非同期版である *Ninf_Call_Async()* などを提供している．これらの関数の第一引数はリモートライブラリの識別子 (モジュール名とエントリ名により構成される) であり，この識別子を利用してサーバ側で実際のパスや引数情報を検索する．Ninf-G でもそれらの API の呼び出しインタフェースには変更がない．

しかし，Ninf ではクライアントから Ninf サーバに対して *connect()* をかけて TCP ストリームを確立し，その後サーバからスタブの情報を獲得したり，実際に (サーバを介して) リモートライブラリとの間で引数や返り値の送受信を行なっていたのに対し，Ninf-G では Globus の API を利用して gatekeeper にジョブを投げる時点で，リモートライブラリのパスを知っておく必要がある．そのため，クライアントは gatekeeper にジョブを投げる前に，第一引数で指定されたリモートライブラリの識別子を鍵としてそのリモートライブラリに関する情報を GRIS に問い合わせ，リモートライブラリのパスを取得する．また，同様に引数情報も最初に取得してしまう．これらの取得した情報はクライアント側で保存しておき，次回以降同じリモートライブラリを呼び出す場合には GRIS に問い合わせずにローカルな検索で済むような実装にした．

5. Grid RPC システムの性能

前説で示したサーバとの間でデータの pingpong 転送を行なうリモートライブラリを用いて，Ninf-G におけるジョブ実行の性能を測定した．測定項目は以下の 4 項目である．

- **GRIS 検索時間**
  クライアントが GRIS にリモートライブラリの情報を問い合わせる時間．実際にはホストの Distinguished Name(Host DN) の検索，Host DN を base とし，ライブラリの識別子をキーとしたリモートライブラリのパスの検索，Host DN を base とし，リモートライブラリの識別子をキーとした引数情報の検索の 3 回の LDAP のライブラリ関数 (*ldap_search_s()*) を用いた検索が行なわれる．
- **GRAM 呼び出し時間**
  クライアントがサーバである gatekeeper にジョブの要求を送信してから，実際にリモートライブラリが実行されてリモートライブラリから引数の送信要求が届くまでの時間．
- **upload 時間**
  クライアントからリモートライブラリにデータを転送する時間．
- **download 時間**
  リモートライブラリからクライアントにデータを転送する時間．

データサイズを 256byte から 64MB まで 4 倍ずつ変化させて上記 4 項目を測定した．クライアントとサーバの両方とも産総研のマシン (LAN 接続されている) で動かした場合と，クライアントは東工大で，サーバは産総研で動かした場合の二通りの環境で実験を行なった．東工大から産総研へのルートは WIDE ～ IMNet 経由で，ftp によるファイルの転送スループットは約 200KB/sec である．産総研から東工大へのルートは IMNet ～ sinet 経由で，ftp によるファイルの転送スループットは約 600KB/sec である．クライアントを産総研で動作させた場合の結果を**図 6** に，東工大で動作させた場合の結果を**図 7** に示す．グラフには，それぞれ 4 回ずつ計測したうちの最小値を示している．図 7 の Y 軸は log scale になっている． LAN 環境では GRIS の検索時間は約 5

```
# Format:  TTL cmd [arg...]
600 ${bindir}/globus-version -ldif
600 ${bindir}/grid-info-host
30 ${libexecdir}/globus-gram-scheduler  -dmdn "${GRID_INFO_ORGANIZATION_DN}"\
  -conf ${sysconfdir}/globus-jobmanager.conf -type fork -rdn jobmanager \
  -machine-type unknown
0 cat ${localstatedir}/gridrpc/*.ldif
```

図 5　grid-info-resource.conf ファイルの例

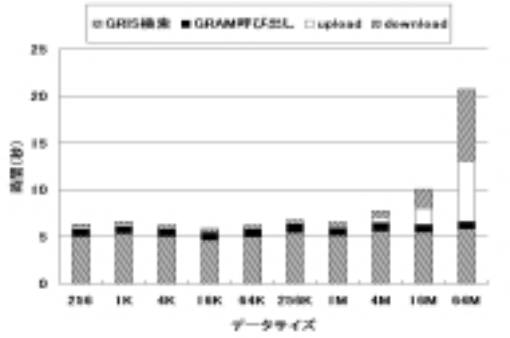

図 6　pingpong 転送の実行時間 (LAN 環境)

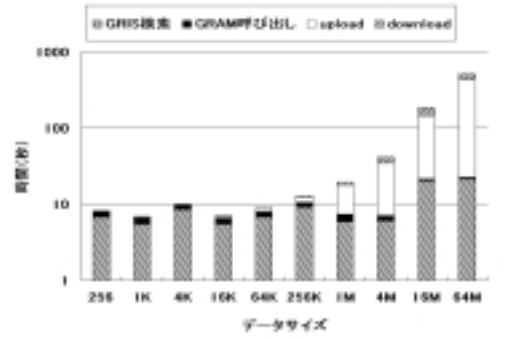

図 7　pingpong 転送の実行時間 (WAN 環境)

秒，GRAM 呼び出しは 1 秒弱程度であり，その時間にほとんど揺れはなかった．WAN 環境での GRIS の検索時間は平均的には 6 秒程度であったが，その揺らぎが大きく，時には 20 秒程度かかることもあった．GRAM 呼び出しは平均的には 1 秒強程度であったが，やはり 4 秒から 6 秒程度かかることもあった．これは WAN 環境における通信性能の変動によるものと考えられる．GRIS は最初に一度検索されるだけであり，この処理をプログラムの初期化時に行なえば，リモートライブラリ呼び出し時のオーバーヘッドとはならない．GRAM の呼び出しに 1 秒程度かかることがリモートライブラリ呼び出し時のオーバーヘッドとして無視できるかどうかは，データの転送時間やリモートライブラリの実行時間との兼ね合いによる．しかし，グリッドで動作させるような応用プログラムの粒度は比較的大きいことが想定される．また，パラメータサーチのような比較的粒度の小さな計算を多数行なうような場合には，GRAM 呼び出しは一度だけ行ない，以降は起動されたリモートライブラリとクライアントとの間で接続を保持して計算を進めるという機能を提供することにより，GRAM 呼び出しのオーバーヘッドを相対的に抑えることが可能となる．

6. まとめと今後の展望

本稿では Globus Toolkit を用いて実装した Grid RPC システムである Ninf-G について，その設計方針，実装方法および基本性能を報告した．Ninf-G は実装中であり，今後下記のような開発を進める予定である．

- GRAM を用いたエラーハンドリングを実装する．
- Globus IO を用いてリモートライブラリからクライアントに計算結果を通知するように修正する．
- クライアントとリモートライブラリとの接続を保持するプロトコルを実装する．
- クラスタを利用するための GRAM(jobmanager) を作成する．
- 階層化の確認や Condor の Match Maker など他のシステムとの適合性などに関して設計を確認する．

謝辞　本研究に際し，日頃議論に参加して頂いている Ninf チームの皆様に感謝致します．

参 考 文 献

1) Foster, I. et.al.(eds.): *The GRID: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann (1999).
2) Matsuoka, S., et.al.: Design issues of Network Enabled Server Systems for the Grid. http://www.eece.unm.edu/~dbader/grid/.
3) Nakada, H., et.al.: Design and implementations of Ninf: towards a global computing infrastructure, *FGCS*, Vol. 15, pp. 649–658 (1999).
4) Casanova, H. et.al.: NetSolve: A Network Server for Solving Computational Science Problems, *Supercomputer Applications and High Performance Computing*, Vol. 11, No. 3, pp. 212–223 (1997).
5) Abramson, D., et.al.: High Performance Modeling with Nimrod/G: Killer Application for the Global Grid?, *IPDPS*, pp. 520–528 (2000).
6) http://www.gridforum.org/.
7) http://www.globus.org.
8) 中田秀基 他: Network Enabled Server System の設計, HPC 研究会報告, Vol. 2000, No. 57, pp. 69–74 (2000).