Ninf-G2：大規模Grid環境での利用に即した高機能，高性能GridRPCシステムの実装と評価

武宮 博[†] 田中良夫[†]
中田秀基[†,††] 関口智嗣[†]

Gridプログラミングモデルの1つであるGridRPCの参照実装としてNinf-G2の開発を行い，性能を評価した．広域に分散した複数台のクラスタから構成される大規模Grid環境上でアプリケーションを効率良く実行することを目的とするNinf-G2は，関数ハンドル同時生成機能やリモートオブジェクトを実装することで，遠隔手続き呼び出しにともなう起動コストや通信コストの低減を図るとともに，ハートビート機能や関数ハンドル作成タイムアウト機能，サーバ属性の個別設定機能を提供することで，非均質，不安定で動的に変化するGrid環境への対応を図っている．典型的なタスク並列アプリケーションである気象シミュレーションプログラムを対象に，6台のクラスタから構成されるGridテストベッド上でNinf-G2の性能評価を行った．その結果，個々のタスクの実行時間が十数秒から数十秒程度の比較的粒度の小さいシミュレーションであっても，200台以上のプロセッサを用いて効率的に実行可能であることが分かった．

Ninf-G2: Development and Evaluation of a High Performance GridRPC System for Large Scale Grid Environment

Hiroshi Takemiya,[†] Yoshio Tanaka,[†] Hidemoto Nakada[†,††] and Satoshi Sekiguchi[†]

A high performance GridRPC system called Ninf-G2 has been developed and its performance was evaluated. Ninf-G2 aims to enable applications to run efficiently on a large scale Grid environment which consists of clusters widely distributed over a network. It tries to reduce costs for start-up and communication by simultaneous function handles creation function and remote object mechanism. In addition, it tries to cope with heterogeneous, unstable, and dynamically varying grid environment by heart-beat monitoring function, timeout mechanism in creating function handles, and methods to specify server-dependent attributes. Using 6 distributed clusters, performance of Ninf-G2 was evaluated by running an atmospheric simulation program which is a typical task parallel application. Good performance was attained on a grid environment with more than 200 processors even in the case of applications having many small grained tasks.

1. はじめに

Grid環境での遠隔手続き呼び出しを支援するGridRPCは，Grid環境における有効なプログラミングモデルの1つとして期待されている[1),2)]．Grid技術の標準化を進めるGlobal Grid Forum（GGF）[3)]においてもGridRPC Working Groupが作られ，GridPRC APIの標準化に関する議論が進められている[4)]．

GridRPCは，パラメータサーベイ等多くの科学，工学分野において用いられるタスク並列型の処理や，リモートライブラリコール☆を容易に実現できるという特徴を持つ．したがって，その実装システムを開発，提供することは，大規模な計算資源を必要としながらもGridシステムの複雑さによりその利用を断念していた科学者，工学者によるGridアプリケーションの構築を加速，推進する．実際，独立なタスクが多数生成される分子立体配座探索にGridRPCを利用した事例[17)]や，重力計算を高速に実行できるGRAPE-6ク

† 産業技術総合研究所グリッド研究センター
National Institute of Advanced Industrial Science and Technology
†† 東京工業大学
Tokyo Institute of Technology

☆ 大量のメモリあるいは長時間の実行を要するため手元の計算機では実行不可能な処理を，遠隔地に存在する大規模，高速な計算機資源上に実装しアクセスすることで実行可能とすること．

ラスタへのアクセスに GridRPC を利用した事例[18]が報告されてきている．また，より高機能な Grid ミドルウェア構築の際のインフラとして利用できるという特徴もあり，GridRPC を利用したタスクファーミングシステム構築事例[21] 等が報告されている．

これまでに， Ninf-G[5]， NetSolve[6]， DIET[7]，OmniRPC[8] 等，いくつかの GridRPC 実装システムが開発され，性能評価や実用性評価が行われている．しかしながら，現状では基本的な通信性能や，数十プロセッサ規模の処理における性能評価にとどまっており，GridRPC 実装システムの大規模環境への適用可能性はまだ明らかではない．

Ninf-G2 は，数サイト〜十数サイトに分散配置された数十〜数百プロセッサ規模のクラスタにより構成される Grid 環境上でアプリケーションの効率的な実行を可能にすることを目的として開発されたシステムである．我々が対象としている環境は，計算機，ネットワーク等が非均質であり，その稼動状況は不安定で，複数の利用者によって共有されているという特徴を持つ．したがって，その環境上で動作するシステムには，それらの特徴に適応できる機能が求められる．さらに，その環境上でアプリケーションを効率的に実行するためには，処理の高性能性，スケーラビリティが求められる．我々が開発を進めている Ninf-G2 には，これらの要件に対処するために種々の機能が実装されている．本稿では，それらの機能について述べる．

また，潜在的な Grid アプリケーションプログラマに対して彼らの有するアプリケーションへの Ninf-G2 の適用性に関する指針を与えることを念頭に，通信性能等 Ninf-G2 の基本的な性能を測定するとともに，典型的なタスク並列型アプリケーションである気象シミュレーションプログラム[11] を対象とし，4 つのサイトに配置されたクラスタ，最大 250 プロセッサを用いて Ninf-G2 の性能評価実験を行った．実験では，1 回の実行時間が 10 数秒程度の比較的粒度の小さいシミュレーション（10 日予報），数分程度の粒度のシミュレーション（100 日予報），およびその間の粒度のシミュレーション（1 カ月予報），計 3 種を対象とし，実行性能およびスケーラビリティを評価した．

本稿では，次章で Ninf-G2 の概要を紹介した後，3 章において Ninf-G2 の実装方針およびその方針に基づいて実装された機能について述べる．4 章では，気象シミュレーションプログラムを用いて行った性能評価結果について述べ，Ninf-G2 を用いて数百プロセッサを使ったシミュレーションが可能であること，比較的粒度の細かい処理でも Grid 環境上で効率的に実行可能であることを示す．5 章で関連研究との比較を行った後，最後に現状と今後の課題についてまとめる．

2. Ninf-G2 概要

Ninf-G2 は Grid プログラミングモデルの 1 つである GridRPC の参照実装であり，Ninf-G の開発経験および Ninf-G を用いたアプリケーションによる性能評価[9),10)] から得られた知見に基づいて設計[12]，開発されたものである．

Ninf-G2 は，先に述べた GridRPC の持つ Grid 環境への親和性の高さを活かし，アプリケーションプログラマや Grid ミドルウェア開発者に対して彼らの持つソフトウェアを容易に Grid 環境上に実装可能とすることを目的とする．特に，数サイト〜十数サイトに分散配置された数十〜数百プロセッサ規模のクラスタにより構成される Grid 環境上でアプリケーションの効率的な実行を可能とし，これまでハードウェア等の物理的な制約により困難であった大規模アプリケーションを実現可能とする．近年の急速なクラスタ構築技術の進展，普及により，安価かつ高性能なクラスタを利用した並列計算が一般的になってきている．Ninf-G2 は各サイトに散在するこれらのクラスタを連携させ，1 つの巨大な仮想計算機として容易に利用可能とすることを狙う．

Ninf-G2 はローカルな環境にある計算機（クライアント計算機）から，リモートに分散配置された計算機（バックエンド計算機）上に実装されたプログラムを RPC の形式で呼び出し，実行する機能を提供する．クライアント計算機上で実行され，遠隔手続きを呼び出すプログラムをクライアントコンポーネントと呼ぶ．クライアントコンポーネントから呼び出されるバックエンド計算機上のプログラムを遠隔実行プログラムと呼ぶ．遠隔実行プログラムは関数ハンドルという形で抽象化され，クライアントコンポーネントからの呼び出しは関数ハンドルに対する呼び出しという形で実現される．

Ninf-G2 の特徴の 1 つとして，Grid 環境構築のためのインフラとして最も普及している Globus Toolkit のコンポーネントを基盤として利用していることがあげられる．具体的には，以下のようなコンポーネントを利用している（**図 1**）．

- MDS の利用
  MDS（Monitoring and Discovery Service）と呼ばれるディレクトリサービスを用いて，遠隔実行プログラムの呼び出し情報（プログラムのパス情報や引数情報）を管理する．MDS がこれらの情

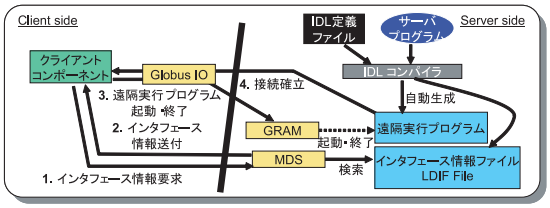

図 1 Ninf-G2 ソフトウェアアーキテクチャ
Fig. 1 Software architecture of Ninf-G2.

報を一括管理することにより，クライアントコンポーネントが個々に呼び出し情報を管理する必要がなくなる．

- GRAM の利用
  GRAM（Globus Resource Allocation Manager）と呼ばれる資源管理サービスを用いて，遠隔実行プログラムの起動を行う．これにより，遠隔地に配置されたクラスタのバッチキューをセキュアに利用できる．
- Globus IO の利用
  Globus IO と呼ばれる通信サービスを用いて計算機間のデータ送受信を行う．Globus IO は GSI（Globus Security Infrastructure）上に構築されているため，GSI に基づくセキュアな通信が実現される．

Globus Toolkit 上に実装された Ninf-G2 の高レベルな API を利用することにより，アプリケーションプログラマは Globus Toolkit の低レベルな API を用いて複雑なプログラミングを行うことなく標準技術に基づく Grid アプリケーションを開発でき，多くのサイトで動作させることができる．

3. Ninf-G2 の実装

3.1 実装方針

前章で述べた目的を達成するためには，資源の非均質性，動的な変化，不安定性といった特質を持つ Grid 環境への適応と，高性能性，スケーラビリティ，開発容易性といった Ninf-G2 を利用するアプリケーションプログラマからの要求への対処が求められる．以下に，Ninf-G2 開発に際して考慮した項目をあげる．

（1） Grid 環境への適応

- 資源の非均質性への対処
  Grid 環境上に存在するクラスタは，機種や規模が異なるだけでなく，運用されているバッチシステム，デフォルトで設定される環境変数，遠隔実行プログラムの実行パスといった実行環境も異なっている．さらに，サイトによってセキュリティポリシが異なるため，ネットワーク接続を許可されるポート番号や利用可能なプロトコル，認証方法等も異なる可能性がある．これらの違いを吸収し，クライアント計算機—バックエンド計算機間の接続や遠隔実行プログラムの実行を実現できる必要がある．
- 資源の動的な変化，不安定性への対処
  Grid 環境上のクラスタやそれらを接続するネットワークは多数の利用者によって共有されていることが一般的である．その結果，ネットワークやクラスタの負荷が動的に変動する．また，Grid 環境が大規模，広域になればなるほど，構成要素であるクラスタやネットワークの一部がダウンし利用不可能になることが頻繁になる．このような動的な資源の変化，不安定性に対処する必要がある．

（2） アプリケーションプログラマからの要求への対処

- 高性能性，スケーラビリティへの対処
  アプリケーションプログラマにとっての関心は，Ninf-G2 を利用することにより Grid 環境上でどの程度大規模な処理をどの程度効率的に実現できるかという点にある．したがって，遠隔実行プログラムの起動や終了コストの低減，通信コストの低減等を図って効率的な処理を実現し，大規模な処理を可能にする機能の実現が求められる．
- 開発容易性への対処
  どれだけアプリケーションの実行が効率的になっても，プログラム開発が困難であると，アプリケーションプログラマの負担が増大し，開発に労力と時間が必要になってしまう．もともと Ninf-G2 が準拠している GridRPC は，局所的な関数呼び出しと同様な形式で遠隔実行プログラムを実行する RPC のセマンティクスを提供しているため，Grid 環境の複雑さを意識することなく Grid アプリケーションを開発できる．しかし実際にそれが正常に動作することを Grid 環境上で検証することは困難である．この困難さを軽減するための機能の提供が必要である．
  また，RPC の途中経過のモニタリングやインタラクティブな制御等，より柔軟な制御を求めるプログラマの要求を実現可能にするための機構を提供することにより，プログラマの負担を軽減することも重要である．

3.2 Ninf-G2 の実現機能

3.1 節で述べた実装方針に基づき，Ninf-G2 では以下のような機能を実装，提供している．

(1) Grid 環境への適応

- 資源の非均質性への対処

サーバごとの属性設定機能

Ninf-G2 では，非均質性への対処として，データ転送モード，通信路の暗号化手法，利用通信ポート，バックエンド計算機上で有効なバッチシステム，設定したい環境変数等に対して種々のオプションを提供し，利用者がバックエンド計算機単位で指定できるようになっている．たとえば，バッチシステムの違いに関しては対応する GRAM の job manager の種類を，通信路の暗号化に関しては SSL を用いるかどうかを指定できる．これらの情報は，コンフィグレーションファイルに規定することで有効になる．

- 資源の動的な変化への対処

関数ハンドル作成タイムアウト機能

複数の利用者がバックエンド計算機にアクセスしている状態では，多数のジョブがバッチシステムに投入され，GRAM を介してジョブを投入しても長時間にわたって実行が開始されない可能性がある．このような場合，一定時間を過ぎても実行が開始されないジョブをキャンセルするための機能が必要である．Ninf-G2 では，タイムアウト値をコンフィグレーションファイルに設定することで，自動的に遠隔実行プログラムの起動がキャンセルされ，クライアントコンポーネントに対し関数ハンドル作成の失敗が通知される．

ハートビート機能

不安定な Grid 環境で長時間にわたってアプリケーションを実行する場合，定期的に遠隔実行プログラムの実行状況をチェックし，正常に動作していることを確認できることが望ましい．Ninf-G2 では，遠隔実行プログラムから定期的にクライアントコンポーネントに対しハートビートを発行することにより，遠隔実行プログラムの実行状態を確認できる機能を提供している．

(2) アプリケーションプログラマからの要求への対処

- 高性能性，スケーラビリティへの対処

リモートオブジェクトの実装

タスク並列型の処理では，遠隔実行プログラムに渡されるデータの大半が共通で，一部のパラメータのみが変化することが多い．このような処理では，遠隔実行プログラムに何度も同じデータを転送しなければならず，無駄な通信が発生する．遠隔実行プログラムが呼び出し終了後も起動を継続して状態を保持するようにし，さらに複数の関数呼び出しを受付可能にすれば，最初の呼び出し時のみすべてのデータを転送し，その後は変化するパラメータのみを別の関数呼び出しとして渡すことで，通信量を低減できる．Ninf-G2 では，そのような機能を持つ遠隔実行プログラムをリモートオブジェクトとして実装し，呼び出すことを可能にしている．

関数ハンドル同時作成機能

クラスタを利用してタスク並列型処理を実行すると，クラスタ上に多数の遠隔実行プログラムが起動される．遠隔実行プログラムの起動は Globus Toolkit の GRAM を介して行われるが，GRAM の呼び出しには，認証やバックエンド計算機上のバッチ処理システムへのジョブ投入のためのコストが発生する．したがって，個々の遠隔実行プログラムの起動のたびに GRAM を呼び出すと起動コストが増大する．Ninf-G2 では，GRAM の提供する複数のジョブを一度に起動する機能を利用し，1 回の GRAM 呼び出しでまとめて複数の遠隔実行プログラムを起動する機能を実装している．

MDS バイパス機能の提供

Ninf-G2 では，遠隔実行プログラム呼び出しの際に必要な引数情報を Globus Toolkit の MDS を利用して取得している．しかし，MDS の検索コストは大きく，場合によっては数十秒程度かかってしまうことがある．また，MDS サーバの動作は不安定で，つねにインタフェース情報が取得できるとは限らない．

そこで，Ninf-G2 では MDS を利用した引数情報取得に加え，MDS を利用せずクライアント計算機上に格納されたローカルなファイルから引数情報を取得する機能も提供することで引数情報取得コストの低減を図っている．

- 開発容易性への対処

デバッグ支援機能の提供

Grid 環境におけるプログラムのデバッギングは分散した計算機上で複数のプログラムが同時に実行されるため，非常に困難である．Ninf-G2 は，デバッギングを支援するために，遠隔実行プログラムの出力をクライアント計算機にリダイレクトする機能や，Ninf-G2 あるいは Globus Toolkit から出力されるエラーメッセージのロギング機能を提供するほか，遠隔実行プログラム起動時に自動的に gdb を起動する機能も提供している．これらの機能は，コンフィグレーションファイルでオ

プションを指定するだけで有効となる．

コールバック機能の提供

計算の途中経過を出力させ，その結果に基づいてそのプログラムの実行を制御したりすることは多くの処理で行われているが，GridRPC の枠組みでは遠隔実行プログラムが実行を終了するまでクライアントコンポーネントにアクセスする手段がないため，途中経過の出力は困難である．

Ninf-G2 では，クライアントコンポーネントがあらかじめ登録しておいた関数を遠隔実行プログラム側から呼び出すコールバック機能を提供することで，この機能を容易に実装可能としている．

通常のプログラムが関数呼び出しの際に関数ポインタを引数として指定すると，呼び出された関数内から渡された関数を呼び出すことが可能になるのと同様に，遠隔実行プログラムに渡す引数としてクライアントコンポーネントが関数ポインタを指定することで，遠隔実行プログラムからその関数を呼び出すことが可能になっている．

4. 性能評価

4 つのサイトに分散配置された 6 台のクラスタを用いて Ninf-G2 の性能評価を行った．実験に際して，Ninf-G2 の基本性能に関する測定のほか，典型的なタスク並列アプリケーションである気象シミュレーションプログラムを用いて実行性能を測定した．実施した実験および目的は，以下のとおりである．

- 基本性能測定

(1) Ninf-G2 通信性能測定

通信速度の異なる 4 サイトのクラスタを用い，Ninf-G2 の通信性能を測定する．同時に Ninf-G2 が基盤として用いている TCP/IP，Globus-I/O の通信性能を測定し，結果を比較する．

(2) Ninf-G2 遠隔実行プログラム起動時間測定

Ninf-G2 において新たに実装された関数ハンドル同時作成機能と従来の個別関数ハンドル作成機能を利用した場合の遠隔実行プログラム起動時間を測定し，関数ハンドル同時作成機能の有効性を検証する．

- アプリケーション実行性能測定

(1) 単一クラスタ上での実行性能測定

通信速度の異なる 2 サイトのクラスタ上で各々気象シミュレーションを実行し，通信速度の違い，利用プロセッサ数の違い，処理粒度の違いがシミュレーション実行性能に与える影響を調べる．

(2) 複数クラスタ上での実行性能測定

2 サイトのクラスタを同時に利用して気象シミュレーションを実行し，利用プロセッサ数を変更してシミュレーション実行性能に与える影響を調べる．

(3) Ninf-G との実行性能比較

Ninf-G と Ninf-G2 を用いて 3 サイト 4 クラスタ上で気象シミュレーションを実行し，Ninf-G2 に新たに実装された機能の有効性を検証する．

(4) 大規模分散実行における動作安定性の検証

太平洋をまたがって配置された 4 サイト 5 台のクラスタ計 500 プロセッサを用いて気象シミュレーションを実行し，広域に分散する大規模クラスタを利用した場合の Ninf-G2 システムの動作安定性を検証する．

表 1 実験に用いたクラスタの仕様
Table 1 Hardware specification of clusters used for performance evaluation.

サイト名	クラスタ名	CPU	CPU 数
AIST	KOUME	PentiumIII 1.4 GHz	10
	UME	PentiumIII 1.4 GHz	64
TITECH	PRESTO3	Athlon 1.6 GHz	256
KISTI	VENUS	PentiumIV 2.0 GHz	64
	JUPITER	PentiumIV 1.7 GHz	16
KU	AMATA	Athlon 1.1 GHz	14

表 2 AIST-他のサイト間のネットワーク性能
Table 2 Network performance between AIST and other sites.

サイト名	レイテンシ [msec]	スループット [MB/sec]
AIST	0.04	122.1
TITECH	1.5	9.3
KISTI	17.2	1.1
KU	80	0.09

4.1 実験環境

性能評価にあたって，産業技術総合研究所（AIST）の 2 台のクラスタ（UME および KOUME），東京工業大学（TITECH）のクラスタ（PRESTO3），韓国 Korea Institute of Science and Technology Information（KISTI）の 2 台のクラスタ（VENUS および JUPITER），タイ Kasetsart University（KU）のクラスタ（AMATA）を利用した．各クラスタのハードウェア仕様を**表 1** に示す．

実施したすべての実験において，KOUME クラスタをクライアントとし，他の計算機をバックエンド計算機として利用した．UME クラスタは KOUME クラスタとギガビット LAN で接続されている．その他のサイトは，SuperSINET（TITECH）および APAN（KISTI および KU）を介して AIST と接続されている．KOUME クラスタと各サイトのクラスタ間でソケットを用いた ping-pong プログラムを実行して測定したネットワーク性能を**表 2** に示す．

表 1 および表 2 から分かるように，今回の実験環境を構成するクラスタおよびネットワークの性能は非均質であり，典型的な Grid 環境となっている．

4.2 気象シミュレーションシステム

性能評価に用いた気象シミュレーションシステムは，Ninf-G2 を用いて順圧 S-model[9] と呼ばれる逐次 FORTRAN プログラムを Grid アプリケーションとした（Ninf 化した）ものである．順圧 S-model は，短中期における大域的な気象変動を予測することを目的として開発されたプログラムである．

気象予報は系の持つカオス的な振舞いのために，精度を維持することが困難である．順圧 S-model では，予報精度を維持する 1 つの手段として，アンサンブル予報の手法を採用している．アンサンブル予報は，擾乱を加えた多数のシミュレーション（サンプルシミュレーション）を行い，それらの結果に対して統計処理を行うことで，初期データとして用いられる観測データに含まれる誤差や計算途中に発生する誤差の成長を抑え，予報精度を維持する手法である．

各サンプルシミュレーションは独立に実行できるため，順圧 S-model は典型的なタスク並列プログラムとなっている．我々は原プログラムからシミュレーション実行部分を抜き出し遠隔実行プログラムとすることで，順圧 S-model を Ninf 化した．

性能評価においては，10 日予報，1 カ月予報，100 日予報の 3 種類のシミュレーションを実行した．サンプルシミュレーションの実行時間は，各々12 秒（10 日予報），35 秒（1 カ月予報），および 120 秒程度（100 日予報）となっており，10 日予報の処理粒度はかなり小さい．

また，各シミュレーションでは 3.4 KB の入力データがバックエンド計算機に転送され，各々138 KB（10 日予報），408 KB（1 カ月予報），1.35 MB（100 日予報）の結果データがクライアントに転送される．

4.3 Ninf-G2 基本性能の測定

4.3.1 Ninf-G2 通信性能の測定

Ninf-G2 および Ninf-G2 が基盤として利用している Globus IO，さらにその基盤となる TCP/IP を用いた ping-pong プログラムを実装し，各々の通信性能を測定し比較することで，Ninf-G2 固有のオーバヘッドを調査した．測定に際しては，KOUME クラスタをクライアントとし，AIST（UME），TITECH，KISTI（VENUS），KU のクラスタ 1 台ずつをバックエンド計算機として利用した．

図 2 は各バックエンド計算機とクライアント計算機間で規定長データを転送する際の通信速度を示している．図から分かるように，TCP/IP と Globus-IO ではすべてのサイトにおいてほとんど性能差がでない．それに対して，Ninf-G2 は，両者と比較してデータ長が短い領域で性能の低下が見られる．これは，Ninf-G2 が Globus-IO 上で Ninf プロトコルを用いて通信を行っており，その通信コストが大きいことを示している．しかしながら，広域環境での実験結果ではデータ長が 10 Kbyte を超えると TCP/IP や Globus-IO とほぼ同一の通信性能を達成している．一方，高速 LAN で接続された AIST サイト内での実験では，データ長が 10 KB を超えても Ninf-G2 と TCP/IP，Globus-IO

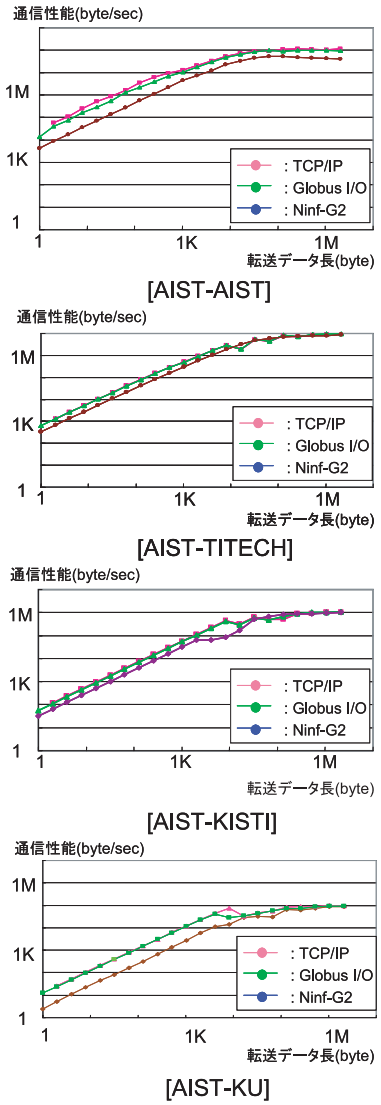

図 2 Ninf-G2，Globus-IO，TCP/IP を用いたサイト間通信性能
Fig. 2 Network performance between AIST and other sites using Ninf-G2, Globus-I/O, and TCP/IP.

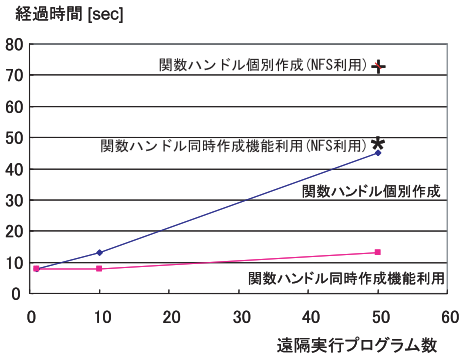

図 3 UME クラスタにおける遠隔実行プログラム起動時間
Fig. 3 Invocation time of remote executables on the UME cluster.

との通信性能差が残っている．これは，Ninf-G2 が独自で管理している通信バッファからアプリケーションの変数にデータをコピーしていることが主原因である．

このように，高速 LAN 環境での通信性能およびデータ長が短い場合の通信性能には改善の余地があるが，Ninf-G2 が想定している広域に分散した複数サイトにわたって大規模データを送受信するアプリケーションの実行においては，TCP/IP や Globus IO と同等の性能を実現しているということができる．

4.3.2 Ninf-G2 遠隔実行プログラム起動時間の測定

Ninf-G2 において新たに実装された関数ハンドル同時作成機能の有効性を検証するために，関数ハンドル同時作成機能を利用した場合と個々に関数ハンドルを作成した場合において，すべての気象シミュレーションプログラムが起動され引数データを受信するまでの時間を測定した．実験にはクライアント計算機として KOUME クラスタ，バックエンド計算機として UME クラスタを用いた．

図 3 に計測結果を示す．利用プロセッサ数が 10 程度であれば，どちらの機能を利用しても起動時間にそれほど大きな差は見られない．しかし，50 プロセッサになると，関数ハンドル同時作成機能を利用した場合に比べ個々に関数ハンドルを作成した場合では起動に 4 倍程度の時間がかかっている．原因は，個々に関数ハンドルを作成した場合，利用プロセッサの数に比例して GRAM の job manager がクラスタのフロントノードに起動され負荷が増大すること，および job manager が発行するジョブ実行要求がバッチシステムにおいてシリアライズされることによる．実際，50 プロセッサを利用した場合にフロントノードの負荷を計測すると，20 を超える値が観測された．

また，100 以上の関数ハンドルを個々に作成しようとすると，フロントノードの負荷が大きくなりすぎ起動に失敗するという現象が見られた．それに対し，関数ハンドル同時作成機能を利用すると安定して 200 個以上のプロセッサを利用できた．

図 3 において，∗ 印および+印で示されているのは NFS 共有されたディレクトリに遠隔実行プログラムを格納した場合の起動コストを示している．NFS を利用しない場合と比較して，平均して 30 秒程度の遅延が発生している．これは，NFS に対して各プロセッサからのプログラムロード要求が集中し，通信が錯綜してしまったことが原因である．

NFS におけるデータ転送の遅延は，起動だけでなく最初のシミュレーションの実行にも影響を与える．気象シミュレーションでは，シミュレーションパラメータに依存しない定常的な入力データをサーバ側に持たせファイル入力している．このファイルが NFS に置かれている場合，通信の錯綜により読み込みが遅延する．このため，以下の性能測定に際しては実行プログラムおよび入力データを各クラスタのローカルディスクに格納し，NFS に起因する実行の遅延を避けている．

クラスタのローカルディスクに実行ファイルやデータを格納すると，プログラムやデータがクラスタ全体に散在することになる．そのため，それらを修正する場合，NFS を利用した場合と比較して作業コストが増大する．しかし，頻繁にプログラムやデータを修正する開発フェーズでは NFS 環境で作業を行い，実行フェーズに移行した段階で最終的にプログラムやデータファイルをローカルディスクに転送するようにすれば，そのコストは軽減される．また，クラスタ内の環境は一般に均質であることから，NFS 環境からローカル環境への移行時に発生する作業は基本的にデータや遠隔実行プログラムの転送作業のみであり，大きな負担とはならない．

4.4 気象シミュレーション実行性能評価

4.4.1 単一クラスタにおける性能評価

処理性能に対する通信速度，処理粒度，利用ノード数の影響を調べるために，TITECH および KISTI に設置されたクラスタを利用して各々気象シミュレーションを実行した．実験では，遠隔実行プログラムの起動，終了時間，送受信時間，サンプルシミュレーション実行時間（Tr），RPC 実行時間（Tc），結果転送時間（Trec），総経過時間（Te）を測定した（**図 4** 参照）．スケーラビリティを調べるために，サンプルシミュレーションの総数は，プロセッサ数の 5 倍になるよう調整した．

図 5，**図 6** に，PRESTO3 クラスタと VENUS ク

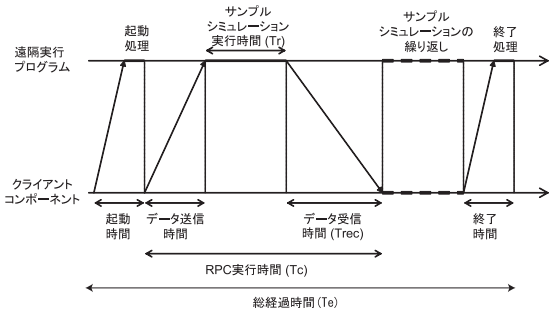

図 4 Ninf 化された気象シミュレーションの実行模式図
Fig. 4 Schematic execution model of the ninfied weather simulation.

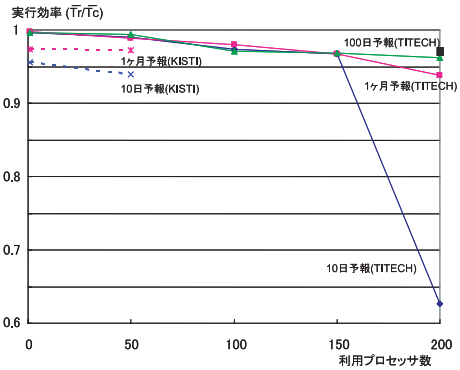

図 5 PRESTO3，VENUS を用いた気象シミュレーションの実行性能（平均サンプルシミュレーション時間と平均 RPC 時間比）
Fig. 5 Performance result of executing a weather simulation on PRESTO3 and VENUS clusters. The vertical axis means the ratio of the mean execution time of a sample simulation to the mean execution time of one RPC.

ラスタを利用した場合の実行性能結果を示す．図 5 は，平均サンプルシミュレーション実行時間と平均 RPC 実行時間の比を表したもので，RPC を利用したサンプルシミュレーション実行における通信の寄与を表す．一方，図 6 は，5 回のサンプルシミュレーションに要する平均時間と総経過時間の比を表しており，図 5 と比較して，初期，終了コスト，各プロセッサにおけるサンプルシミュレーション終了時間のばらつき等の効果が含まれる．図 5 は非常に多数のサンプルシミュレーションを行った場合の実行効率を，図 6 は実際の実行効率を表す指標となる．

図 5 では，TITECH において 200 プロセッサを利用して 10 日間予報を行った場合（以下，例外ケースと呼ぶ）を除き，すべての場合で 95%程度の実行効率を達成している．このことは，気象シミュレーション程度の粒度を持つ処理であれば，数十〜数百プロセッサ規模のクラスタを用いて効率良く実行可能であるこ

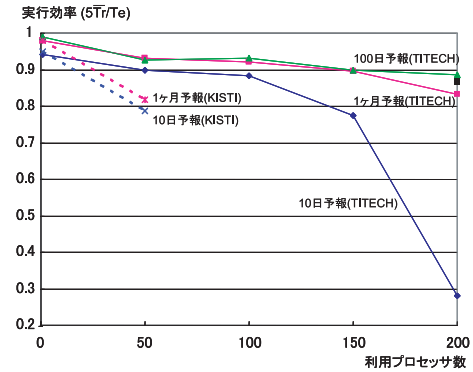

図 6 PRESTO3，VENUS を用いた気象シミュレーションの実行性能（平均 5 サンプルシミュレーション時間と総経過時間比）
Fig. 6 Performance result of executing a weather simulation on PRESTO3 and VENUS clusters. The vertical axis means the ratio of the mean time of executing 5 sample simulations to the total elapsed time.

とを示している．図 6 では，実行効率が 5〜10%低下してしまうが，例外ケースを除き 80%以上を維持している．

以下では図 5 の結果をもとに，処理性能に対する通信速度，処理粒度，利用ノード数の影響に関して考察する．

- 通信速度の影響
  TITECH と KISTI における同一利用プロセッサ数，同一予報期間のシミュレーションを比較すると，KISTI で実行したシミュレーションの実行効率が低くなっている．これは明らかに KISTI の通信コストが TITECH と比べて 10 倍程度大きい影響である．しかしながら，その低下は 50 プロセッサを利用した場合でも 3〜5%にとどまっている．
- 処理粒度の影響
  同一利用プロセッサ数で異なる予報期間のシミュレーション実行結果を比較すると，TITECH の場合，例外ケースを除いて処理粒度によらずほぼ同一の実行効率を達成している．これは，シミュレーション実行コストの増大と通信コストの増大が比例している，すなわちシミュレーションがうまくスケールして実行されていることを示している．
  一方，KISTI では，10 日間予報シミュレーションの実行効率が 1 カ月予報シミュレーションの実行効率よりも低下している．これは，データ送信時間が原因ではないかと推測される．気象シミュレーションでは，予報期間に比例してサンプルシ

ミュレーション実行時間および結果データのサイズは増大する．しかし，送信される初期データのサイズは予報期間に依存せず一定である．TITECH のように高速かつ低レイテンシのネットワークを利用する場合，サンプルシミュレーションの実行時間と比較してその転送コストは非常に小さいため実行効率にはほとんど影響しない．しかし，KISTI のように低速かつ高レイテンシのネットワークを利用する場合にはその効果が現れ，相対的に実行時間の短い 10 日予報シミュレーションの実行効率がより大きく低下してしまうと考えられる．しかしながら，今回の実験結果からは上記の推測を裏付ける明確な傾向を見い出すことはできなかった．この原因に関しては，さらなる調査が必要である．

- 利用プロセッサ数の影響
  同一予報期間のシミュレーションにおいて利用プロセッサ数を変化させると，すべての場合において，利用ノード数が増加するに従って徐々に実行効率が低下している．この実行効率の低下原因は，2 通り考えられる．すなわち，1 つは単位時間あたりのデータ転送量がクライアント計算機とバックエンド計算機を結ぶネットワークの転送能力を超えてしまい，ネットワークがボトルネックになってしまっている可能性，もう 1 つは多数の遠隔実行プログラムからの通信がクライアントに集中してしまうために，クライアントの通信処理に時間がかかってしまっている可能性である．この原因を調査するために，特に実効効率の低下が著しい例外ケースを取り上げ，クライアント計算機の 2 ノードを用いて同時にクライアントコンポーネントを起動し，各々PRESTO3 クラスタの 100 プロセッサを使って 10 日予報 500 サンプルシミュレーションを実行させた．その結果，計 1000 サンプルシミュレーションを約 90%の実行効率で行うことができた（図 5 および図 6 で黒い四角で表示）．このことから，例外ケースにおける性能劣化の原因はクライアントがボトルネックになっていたためであると結論できる．

このことは，クライアントコンポーネントをクラスタ上で並列実行させることで，より大規模なシミュレーションを効率良く実行できる可能性があることを示唆している．そのためには，クライアントコンポーネントを MPI，OpenMP 等を用いて実装することが考えられる．また，Ninf-G2 をカスケードさせ，クライアントから同一クラスタ上に起動された複数のサブクライアントを呼び出し，さらにバックエンド計算機上の遠隔実行プログラムを呼び出すといったことも考えられる．

表 3 PRESTO3 および UME クラスタを同時に用いた気象シミュレーション実行性能結果（秒）
Table 3 Performance results of executing a weather simulation using PRESTO3 and UME clusters simultaneously (sec).

CPU 数 (UME+PRESTO3)	Trec	Tc	Te
50+50	0.21	37.3	208
0+100	0.60	36.4	194
50+150	0.51	37.6	225
0+200	1.12	38.1	215
50+200	0.53	37.6	225

しかしながら，MPI や OpenMP を利用したり Ninf-G2 をカスケードさせてクライアントを並列化したりすることは，一般のアプリケーションプログラマにとって負担が大きく，また Ninf-G2 の特徴の 1 つである “アプリケーションの容易な Grid 化” という特徴を損ねてしまうことになる．

また，今回の実験ではサンプルシミュレーション数を静的に等分し 2 つのクライアントコンポーネントに担当させるだけで効率的に実行できたが，非均質で動的に負荷が変動する Grid 環境では，一般に実行するサンプルシミュレーションの動的な配分が必要とされる．並列に動作するプロセス間で動的な配分を実現するアルゴリズムをアプリケーションレベルで実装することも，一般のアプリケーションプログラマにとって負担が大きい．

したがって，クライアントコンポーネントの並列化，それにともなう並列プロセス間での動的なタスク配分を隠蔽するより抽象的なアプリケーションインタフェースを Ninf-G2 上に構築し，提供することが望ましい．

4.4.2 複数クラスタにおける性能評価

複数クラスタを利用したシミュレーションの実行性能を評価するために，UME クラスタと PRESTO3 クラスタを同時に利用し，1 カ月予報シミュレーションを行った．UME クラスタのプロセッサ数は 50 で固定し，PRESTO3 クラスタのプロセッサ数を 50，150，200 と変化させた．前節の実験と同様，サンプルシミュレーション数は利用プロセッサ数の 5 倍に設定した．

実行結果を**表 3** に示す．表中の PRESTO3 クラスタのみを使った同一規模の結果と比較して，平均結果転送時間は短くなっている．これは，サイト間通信を行わなければならない PRESTO3 クラスタに対して，LAN 環境での高速な通信を利用できる UME クラス

タを加えた影響であると考えられる．

平均 RPC 実行時間および総経過時間について比較すると，平均 RPC 実行時間はほとんど差異がないにもかかわらず，経過時間は 2 台のクラスタを利用した場合のほうが若干長くなっている．これは，UME クラスタでの遠隔実行プログラム起動時間が PRESTO3 クラスタより若干長くなっているためである．

また，2 台のクラスタで 50 プロセッサずつ利用した場合を基準にすると，50+150 プロセッサを使った場合および 50+200 プロセッサを用いた場合の実行効率は 90%を超える．これは，200 プロセッサを超える大規模な Grid 環境でも Ninf-G2 を用いて効率的なシミュレーションが可能であることを示している．

4.5 Ninf-G との実行性能比較

先に述べたように，Ninf-G2 は Ninf-G の構築経験に基づいて設計，開発されたものである．この両者の性能を比較し，新たに実装された機能の有効性を検証するために，実行性能比較実験を行った．実験では UME クラスタ（3CPU），KISTI の 2 台のクラスタ（各々3CPU），および KU クラスタ（1CPU），計 10CPU を利用し，100 日予報を行う 50 サンプルシミュレーションを実施した．これは，文献 11) で行った実験と同一の仕様となっている．プログラムのアルゴリズムも前回のものとほぼ同一であるが，各バックエンド計算機に対して関数ハンドルを作成する際，関数ハンドル同時作成機能を利用している点が前回と異なっている．

実行結果を**図 7** に示す．グラフの横軸は経過時間を，縦軸は利用したバックエンド計算機の各ノードにおける処理の経過を示している．図より，前回 1,200 秒要した処理が 850 秒程度で終了していることが分かる．実行時間短縮の原因は，以下の 2 点である．

(1) 起動時間の短縮

前回の実験では遠隔実行プログラムの起動に 60〜180 秒程度要していたのが，今回の実験では 10〜30 秒程度で起動が終了した．起動時間が短縮されたのは，関数ハンドル同時作成機能の利用および非同期 RPC 呼び出し関数がバックエンド計算機への引数転送の終了を待たずに戻るようになったこと，および複数の遠隔実行プログラムへの引数転送がマルチスレッドにより並列処理可能となったことによる．

まず，関数ハンドル同時作成機能を利用することにより，関数ハンドル作成時間が短縮された（図中，初期処理コストとして表示）．また，非同期 RPC 関数が引数転送終了を待たずに戻るようになったため，すべてのノードに対しほぼ同時に RPC 呼び出しを行えるようになった．引数転送処理は Ninf-G2 内でマルチスレッドにより並列処理され，サーバが起動すると即座に引数転送が開始される．それに対し，Ninf-G では RPC 関数の呼び出し順に引数転送が行われ，かつ個々の関数は引数の転送が完了するまでブロックしていた．そのため，遠隔実行プログラムの起動が遅いノードや引数転送に時間がかかるノードが存在すると，そのほかに早く起動されたノード，引数転送の早いノードが存在しても待たされてしまっていた．

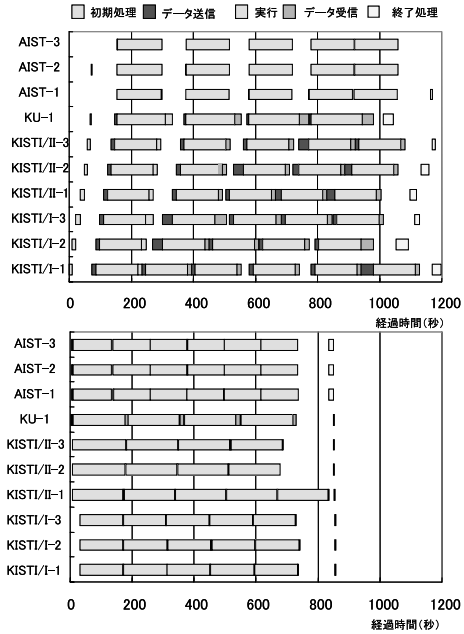

図 7 UME, VENUS, JUPITER, AMATA クラスタを用いた気象シミュレーションの実行経過．上図は Ninf-G を用いた場合，下図は Ninf-G2 を用いた場合の結果を示す

Fig. 7 Execution profile of the weather prediction simulation using UME, VENUS, JUPITER, and AMATA simultaneously. The upper diagram shows the result of using Ninf-G and the lower shows that of using Ninf-G2.

(2) 通信効率の向上

Ninf-G においては引数データを XML 形式に変換し転送を行っていた．また，複数の引数を転送する場合，各引数の転送ごとにバックエンド計算機からの ACK を待っていた．それに対し，Ninf-G2 では XDR 形式あるいは binary 形式で一括転送している．そのため，通信コストが大幅に減少するとともに，複数のバックエンド計算機間の競合が起こらなくなり，バックエンド計算機のアイドル時間がほとんどなくなった．

これらの違い以外にも，Ninf-G では関数ハンドルを個々に作成する仕様となっていることから，前節で述べたように単一クラスタ上に多数の遠隔実行プログラ

ムを起動することが困難であるという問題が存在する．

これらの結果から，Ninf-G2 は Ninf-G と比較して，Grid 環境での複数のクラスタを利用した大規模シミュレーションの実行に関して大きく改善されたということができる．

4.6 大規模分散実行における動作安定性の検証

Ninf-G2 の開発目標の 1 つである数サイト〜十週サイトに分散配置された数十〜数百プロセッサ規模のクラスタにより構成される Grid 環境上での大規模アプリケーションの実行が可能であるかどうかを検証するために，米国フェニックスで開催された国際会議 SC2003 において，気象シミュレーションのデモンストレーションを行った．デモンストレーションでは，AIST の KOUME クラスタをクライアントとし，TITECH，AIST，KISTI のクラスタに加えて米国 TeraGrid [16] のクラスタをバックエンド計算機として利用し，計 500 プロセッサを用いて 1000 個の 10 日予報サンプルシミュレーションを安定して実行することができた．

実行に要した時間は，約 150 秒であった．4.4.1 項，4.4.2 項で示したように，この場合，シミュレーションの粒度に対してバックエンドプロセッサの数が多すぎてクライアントがボトルネックとなるため，実行効率は高くない．しかし，日米韓にわたって広域に散在する大規模クラスタを用いて Ninf-G2 が安定して動作したことは，数百〜千プロセッサ規模の Grid 環境における Ninf-G2 を用いたシミュレーションの実行可能性を示唆したという点で評価できる．

5. 関 連 研 究

GridRPC は，Grid 環境上でタスク並列処理やリモートライブラリコールを実現するための唯一の手段ではない．また，Ninf-G2 以外にいくつかの GridRPC 実装系，あるいは広域分散環境への適用を考慮した RPC 実装系が存在する．本章では，タスク並列処理やリモートライブラリコールを実現するという観点から，GridRPC と他のプログラミングモデル，ツールとの比較を行うとともに，Ninf-G2 と他の RPC 実装系との比較を行う．

5.1 GridRPC と他のプログラミングモデル，ツールとの比較

文献 1)，2) では，種々の Grid プログラミングモデルを紹介し，それらの特質について論じている．それらの一部は上記目的の実現に利用可能である．本節では，これらのモデルやツールについて言及する．

まず，HPF，OpenMP，Linda，MPI，Thread，CORBA 等，既存の並列，分散処理用プログラミングモデルを適用することが考えられる．これらのモデルを用いてプログラムを開発することは，すでにこれらのモデルが多くのアプリケーションプログラマにとって馴染みとなっていることからプログラム開発の際の障壁が低いという利点がある．しかし，これらのモデルに基づいて開発されたプログラムをそのままグリッド環境上で実行することは困難である．Grid 環境でのプログラムの実行には，これらのモデルの処理系が本来想定していない非均質性，環境の動的な変動への対処や耐故障性，セキュリティといった特質を求められるからである．

いくつかの研究では，これら既存のモデルを Grid 環境に適応させるという試みを行っている．たとえば，CORBA CoG Kit [22] は，CORBA 環境と Grid 環境を統合し，CORBA 環境から Grid 環境の種々の資源を利用できる一般的な枠組みを提供している．そのために，CORBA CoG Kit では，Globus Toolkit における GRAM，MDS 等のコンポーネントを CORBA のサービスとしてラップし，クライアントから参照可能としている．提供されているラッパは，GRAM サービス，GASS サービスという形で Globus Toolkit のコンポーネントと 1 対 1 に対応している．したがって，Grid 上でのタスク並列処理等への適用を考えた場合，利用者は Globus Toolkit の機能を理解したうえで CORBA クライアントプログラムを開発する必要がある．この点が実装の詳細を隠蔽し GridRPC のセマンティクスを提供している Ninf-G2 と異なっている．

別の例としては，MPI の Grid 環境への適用を試みている一連の研究がある [23)〜25)]．たとえば MPICH-G2 では，Globus 上に MPI のインタフェースを構築し，さらに通信コストの高い集合型通信の最適化を図ることで，既存の MPI プログラムを Grid 環境上で効率的に動作可能としている．タスク並列処理への MPI の適用を考えた場合，動的な環境変化に対する MPI の対応の弱さが問題となる．一般に，MPI では初期化関数実行時にすべての計算資源上にプロセスを起動し，それらの間での接続を完了した後に処理を開始する．しかし，複数の利用者により資源を共有されている Grid 環境上ではつねに対象とする資源を利用できるとは限らない．その場合，MPI プログラムは処理を開始することができない．したがって，coallocation の問題をつねに意識しなければならない．GridRPC の場合，動的に遠隔実行プログラムを起動するため柔軟な処理が可能である．

耐故障性という観点からも，MPI プログラムは柔軟な処理を実現することが困難である．チェックポイント機構を導入したり，メッセージの複製を保持したりすることにより MPI の耐故障性を向上しようとするいくつかの試み[38),39)] が報告されているが，現在の MPI の仕様ではクラスタやネットワークの障害により一部のプロセスが動作しなくなった場合の復旧が困難である．GridRPC の場合，関数ハンドルの作成失敗やハートビート機能の利用により，アプリケーションの耐故障性を高めることが可能になっている．

タスク並列処理の実現という観点から考えると，Globus Toolkit が提供しているコマンドをスクリプトとしてまとめ，実行するというアプローチも可能である．しかし，このアプローチでは，資源の動的な変化に対応するために Ninf-G2 で実装されているタイムアウト機能やハートビート機能等をスクリプト中で実現する必要があるため，利用者の負担は非常に大きい．

Condor[34)] のようなスケジューリングシステムが提供しているコマンドをスクリプトとしてまとめ，実行することは，スケジューリングシステムが Grid 環境上でのタスクの実行を管理してくれることから，Globus Toolkit のコマンドを直接利用するアプローチより利用者の負担は小さい．さらに，Grid 環境上でのタスク並列処理の支援を目的として開発されたプログラミングツール[26)~28)] を利用すればその負担はさらに小さくなる．これらのツールは Globus Toolkit，Condor，GridRPC 等の Grid ミドルウェア上に構築されており，利用者は Grid 環境上に目的とするプログラムを実装し，GUI あるいは XML ベースのスクリプトにより入出力データ，利用計算機等を規定するだけで，タスク並列処理が実現される．

これらのツールは実行前にすべてのパラメータが定まっている単純なタスク並列処理を想定して構築されており，文献 29) で報告されている事例のように，実行すべきパラメータの値がそれ以前に実行された RPC の結果に基づき動的に定まるような処理を効率的に行うことは難しい．それに対し，Ninf-G2 はより多様なタスク並列処理を想定し，関数レベルでのタスク並列処理を実現可能とするとともに，遠隔実行関数の実行モード（同期，非同期），実行待ち合わせモード（`grpc_wait_or`，`grpc_wait_any`，`grpc_wait_all` 等）を複数提供している．したがって，必要に応じてそれらのモードを使い分けることで柔軟な処理を実現することができる．

P2P 技術を利用したタスク並列処理プログラムの開発も可能である．P2P computing は主として広域に分散した遊休 PC の集約を目的とし，Grid とは独立に発展した処理形態であるが，耐故障性，セキュリティ等に対する考慮がなされており，大量の計算資源を要するタスク並列処理プログラムの実行に適用可能である．実際，いくつかのプロジェクト[30),31)] において，大規模なタスク並列処理の実行が報告されている．しかし，これらのプロジェクトは特定のアプリケーションの実行を目的としたものであり，一般の利用者がタスク並列処理プログラムを開発するための枠組みは提供していない．そのような枠組みの提供を目的とした研究として，XtremWeb[32)] があげられる．しかし，XtremWeb では client と実際にタスクを実行する collaborator 間の入出力データ転送はファイル転送によって実現されており，関数の引数として入出力データの転送を抽象化している GridRPC に比べ，利用者のユーザビリティが低い．この問題を解決するために，XtremWeb 上に RPC を実装する試みが行われている[33)]．

5.2 Ninf-G2 と他の RPC 実装系との比較

1 章で述べたように，Ninf-G2 以外にもいくつかの GridRPC 実装系が存在する．これらと Ninf-G2 の大きな違いはその設計方針にある．

Ninf-G2 は，現在最も頻繁に利用されている Grid 基盤である Globus Toolkit のコンポーネントを組み合わせて RPC の機能を実現している．したがって，Ninf-G2 の利用に際しては，Globus Toolkit を利用するためのサーバ（GRAM Gatekeeper および MDS サーバ）のみを必要とし，ほかに特別なデーモンプロセスを必要としない．また，クライアントとサーバとの通信の際には GSI による認証が行われるように設計されている．そのため，すでに Globus Toolkit を配備している組織が Ninf-G2 を新たに導入する際，ファイアウォールのポートフィルタリングの設定等セキュリティポリシの変更を行う必要がない．このように，Ninf-G2 は標準 Grid 基盤として広く利用されている Globus Toolkit を用いることにより，複数の組織による計算資源の共有が常態となっている Grid 環境において広く利用されるようなシステムを目指して設計されている．

それに対し，NetSolve では遠隔実行プログラムを起動するサーバプロセス，遠隔実行プログラム情報の提供や適当なバックエンド計算機の選択を行う Netsolve agent と呼ばれるプロセスを構成要素としている．DIET では，多数の利用者からのアクセスによる agent のボトルネック化を回避するため agent を階層化させており，さらに多数のプロセスから構成される．これらのプロセスは各々独自のプロトコルで通信する

ため，Grid 環境上でこれらのシステムを動作させようとすると各組織において多数のポートを開ける必要がある等，セキュリティポリシの変更が必要である．

Ninf-G2 と OmniRPC の設計方針の違いは，バックエンド計算機の扱い方にある．Ninf-G2 では RPC の実現において特定のバックエンド計算機上の特定の遠隔実行モジュールを抽象化した関数ハンドルを利用し，利用者自身が実行対象とするバックエンド計算機を指定する形式を採用しているのに対し，OmniRPC ではバックエンド計算機の存在を隠蔽し，クライアントからの RPC 呼び出しにおいて指定される関数名をキーとして，OmniRPC 自身が適当なバックエンド計算機を選択し，そのうえで遠隔実行プログラムを起動する．このことは，OmniRPC が適当なサーバを決定するスケジューリング機構を内部に実装していることを意味する．実際，OmniRPC では，クライアント計算機上で管理されている遠隔実行プログラム情報に基づくラウンドロビンスケジューリング機構が実装されている．これに対し Ninf-G2 では，スケジューリング手法はアプリケーションのロジックに依存することが多いこと，またスケジューリングシステム等，他の Grid ミドルウェアで実装されている類似機能の再開発を避け，より抽象度の高い Grid ミドルウェアのインフラとして機能することを目的として，スケジューリング機能を内部で実装することを避けている．

また，OmniRPC は個人利用を想定した大規模タスク並列処理の実現に重点を置き，すべての遠隔実行モジュール情報はクライアント計算機上で個々に管理される．これに対し Ninf-G2 では Globus Toolkit の標準コンポーネントである MDS に情報を登録することで，複数の利用者による遠隔実行モジュールの共有も可能にしている．

GridRPC 以外に広域分散環境での利用を目的とした RPC 実装系としては，XML-RPC[35] や SOAP[36] の処理系があげられる．これらは http プロトコルを用いて XML 形式のデータを授受するシステムである．これらのシステムと Ninf-G2 との違いは，Ninf-G2 が Grid 環境上における大規模科学技術計算の効率的な実行に重点を置いている点である．

大規模科学技術計算では，行列データ等，浮動小数点形式の大規模データが頻繁に利用される．XML 形式でこれらのデータを授受する場合には，現状ではエンコーディングやデコーディングのコストが高いといった問題が存在することが指摘されている[37]．それに対して，Ninf-G2 ではこのようなデータの効率的な転送を実現するために，バイナリデータ転送機能や一定のストライドごとにデータを授受する機能が実装されている．

6. Ninf-G2 の現状と利用方法

現在，Ninf-G2 は http://ninf.apgrid.org において第 1 版が公開されている．現在のところ，Globus Toolkit Ver.3 の Pre-WS コンポーネントあるいは Ver.2.2 以上が動作する SPARC/Solaris, IA32/Linux（RedHat, Debian），IA64/Linux（RedHat），R10000/IRIX 上で動作が確認されている．システムをインストールするためには，利用対象とするクライアント計算機およびバックエンド計算機上で Globus Toolkit が動作していることを確認し，各計算機上で configure を実行する．実行時には，Globus Toolkit の flavor を指定する必要がある．MDS を利用して遠隔実行プログラムのインタフェース情報を管理する場合には，`${GLOBUS_LOCATION}/etc/grid-info-slapd.conf` ファイルに Ninf-G2 が利用する schema を追加する．

Ninf-G2 を利用したプログラムの開発（既存プログラムの Grid 化作業）は，一般に以下の手順で行われる．

(1) 遠隔呼び出し関数の作成
逐次プログラムにおいて，サーバ上で実行されるべき処理を関数として抽出する．

(2) 遠隔呼び出し関数におけるデータ依存関係の除去
遠隔呼び出し関数を並列実行可能とするため，遠隔呼び出し関数に関連するデータ依存関係を除去する．

(3) GridRPC 関数の挿入
(1)，(2) の作業によって作成した関数とクライアント計算機上で実行される処理を別プログラムとして切り出した後，クライアントプログラム内に GridRPC 関数を挿入する．サーバプログラムに対しては，呼び出される関数のインタフェースを規定する IDL ファイルを作成する．

(4) サーバプログラムのバックエンド計算機上でのコンパイル
両プログラムを各々実行対象計算機上でコンパイルする．Ninf-G2 パッケージにはクライアントプログラムのコンパイルを支援する `ng_cc`，遠隔実行プログラムの生成を支援する IDL コンパイラが提供されており，これらを用いて両プログラムをコンパイルすればよい．また，Ninf-G2 には遠隔実行プログラムを実行対象計算機上に転送して（staging）実行する機能も実装されており，遠隔実行プログラムを 1 台の計算機で一括管理することも可能である．

Ninf-G2 のインストール手法，利用方法，プログラム開発手順等に関する詳細は，マニュアル[19] および文献 11)，20) を参照いただきたい．

7. まとめと今後の課題

複数のクラスタから構成される Grid 環境での大規模シミュレーションに適した GridRPC システム Ninf-G2 の開発および性能評価について述べた．

Ninf-G2 は，関数ハンドル同時生成機能やリモートオブジェクトのメカニズムを実装することで，遠隔手続き呼び出しにともなう起動コストや通信コストの低減を図るとともに，ハートビート機能や関数ハンドル作成タイムアウト機能，サーバ属性の個別設定機能を提供することで，非均質，不安定で動的に変化する Grid 環境への対応を図っている．

4 サイトに配置されたクラスタを用いて，処理粒度の異なる 3 種類の気象シミュレーションの実行性能を測定した結果，Ninf-G2 を用いることにより 200 を超えるプロセッサ上で効率的にシミュレーションを実行できること，実行時間が十数秒という比較的粒度の小さいプログラムでも，クライアントコンポーネントを多重化するといった工夫をすることで，低オーバヘッドで実行可能であることが分かった．この程度の粒度の処理が Grid 上で効率的に実行可能であることを示せたことは，これまで比較的粒度の大きいアプリケーションに限定されていた Grid アプリケーションの裾野を拡大する効果があると考えられる．

また，日米韓にわたって構築された Grid テストベッド上で 500 プロセッサを利用した大規模シミュレーションを実行できた．これにより，Ninf-G2 が数百プロセッサ規模の計算資源を安定して制御できる能力を持つことを示すことができた．

さらに，Ninf-G，Ninf-G2 を利用して同一アプリケーションを Ninf 化し，実行性能を測定，比較することで，Ninf-G2 において新たに実装された機能がタスク並列処理の効率的な実行に有効であることを示した．

今後の課題は，2003 年度末に導入が予定されている AIST スーパクラスタや TeraGrid 等をテストベッドに加え，広域に分散した数千プロセッサ規模の Grid 環境上で性能評価を行うことである．

謝辞 本研究に際し，順圧 S-model プログラムをご提供いただいた筑波大学田中博助教授に感謝いたします．

本研究は，Asia-Pacific Grid（ApGrid）[13] および Pacific Rim Applications and Grid Middleware Assembly（PRAGMA）[14] における研究活動の一環として行われました．ApGrid および PRAGMA の参加研究機関，特に実験に計算資源を提供いただいた KISTI, KU，TITECH に感謝いたします．

また，SC2003 における実験に協力していただいた，TeraGrid Exective Commitee メンバ諸氏および Tera Grid ヘルプチームに感謝いたします．

また，アジアー太平洋地域におけるネットワーク環境設定にご尽力いただいた Asia Pacific Advanced Network（APAN）[15] 参加機関に感謝いたします．

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）」により遂行されました．

参 考 文 献

1) Lee, C. and Talia, D.: Grid Programming Models: Current Tools, Issues and Directions, *Grid Computing: Making the Global Infrastructure a Reality*, pp.555–578 (2003).
2) Lee, C., Matsuoka, S., Talia, D., Sussman, A., Mueller, M., Allen, G. and Saltz, J.: A Grid Programming Primer, *GWD-I, GGF Advanced Programming Models Research Group* (2001).
3) Catlett, C.: Standards for Grid Computing: Global Grid Forum, *Journal of Grid Computing*, Vol.1, No.1, pp.3–7 (2003).
4) Seymour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C. and Casanova, H.: Overview of GridRPC: A Remote procedure Call API for Grid Computing, Lecture Notes on Computer Science, Vol.2536, pp.274–278 (2002).
5) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T. and Matsuoka, S.: Ninf-G: A Reference Implementation of RPC-based Programmin Middleware for Grid Computing, *Journal of Grid Computing*, Vol.1, No.1, pp.41–51 (2003).
6) Casanova, H. and Dongarra, J.: NetSolve: A Network Server for Solving Computational Science Problems, *Proc. Supercomputing' 96* (1996).
7) Caron, E., Deprez, F., Lombard, F., Nicod, J.-M., Quinson, M. and Suter, F.: A Scalable Approach to Network Enabled Servers, *Proc. 8th International EuroPar Conference*, LNCS Vol.2400, pp.907–910 (2002).
8) 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 GridRPC 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.SIG 11(ACS

3), pp.34–45 (2003).
9) Tanaka, H.-L. and Nohara, D.: A Study of Deterministic Predictability for the Barotropic component of the Atmosphere, section A, *Science Reports of the Institute of Geoscience*, Vol.22, pp.1–21, University of Tsukuba (2001).
10) 合田憲人，中村心至：細粒度最適化問題アプリケーションのグリッドテストベッド上への実装，HPCS2004 論文集，pp.75–76 (2003).
11) 武宮 博，首藤一幸，田中良夫，関口智嗣：Grid 環境上における気象予報シミュレーションシステムの構築，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(ACS 3), pp.23–33 (2003).
12) 田中良夫，中田秀基，朝生正人，関口智嗣：Ninf-G2: 大規模環境での利用に即した高機能，高性能 GridRPC システム，情報処理学会研究報告，Vol.2003, No.83, pp.65–70 (2003).
13) ApGrid: Asia-Pacific Network. http://www.apgrid.org/
14) PRAGMA: Pacific Rim Applications and Grid Middleware Assemble. http://www.pragma-grid.net/
15) APAN: Asia-Pacific Advanced Network. http://www.apan.net/
16) TeraGrid. http://www.teragrid.org/
17) 中島佳宏，佐藤三久，後藤仁志，朴 泰祐，高橋大介：CONFLEX-G：OmniRPC によるグリッド環境上での分子立体配座探索，HPCS 論文集，pp.95–102 (2003).
18) 朴 泰祐，佐藤三久，小沼賢治，牧野淳一郎，須佐 元，高橋大介，梅村雅之：HMCS-G：グリッド環境における計算宇宙物理のためのハイブリッド計算システム，情報処理学会論文誌，Vol.44, No.SIG 11, pp.1–13 (2003).
19) http://ninf.apgrid.org/documents/ng2-manual/user-manual.html
20) 田中良夫，中田秀基，平野基孝，佐藤三久，関口智嗣：Globus による GridRPC システムの実装と評価，情報処理学会 HPC 研究会，Vol.2001, No.77, pp.165–170 (2001).
21) 中田秀基，田中良夫，松岡 聡，関口智嗣：GridRPC を用いたタスクファーミング API の試作，情報処理学会 HPC 研究会，Vol.2003, No.96, pp.61–66 (2003).
22) Lasewski, G., Parashar, M., Verma, S., Gawor, J., Keahey, K. and Rehn, N.: A CORBA Commodity Grid Kit, *Concurrency and Computation: Practice and Experience* (2001).
23) Karonis, N., Toonen, B. and Foster, I.: MPICH-G2: A Grid-enabled Implementation of the Message Passing Interface, *Journal of Parallel and Distributed Computing* (2003).
24) Gabriel, E., Resch, M. and Ruhle, R.: Implementing MPI with Optimized Algorithms for Metacomputing, *Proc. MPIDC'99*, pp.31–41 (1999).
25) Imamura, T., Tsujita, Y., Koide, H. and Takemiya, H.: An Architecture of Stampi: MPI library on a Cluster of Parallel Computers, Lecture Notes on Computer Science, Vol.1908, pp.200–207 (2000).
26) Casanova, H. and Berman, F.: Parameter Sweeps on the Grid with APST, *Concurrency, Practice and Experience*, Vol.1, No.1 (2002).
27) Abramson, D., Giddy, J. and Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, *Proc. IPDPS*, pp.520–528 (2000).
28) Yarrow, M., Mccann, K., Biswas, R. and Wijngaart, R.V.: An Advanced User Interface Approach for Complex Parameter Study Process Specification on the Information Power Grid, *Proc. Grid2000* (2000).
29) Ikegami, T., Takemiya, H., Nagashima, U., Tanaka, Y. and Sekiguchi, S.：Grid：広域分散並列処理環境での高精度分子シミュレーション—C20 分子のレプリカ交換モンテカルロ，情報処理学会論文誌，Vol.44, No.SIG 11, pp.14–22 (2003).
30) SETI@home (2001). http://setiathome.ssl.berkeley.edu
31) Great Internet Mersenne Prime search (1997). http://www.mersenne.org
32) Fedak, G., Germain, C., Neri, B. and Cappello, F.: XtremWeb: A Generic Global Computing System, *Proc. CCGrid2001, workshop on Global Computing on Personal Devices* (2001).
33) Djilali, S.: P2P-RPC: Programming Scientific Applications on Peer-to-peer Systems with Remote Procedure Call, *Proc. GP2PC2003* (2003).
34) Livny, M., Basney, J., Raman, R. and Tannenbaum, T.: Mechanisms for High Throughput Computing, *SPEEDUP Journal*, Vol.11, No.1 (1997).
35) XML-RPC. http://www.xml-rpc.com/
36) Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/soap12
37) Govindaraju, M., Slominski, A., Choppla, V., Bramley, R. and Gannon, D.: Requirements for and Evaluation of RMI Protocols for Scientific Computing, *Proc. SC'2000* (2000).
38) Agbaria, A. and Friedman, R.: Starfish: Fault-Tolerant Dynamic MPI Programs on Clusters of workstations, *Proc. 8th IEEE International Symposium on High Performance Distributed Computing* (1999).

39) Gabriel, E., Fagg, G., Bukovsky, A., Angskun, T. and Dongarra, J.: A Fault-Tolerant Communication Library for Grid Environments, *Proc. 17th Annual ACM International Conference on Supercomputing* (*ICS'03*), International Workshop on Grid Computing and e-Science (2003).

(平成 16 年 1 月 31 日受付)
(平成 16 年 5 月 29 日採録)

武宮　博（正会員）

日立東日本ソリューションズ（株）公共ソリューション本部サイエンス&テクノロジーセンター主任研究員．昭和 61 年東北大学大学院理学研究科天文学博士前期課程修了．平成元年同博士後期課程中退．同年日立東日本ソリューションズ（株）入社．平成 14 年より産業技術総合研究所グリッド研究センターへ派遣．

中田　秀基（正会員）

昭和 42 年生．平成 2 年東京大学工学部精密機械工学科卒業．平成 7 年同大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．同年電子技術総合研究所研究官．平成 13 年独立行政法人産業技術総合研究所に改組．現在同所グリッド研究センター主任研究官．平成 13 年より東京工業大学客員助教授を兼務．グローバルコンピューティング，並列実行環境に関する研究に従事．

田中　良夫（正会員）

昭和 40 年生．平成 7 年慶應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．平成 8 年技術研究組合新情報処理開発機構入所．平成 12 年通産省電子技術総合研究所入所．平成 13 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．

関口　智嗣（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 59 年筑波大学大学院理工学研究科修了．同年電子技術総合研究所入所．情報処理アーキテクチャ部主任研究官．以来，データ駆動型スーパーコンピュータ SIGMA-1 の開発等の研究に従事．平成 13 年独立行政法人産業技術総合研究所に改組．平成 14 年 1 月より同所グリッド研究センターセンター長．並列数値アルゴリズム，計算機性能評価技術，グリッドコンピューティングに興味を持つ．市村賞受賞．日本応用数理学会，ソフトウェア科学会，SIAM，IEEE 各会員．