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

2005－ARC－162 （16）
2005－HPC－101 （16）
2005／3／8

GridRPC システムの比較 — アプリケーション開発における違い

谷村勇輔[†]　中田秀基[†,††]
田中良夫[†]　関口智嗣[†]

本研究では，代表的な GridRPC システムである Ninf-G，NetSolve および OmniRPC に注目し，それぞれのシステムを使ってアプリケーションを開発する上での違いを調査した．システムが想定している用途の違いを踏まえて，プログラミングや遠隔実行プログラムの起動方法，利用できる環境，実行をサポートする機能の違いについて明らかにし，単一 RPC 実行時の性能比較を行った．また，将来的にサポートが進むであろう機能や発展的な RPC についても言及した．こうした調査結果は，どのシステムを利用してアプリケーションを開発するかを決める上で有用な情報になると考える．

Comparison of The GridRPC System — Difference in Application Development

Yusuke Tanimura,[†] Hidemoto Nakada,[†,††] Yoshio Tanaka[†]
and Satoshi Sekiguchi[†]

In this research, Ninf-G, NetSolve and OmniRPC which are three of the famous GridRPC systems are compared in the view of application development and its execution. Based on each target of those systems, programming, invokation of a remote program, supported environment, and useful functions in execution are revealed informatively. Single RPC performance of each systems is also evaluated and then advanced RPC is introduced in this survey. These information is useful for application developers in choosing their middleware.

1. はじめに

GridRPC[1] システムは，RPC を広域ネットワーク環境に拡張して分散計算を可能にするシステムを目指して，1994 年頃より開発が始まり，初期のシステムとしては Ninf[2] や NetSolve[3] が有名である．そして，実際のグリッド環境においてアプリケーションを動かす段階になっている現在，これらは Ninf の後継である Ninf-G[4] や Ninf から派生した OmniRPC[5]，グリッド環境を意識して拡張が続けられ，GridSolve とも呼ばれる NetSolve（Version 2）として発展した．

いくつかの実装が存在する一方で，アプリケーション・プログラマが一様な形式で遠隔資源に効率的にアクセスして，分散計算を行うコードを書くことができるよう，GGF（Global Grid Forum）[6] において API 仕様の標準化が進められている[1]．Ninf-G と NetSolve は既に GridRPC API を実装し，標準化の恩恵を受けることができるようになっている．しかし，それでもなお，各 GridRPC システムが想定する利用シナリオや初期のシステムアーキテクチャの違いから，システム内部の実装や挙動が異なっているため，アプリケーションの開発や実行に違いが残っている．また，アプリケーションによっては，下位のシステムに依存して，開発の手間や実現性に違いが生じる可能性もある．

そこで，本稿では，アプリケーション・ユーザの観点に立ち，Ninf-G，NetSolve，OmniRPC の 3 つのシステムの機能や性能を比較する．特に，各システムにおいて，何がアプリケーションから可能であり，難しいかを明らかにすることで，アプリケーション開発者がその下位に用いるシステムを選ぶ際に参考にできる情報を示す．同時に，今後の GridRPC システムにおいて，必要とされる機能や改善されるべき機能について検討を行う．

2. GridRPC システム

GridRPC は，RPC（Remote Procedure Call）の仕組みをグリッドに拡張し，グリッド・アプリケーションのための代表的なプログラミングモデルを提供する枠組である．GridRPC では，アプリケーション・プログラムの中

† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†† 東京工業大学
Tokyo Institute of Technology

で RPC を発行する API を記述して，遠隔の計算機を利用して計算の一部を行うことができる．RPC を非同期に発行することで，複数の計算機を用いてタスクを並列実行することもできる．

本研究では，それぞれの最新版となる下記のバージョンにおいて，実装されている機能やその性能を比較する．

- Ninf-G： バージョン 2.3.0（2004 年 12 月 21 日リリース）

Ninf-G は，Ninf の後継として産総研や東工大らによって開発され，当初のクライアント・サーバ型の遠隔プログラムの実行システムから，細粒度のタスクを大規模に並列実行するためのミドルウェアとして再設計されている．その際に，アーキテクチャの異なる計算機において互換性が保証されたミドルウェアの開発を可能にし，グリッドの標準的な下位ミドルウェアとして注目されていた Globus Toolit（以下，Globus）[7] を利用して，多くのグリッドの試験環境で利用可能なシステムとなっている．そのシステムアーキテクチャは，**図 1** に示すように，遠隔に用意されたプログラムの情報を得るための情報サービスとして MDS，遠隔プログラムを起動するために GRAM，遠隔で実行するプログラムやデータファイルの転送に GASS を使っている．そして，クライアントとサーバは Globus I/O を利用して通信を行う．

- NetSolve/GridSolve： バージョン 2.0（2003 年 10 月 10 日リリース）

NetSolve は，テネシー大学により開発されているシステムであり，高性能な計算機に用意された優れた線形数値計算ライブラリを使って，科学計算を分散処理できるよう設計されている．また，科学計算を個人 PC から簡単に試せるように Fortran や C だけでなく，Matlab や Mathematica, Octave を使って，対話的に遠隔のプログラムを呼び出すことも可能である．**図 2** に示すように，そのアーキテクチャは NetSolve Server，NetSolve Agent，NetSolve Client の 3 要素から構成される．Agent は動作中の Server を管理し，Client が要求するサービスを起動できる Server を斡旋する役割を持つ．特に，複数の Client からの RPC 要求に対するスケジューリングや，障害により失敗した RPC を別の Server に再び割り当てる機能を持つ．

- OmniRPC： バージョン 1.0（2003 年 10 月 14 日リリース）

OmniRPC は筑波大により開発されており，個人が分散計算をより簡単に，手間なく行えることを目指したシステムである．Globus がインストールされていない計算機でも広く利用可能なように，Globus だけでなく RSH や SSH を使って遠隔プログラムを起動できる．他の 2 システムと比べて，遠隔プログラムの情報は個々の計算機に保存され，それらをまとめた情報サービスをもたないアーキテクチャである．提供される API は GridRPC API に準拠しておらず，ホストを指定せずに RPC を実行することができ，その場合は，利用可能なホストが順に割り当てられる機能が提供されている．

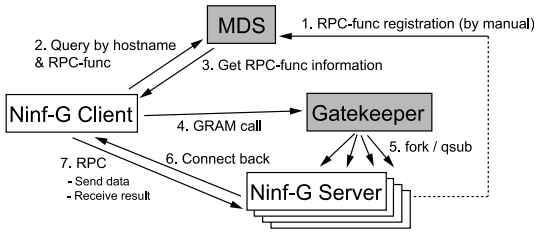

図 1 Ninf-G のシステムアーキテクチャ

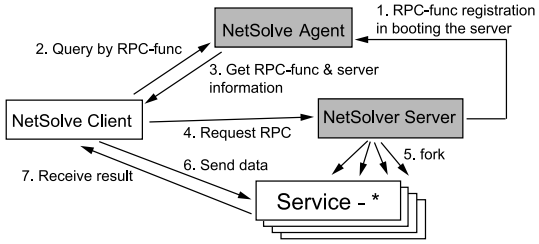

図 2 NetSolve のシステムアーキテクチャ

3. プログラミング

3.1 クライアントプログラムの開発

GridRPC のアプリケーションでは，クライアントと遠隔実行プログラムを別々に開発することになる．そのうち，クライアントから遠隔プログラムに実装された関数を呼び出す API について，標準化が進められている．予定としては，エンドユーザ向けの API が 2005 年 2 月より 60 日間のパブリックコメントの受付に入り，議論が交わされた後，標準として正式に策定される．各システムの標準 API への対応具合は異なり，OmniRPC は類似の API を提供しているものの対応はしていない．NetSolve も NetSolve 固有の API が存在し，GridRPC API はそれをラッピングしたモジュールとして提供され，エラーコードなど一部は未対応である．Ninf-G は全てに対応しているため，NetSolve と Ninf-G 間でプログラムの移植を行う場合は若干の修正が必要である．

クライアントでは，**図 3** に示す GridRPC API を用いて，遠隔の関数を同期または非同期に実行する．同期呼び出しを行う grpc_call() は，図 3 のように「引数データの転送」,「リモートでの計算」,「結果データの転送」のフェーズが全て終了するまで関数が返らない．それに対して，非同期呼び出しを行う grpc_call_async() では「引数データの転送」の前後で関数が返り，後に grpc_wait() などの待機関数を用いて計算の完了を待つことになる．この

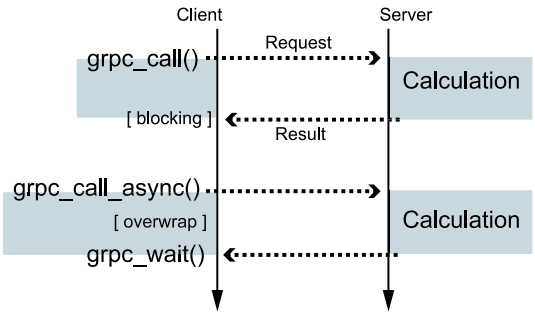

図 3 GridRPC の主要関数

時，Ninf-G は非同期呼び出しの関数から返るタイミングを wait，nowait，copy のいずれかで指定できる．wait では入力データがサーバに渡ってから関数が返るが，nowait ではデータの転送を待たずに関数から返る．copy はデータの転送を待たないが，クライアント内でデータが複製された後に関数から返る仕様となっている．いずれを指定するかは，アプリケーションの計算手順として，非同期呼び出しの際に指定したデータを呼び出した関数の計算が完了するまでに修正するかやデータサイズに依存する．これに関して，NetSolve は wait に相当する RPC のみ，OmniRPC は copy に相当する RPC のみをサポートしている．

GridRPC API では，RPC を実行する前に，遠隔プログラムを抽象化した「関数ハンドル」を作成する必要がある．これは，特定のサーバで動作するプログラムに対応し，Ninf-G では関数ハンドルの作成時に遠隔プログラムが起動される．それに対して，NetSolve では RPC の実行時に遠隔プログラムが起動され，両システムで内部実装に違いがある．このため，エラーコードを単純に統一することは難しく，異なる GridRPC API において RPC の実行制限のエラーが返る現状である．OmniRPC では，関数ハンドルを作成する場合は Ninf-G と同じ挙動となるが，関数ハンドルを作らずに，RPC の実行先の選択をシステムに任せることもできる．

3.2 遠隔実行プログラムの開発

GridRPC を実行するサーバ側，すなわち，遠隔プログラムの開発においては，サービスとして提供する関数のインタフェースの記述に用いる IDL（Interface Definition Language）の文法が最も大きな違いとなる．Ninf-G と OmniRPC は書式が似ているが，互換性が保証されているわけではない．IDL を元にしてクライアント向けに生成される情報ファイルやクライアントとサーバ間のプロトコルが異なるため，システム間でサーバを相互に利用することは困難である．NetSolve では，IDL よりさらに低レベルの PDF（Problem Description File）を最終的に用意しなければならない．IDL ファイルから PDF を生成するための idltopdf は洗練されておらず，複雑なインタフェースを記述する場合にはユーザが直接編集を行わなければならない． あるいは，NetSolve の Web ページにおいて公開されている PDF Wizard☆を利用して対話的に PDF を作成することになる．

4. 遠隔実行プログラムの起動とセキュリティ

Ninf-G では，図 1 に示したように，関数ハンドルの初期化時に Globus の Gatekeeper を通して遠隔プログラムが起動される．その際，GSI に基づいた認証がなされ，許可されたユーザの権限でプログラムが実行される．また，クライアントと遠隔プログラム間のデータ転送において，転送データの改ざんをチェックしたり，暗号化を行ったりすることができる．ただし，Globus を利用するために，ユーザは利用する計算機が信頼する CA からユーザ証明書を取得し，その DN を grid-mapfile へ登録してもらう作業が最初に必要である．

OmniRPC も Globus を利用するが，Ninf-G とは異なり，Gatekeeper によって omnirpc-agent が起動される．これは，OmniRPC システムが初期化される際に実行され，マシンファイルに記された全ての計算機で omnirpc-agent が動くことになる．そして，RPC 要求が出された時点で，omnirpc-agent から fork や ssh を利用して遠隔プログラムが起動される．遠隔プログラムは，omnirpc-agent と同じユーザの権限で動作するが，クライアントとの通信を暗号化することはできない．一方，Globus を利用するのに必要な準備を回避するため，SSH や RSH を使って omnirpc-agent を起動する方法も提供されている．通信の暗号化が必要な場合など，omnirpc-agent を SSH で起動し，SSH のポートフォワーディングを利用して，クライアントと遠隔プログラムの通信を omnirpc-agent に中継させる使い方も考えられる．

NetSolve では，遠隔の計算機において常に NetSolve Server がデーモンとして動作していることを仮定し，RPC 要求があった時に，その Server から Server と同じ権限で遠隔プログラムが起動される．NetSolve では，１つの Server が複数のユーザの遠隔プログラムをサービスするため，計算機の管理者が NetSolve の専用アカウントを用意して，遠隔プログラムを登録することが望ましい．セキュリティ機構については，RPC の実行を一部のユーザに制限するために Kerberos V を利用できるほか，後述する遠隔プログラムの配布機能において，プログラムの実行を GNU PG（Privacy Guard）で署名されたものに限ることができる．これらを利用する場合も，ユーザは鍵を KDC に登録したり，自身の GPG 署名を NetSolve の Server に登録してもらう作業が必要である．NetSolve では，通信路の

☆ http://www.cs.utk.edu/ seymour/PdfWizard/pdfwiz.html

暗号化は提供されておらず，また固定ポートで常に Agent や Server が動き続けることの危険性がある．

5. 実行環境による制約

まず，タスク並列のアプリケーションを大規模に実行する場合，複数のクラスタ型の並列計算機を利用することが考えられる．しかし，現実に多くのクラスタでは，バックエンドの計算ノードはプライベートな IP アドレスが振られていたり，ローカルなジョブスケジューラを通して遠隔プログラムを実行することが求められたりする．そこで Ninf-G や OmniRPC では，遠隔プログラムからクライアントに対して TCP の接続がなされ，NAT 機能が利用できれば，そうしたバックエンドの計算機を使えるよう設計されている．OmniRPC では，NAT が使えない場合でもフロントエンドで動く omnirpc-agent からクライアントに対して TCP の接続がなされ，バックエンドで動作する遠隔プログラムの通信を中継する機能も提供されている．これらに対して，NetSolve では，クライアントから遠隔プログラムに対して TCP の接続がなされるため，プライベートなネットワークにあるバックエンドを利用できない．

次に，NetSolve は Matlab や Mathematica，Octave のコマンドラインから対話的に RPC を実行する機能を持つ．特に，Matlab では，Microsoft Windows 上からもそれが可能である．また，Windows においてクライアントを実行するために，Windows 2000 および XP 用に C のライブラリを提供している．それに対して，Ninf-G や OmniRPC では対話的に RPC を実行する環境は提供していない．Windows におけるクライアントの実行についても，Ninf-G は Globus に依存しているため Java のインタフェースに限られ，OmniRPC ではサポートされていない．

6. アプリケーションの実行

6.1 遠隔実行プログラムの配布

GridRPC システムは，元々はクライアントサーバ的な使い方が主流であったため，あらかじめ，遠隔プログラムは実行される計算機においてコンパイルされ，セットアップされることが想定されてきた．現在もそうした使い方は少なくないが，Ninf-G のように大規模に計算を行う場合，呼び出すプログラムを更新する毎に，それを全ノードに配布してコンパイルして回ることは大変な労力である．これを解決するために，Ninf-G では，クライアントに存在する実行ファイルを GASS を使って遠隔に転送する staging 機能を提供している．計算機のアーキテクチャが揃っており，特別なライブラリを利用しないプログラムであれば，遠隔ホストにて事前にプログラムをコンパイルするなどの作業が不要となる．

一方，NetSolve は，ある人が作成した優れたプログラムを別の人が提供する高速な計算機で実行したいという要望に応えて，ハードウェア・ソフトウェアサーバ機能[8] を実装している．これは，Server が提供するサービスをソフトウェアサーバと，ソフトウェアサーバによって提供されたプログラムを動かす計算資源を提供するハードウェアサーバに分離し，クライアントからだけでなく，実行ファイル群を提供するソフトウェアサーバから必要な実行ファイルを転送することを可能にする．利用に際して，NetSolve の管理者は，これまでの固定的に提供するサービスだけでなく，ソフトウェアとハードウェアのいずれか，あるいは両方のサービスを提供するかを選択することになる．

OmniRPC には遠隔プログラムの配布機能はない．

6.2 情報サービスとスケジューリング

グリッド環境における情報サービスは，GridRPC に限らず多くの議論が続いている．Ninf-G では，Globus に含まれる汎用的な情報サービスの機構である MDS を使い，遠隔プログラムの関数名や引数の型，実行ファイルのパスなどの情報を LDIF 形式で登録することができる．Ninf-G のクライアントは，関数ハンドルの作成時にホスト名と RPC 関数名の組合せで MDS の検索を行い，これを情報を取得する．ただし，自動的に負荷の低いサーバを選ぶなどのスケジューリング機能は提供されていない．これは，Ninf-G が参照する GridRPC が基本的な RPC の API だけを規定しているためである．GridRPC では，スケジューリングや障害に関する機能は，その上位に作られるミドルウェアやアプリケーションのレベルで個々に実装されることが想定されている．

一方，NetSolve は，常に NetSolve Agent を介して RPC を実行するため，全 Server の情報は 1 つの Agent に集められる．また，Server は起動時に LAPACK のベンチマークを実行して，計算性能と実行可能なサービスの情報を Agent に通知し，以降も定期的に負荷情報を Agent に通知する．そして，Agent は，Client の検索に対して性能が高く，負荷の低いサーバを紹介するスケジューリング機構を提供する．スケジューリングの精度を向上させるために，NWS（Network Weather System）[9] が収集した情報を取り込むこともできる．障害により RPC が失敗した場合には，Agent は自動的に別の Server を検索し，失敗 RPC を 3 回まで再実行する機能も提供されている．この時，RPC を失敗した Server は，一度 Agent から削除される．

OmniRPC は全体を管理する情報サービスをもたず，遠隔プログラムが展開された計算機に，そのプログラム情報が XML 形式で記述されたファイルが置かれている．OmniRPC システムが初期化される際に起動された omnirpc-agent がその情報を読み込み，クライアントに対してサービスを提供する．また，OmniRPC ではホストを指定せずに RPC を実行することができ，その場合は利用可能なホ

ストをラウンドロビンで割り当てるスケジューリング機構を提供している．

7. 性能評価

本章では，システムを選択する上での 1 つの指標として各システムの性能を比較した結果を示す．評価実験には**表 1** の計算機を用いて，常に AIST のノードをサーバとし，LAN での実験では AIST のノード，Internet 越しの実験では SDSC のノードでクライアントを実行した．

7.1 単一 RPC 実行時のデータ転送

まず，遠隔にデータを転送して計算を行うことで生じるオーバーヘッドを測定した．クライアントからのデータを受信後に 5 秒間スリープし，同じサイズのデータをクライアントに送信する関数を実装した遠隔プログラムを用意し，クライアント側で RPC の実行時間を測ることで，データ転送のオーバーヘッドを計測した．送受信のメッセージサイズを変えながら計測した結果を**表 2** に示す．

Ninf-G と OmniRPC は，遠隔プログラムが起動された時点でクライアントとの TCP 接続を確立し，以降それを維持しているので，Internet 越しのデータ転送の性能が高い．NetSolve は 1 回の RPC においても，TCP の接続確立が 6 回以上行われるため，その分のオーバヘッドが大きいといえる．

次に，RPC 実行時の引数にファイルを指定した場合の RPC の実行のオーバーヘッドを測定した．ファイル転送の機能は，既成のアプリケーションを遠隔プログラムとしてそのまま使いたい場合など，入出力のデータをファイルとしてやり取りする場合に利用され，Ninf-G と NetSolve にて提供されている．Ninf-G は，Globus の GASS を利用して，起動時に確立した TCP 接続とは別の接続を使ってファイルを転送する．一方，NetSolve は，データ転送は通常の手順を踏み，転送の前後にファイルの読み書きをシステム内部で行っている．GASS の性能がそれほど高くないため，表 2 では，LAN において NetSolve のファイル転送が Ninf-G よりも良い性能を示している．

Ninf-G はデータの圧縮や暗号化もサポートしており，表 2 には Internet 越しに実験した結果を示す．実験に用いたデータの平均圧縮率は 18.7%である．これより，Internet 環境においては圧縮により，かなりの高速化が得られる可能性があることが分かる．また，データの暗号化によるオーバーヘッドは，それほど大きくないことが分かる．

7.2 検索速度の比較

情報サービスを積極的に利用し，情報検索にかかるコストが重要となるアプリケーションも考えられるため，検索の応答時間を比較した．ただし，これは登録されている情報量に依存し，NetSolve では検索時間だけでなくスケジューリングの時間も加わるため，一概に比較することは難しい．そこで今回は，1 ホストの情報だけをそれぞれの情報サービスに登録し，クライアントが情報サービスにアクセスし，そのホストの情報を得るまでの時間を計測した．情報サーバは AIST のノードで動かし，AIST および SDSC のノードをクライアントとして用いた．**表 3** のように，LAN での結果を中央値で比較すると両システムの差は小さいが，Internet では Agent の性能が悪い．それに対して，MDS の応答は時折，非常に遅くなることがあった．この理由としては，メモリとディスク間の同期が定期的に働くためであると考えられる．

Ninf-G では，MDS の検索速度が問題になる場合を考慮して，ローカルに置いた LDIF ファイルをクライアントで読み込む機能も提供されている．

表 2 単一 RPC の実行のオーバーヘッド（20 試行の最良値）[秒]

	Data	Ninf-G	NetSolve	OmniRPC
LAN (Array)	1 KB	0.15	0.091	0.00037
	1 MB	0.11	0.12	0.031
	5 MB	0.30	0.23	0.15
Internet (Array)	1 KB	1.1	3.1	0.12
	1 MB	7.3	9.4	6.8
	5 MB	30	32	31
LAN (File)	1 KB	0.16	0.10	–
	5 MB	1.5	0.25	–
Internet (File)	1 KB	1.7	3.2	–
	5 MB	29	32	–
Internet (Compress)	1 MB	3.1	–	–
	5 MB	9.6	–	–
Internet (Encrypt)	1 KB	1.1	–	–
	5 MB	31	–	–

表 3 検索速度の比較結果（20 回試行）[ミリ秒]

	Ninf-G		NetSolve	
	Med.	Max.	Med.	Max.
File	0.062	–	–	–
LAN	6.28	777	5.13	10.3
Internet	415	1960	724	833

8. 発展的な RPC

その他，標準化や有効性についてはまだ議論の余地があるものの，将来的に実装される可能性のある RPC のオプションとしては，リモートオブジェクトと複数 RPC の一括呼び出しが挙げられる．これらは GridRPC API の標準化とは別に，各システムにて独自に検討が進められている．

リモートオブジェクトとは，遠隔で実行されているプログラムをオブジェクトとみなし，そのサービス内部に変数を保存させることができる機能である．関数の代わりに，クラスとそれが保持する変数やメソッドを IDL に定義しておいて，クライアントではクラスを指定してオブジェクトハンドルを作成し，RPC の実行時にメソッドを指定す

表 1 実験環境

Site	# nodes	CPU	Memory	OS
AIST	33	Pentium III 1.4GHz × 2	2.2 GB	RedHat Linux 7.3 (Kernel 2.4.20)
SDSC	29	Xeon 2.4 GHz × 4	3.8 GB	Rocks 3.3 (Kernel 2.4.21)
LAN	1 hop, Latency: 0.085 ms, Throughput in 1 MB message: 928 Mbps			
Internet	12 hops, Latency: 119 ms, Throughput in 1 MB message: (SDSC → AIST) 2.23 Mbps, (AIST → SDSC) 3.99 Mbps			

る．これは Ninf-G と OmniRPC で提供されている．リモートオブジェクトの利点は，毎回同じデータをサーバに送信して計算を行う必要があるアプリケーションにおいて，通信の無駄を省ける点である．

複数 RPC の一括呼び出しの機能は，NetSolve において Request Sequencing と Request Farming という形で試験的に実装されている．Request Sequencing では，関連する RPC 要求をまとめた DAG（Directed Acyclic Graph）を構築して一括要求を行うことにより，最初の RPC の計算で得られたデータの一部をクライアントに戻すことなく，次の RPC において，そのデータを利用することができる．

また，Request Farming は，入力パラメータを変えながら同一の計算を多数行う場合に，反復子を使って簡単に記述できる機能を提供する．これの目指すところは，アプリケーション開発者が個々の要求を管理することなく，タスクの処理速度や利用できるサーバ数に応じて，自動的に最適な形で RPC が実行されることである．しかし，現在の実装では，RPC の実行の最適化はまだ不十分である．

9. ま と め

本稿では，代表的な GridRPC システムである Ninf-G, NetSolve，OmniRPC の比較を行い，各システムの設計目標や用途の違いにより生じているサポート機能の差異について明らかにし，アプリケーション・ユーザが開発時やシステムの選択時に参考にできる情報を示した．特に，Ninf-G は Globus を基盤にしてタスク並列を実行するための枠組を提供し，一方，OmniRPC は個人が手軽に利用できることを目指して，敷居の高い Globus をあえて使わないオプションを提供している．また，NetSolve は遠隔の数値計算ライブラリを容易に利用できることを目指して，スケジューリングや対話的な呼び出しの機能を提供している．今後は，比較によって得られた成果を GridRPC システムの改良や標準化の策定に生かしていく予定である．

謝辞 本研究を行うにあたり，貴重なご意見を頂いた武宮（産総研）氏に深く感謝いたします．

本研究は ApGrid および PRAGMA における研究活動の一環として行われた．ApGrid および PRAGMA の参加研究機関，特に実験に計算資源を提供頂いた SDSC に感謝いたします．

なお，本研究の一部は文部科学省「経済活性化のための重点技術開発プロジェクト」の一環として実施している「超高速コンピュータ網形成プロジェクト（NAREGI: National Research Grid Initiative）」によるものである．

参 考 文 献

1) Seymour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C. and Casanova, H.: Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *Proceedings of 3rd International Workshop on Grid Computing* (Parashar, M.(ed.)), pp. 274–278 (2002).
2) Sekiguchi, S., Sato, M., Nakada, H. and Nagashima, U.: –Ninf–: Network base information library for globally high performance computing, *Proceeding of Parallel Object-Oriented Methods and Applications (POOMA)* (1996).
3) Casanova, H. and Dongarra, J.: Netsolve: A Network Server for Solving Computational Science Problem, *Supercomputer Applications and High Performance Computing*, Vol. 11, No. 3, pp. 212–223 (1997).
4) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T. and Matsuoka, S.: Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing, *Grid Computing*, Vol. 1, No. 1, pp. 41–51 (2003).
5) 佐藤三久, 朴泰祐, 高橋大介: OmniRPC：グリッド環境での並列プログラミングのための Grid RPC システム, 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. 34–45 (2003).
6) GGF: http://www.gridforum.org/.
7) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *Supercomputing Applications and High Performance Computing*, Vol. 11, No. 2, pp. 115–128 (1997).
8) Agrawal, S. and Dongarra, J.: Hardware Software Server in NetSolve, Technical Report : ICL-UT-2002, University of Tennessee (2002).
9) Wolski, R., Spring, N. and Hayes, J.: The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, *Future Generation Computing Systems*, Vol. 15, No. 5–6, pp. 757–768 (1999).