2004－HPC－99　(7)
2004／7／30

Ninf-G2の性能評価：科学技術計算における事例

谷村勇輔[†]　池上努[†]　中田秀基[†,††]
田中良夫[†]　関口智嗣[†]

Ninf-G は，グリッド環境を利用する計算アプリケーションのための上位のミドルウェアである．本稿では量子化学計算の 1 つである TDDFT を例にとり，科学技術計算のために Ninf-G が備えているべき機能を検証した．グリッド化に際して，Ninf-G の Version 2 で加えられたリモート・オブジェクトを活用し，かつ長時間実行のための耐故障性をもつモデルを実装した．評価実験では通信データの暗号化と同時転送における性能を議論し，さらに 4 サイト 64CPU を用いた長時間実行を通して，障害を適切に検知できるなど Ninf-G の実用性を明らかにした．

Evaluation of Ninf-G Version 2: A Case Study in Implementing Scientific Application

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

Ninf-G is a higher level middleware for the Grid-enabled computing application. In this paper, TDDFT, an application of the computational quantum chemistry, is adopted as an example of the scientific application which can be implemented with Ninf-G and also used for evaluation of Ninf-G. The remote-object is useful for implementation of TDDFT and a fault torerant model is considered in this application program. Through performance measurement of parallel or encrypted data transfer, and observation of long-run stability, practical usability of Ninf-G is illustrated.

1. はじめに

近年，大規模な科学技術計算を行うために，膨大な資源を利用できる可能性のあるグリッド環境が注目されている．しかしながら，グリッド環境は単一のシステムである計算機と異なり，地域的に離れた複数のシステムから構成されるため，そこで動作するソフトウェアは様々な問題を考慮しなければならない．広く注目を集めている Globus Toolkit[1] は，それらを解決する基盤を提供するが，低レベルのインタフェースしか提供しない．必然的に，アプリケーション開発にはさらに上位のミドルウェアが必要である．Ninf-G[2] はそのような上位のミドルウェアであり，グリッド向けに科学計算アプリケーションを実行する枠組とアプリケーション開発のための単純な API を提供する．Ninf-G が提供するプログラミングモデルは，RPC の仕組みをグリッドに拡張した GridRPC[3] である．GridRPC では，アプリケーション・プログラムが非同期に RPC を発行することで，複数の計算機を使ってタスクを並列実行できる．すなわち，パラメータ探索型やマスタースレーブ型の並列計算であれば，Ninf-G を使って従来のアルゴリズムを変更することなくグリッド・アプリケーションとして実装可能である．

Ninf-G では，これまでレプリカ交換モンテカルロ[4] や気象予測シミュレーション[5] がアプリケーションとして実装され，システムの評価がなされてきた．気象予測シミュレーションでは 4 サイト 500 プロセッサでの実験が行われ[6]，基本性能だけでなくスケーラビリティについても検討がなされた．本研究では，さらに実用レベルで利用できるか評価するために，科学計算アプリケーションのユーザからの視点を採り入れる．例題として用いる量子化学計算の 1 つ TDDFT (Time-Dependent Density Functional Theory)[7] は大規模計算を必要としており，独立性の高い並列処理を行うアルゴリズムを適用できる．また，データ転送

† 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

量も比較的大きく，長時間実行を伴うことから Ninf-G の耐故障性や安定性に関する議論を行うことができる．本稿ではこれらの背景から，Ninf-G を利用するプログラムの記述性，並列実行の性能，長時間実行での安定性，セキュリティの観点から Ninf-G がもつ機能を明らかにし，科学計算分野における実用性を検討する．

2. Ninf-G と科学技術計算

Ninf-G は産総研，東工大らによって開発が進められてきたグリッドのミドルウェアである．グリッドのアプリケーション開発を容易にする GridRPC の参照実装でもあり，Globus Toolkit Version 2 を基盤として利用する．すなわち，GSI によるセキュリティ機構やクラスタ計算機へのジョブ投入機能を備え，それらを利用して，アジア太平洋地域のテストベッドにおける性能試験がなされている．

2.1 Ninf-G アプリケーションの開発

Ninf-G のアプリケーションは，クライアントプログラムとスタブプログラムからなる．それぞれ Ninf-G のライブラリとリンクされるが，個別に作成される．以下に，アプリケーション開発の流れを述べる．

スタブとは，リモートの計算機に用意された Ninf-G の実行プログラムである．**図 1** に示すように，スタブはクライアントプログラムに記述する GridRPC の「呼び出し API」により起動され，決められたタスクをこなす．スタブのインターフェースは IDL（Interface Description Language）で記述され，IDL をもとに生成されたスタブのテンプレートプログラムと，タスク内容が記されたプログラムがリンクされてスタブの実行ファイルが作られる．スタブはステージング機能を用いて実行時に転送できるが，計算機のアーキテクチャやライブラリのバージョンが異なる場合，利用者がリモートにあらかじめ用意しておく必要がある．

クライアントプログラムでは，GridRPC の「呼び出し API」が記述される．「呼び出し API」は，スタブの IDL と対応した入力，出力データを引数として持つ．同期呼び出し，非同期呼び出しが可能であり，必要に応じて呼び出しをキャンセルできる．これらは並列実行のアルゴリズムに合うよう，アプリケーション開発者が考えて記述することになる．

2.2 科学計算アプリケーションからの要求事項

科学計算アプリケーションには様々なものが存在し，その特徴は個々のアルゴリズムやプログラムに依存する．その中で Ninf-G が対象とするアプリケーションは，パラメータ探索やマスタースレーブ型の並列モデルで見られる独立性の高いタスクの並列処理を行うものや，遠隔にインストールされている優れた数値計算ライブラリを別の計算機から利用するものである．実際，少なくない数の科学計算アプリケーションは前者のモデルを採用でき，かつ遠隔の計算資源を利用することによるオーバーヘッドに見合う計算量と通信量のバランスを確保できる．本研究において，例題として用いる TTDFT も前者のモデルでグリッド化を行う．

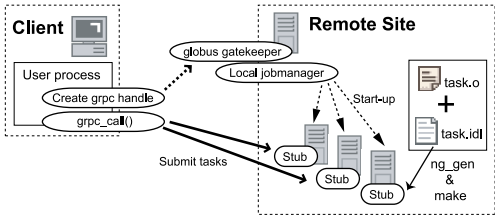

図 1 Ninf-G の概要

一方，Ninf-G が想定しているアプリケーションのユーザにとっては，Ninf-G を用いた開発を少ない労力で行うことができ，プログラムが高速にかつ安定して動作することが望ましい．つまり，アプリケーションをグリッド化するにあたり，Ninf-G に求められる項目として以下が挙げられる．

(1) 既存プログラムの修正を最小限に抑える記述性
(2) 比較的大きい入出力データの転送性能
(3) 計算規模を大きくしたの時のスケーラビリティ
(4) 長時間実行に対する安定性および障害検知と回復機能
(5) 計算内容やデータを隠蔽できる高度なセキュリティ機構

我々は TDDFT をサンプルアプリケーションとして Ninf-G を用いてグリッド化し，グリッド上で長時間にわたる実験を行い，上記の項目について Ninf-G の機能および性能の評価を行った．次章では TDDFT について述べ，次々章では Ninf-G の機能を活用しながらどのように TDDFT が実装されたかを説明する．

3. TDDFT 方程式

近年注目されているナノ材料の分野では分子レベルのシミュレーションが活用されている．これらシミュレーションで用いるポテンシャル関数の精度を上げるため，シミュレーションの過程で電子状態を直接扱う手法も用いられるようになってきた．特に系の光学的な性質を調べるには，多電子励起状態をあらわに取り扱う必要があるので，量子力学に基づいて電子状態を記述しなければならない．こうした手法の 1 つとして TDDFT が提案されている[7]．

TDDFT では時間に依存する N 電子波動関数を

Slater 行列式を用いて

$$|\Psi\rangle = |\psi_1\psi_2\cdots\psi_N\rangle \quad (1)$$

のように近似する．ψ_i は電子の占有軌道を記述する一電子関数である．変分原理を用いると，$|\Psi\rangle$ の時間発展方程式は以下の N 本の方程式に分解される．

$$i\frac{\partial}{\partial t}\psi_i = \left(-\frac{1}{2}\nabla^2 + V_{\rm ion} + V_{\rm H} + V_{\rm xc}\right)\psi_i \quad (2)$$

ここで $V_{\rm ion}$ は電子と原子核の，$V_{\rm H}$ は電子間のクーロン相互作用を記述する項で，後者は電子密度 $n(r) = \langle\Psi|\Psi\rangle_{2,\ldots,N}$ の汎関数である．最後の $V_{\rm xc}$ は，交換相互作用と電子相関の効果を記述する量子力学的な項で，これも $n(r)$ の汎関数である．今回は $V_{\rm xc}$ として局所密度近似に基づく表式を用いた．

一般に TDDFT 方程式は、式 (2) にさらに線形応答近似を施して線形方程式に還元し，周波数領域の問題として解かれることが多い．一方，今回対象にしたプログラムは同方程式を直接，数値積分するアプローチを取っているため，並列処理に適している．

4. Ninf-G を用いた TDDFT の並列化

TDDFT の計算は，入力データの読み込みとパラメータの設定後，1) 占有軌道の時間発展（$\psi_i(t) \rightarrow \psi_i(t+\delta t)$）計算，2)$\psi_i$ を用いた $n(r)$，$V_{\rm H}$，$V_{\rm xc}$ の計算を時間ステップを進めながら繰り返す．中規模以上の分子では，1) は総計算量の 5 割以上を占めるホットスポットであるが，占有軌道毎に独立に並列計算できる．軌道数は分子内の電子数に比例するため，大きな分子では並列度も高くなる．

本実装モデルでは，**図 2** に示すように，クライアントから複数のスタブに対してタスクの実行を非同期に依頼し，ψ_i の時間発展を並列計算する．ただし，時間ステップの更新時に，依頼した全タスクの完了を待つ同期が必要である．また，各占有軌道に対して時間発展の計算を受けもつスタブを，計算の都度，動的に割りつけるアプローチをとり，タスクとスタブの対応を自由に決定できるようにした．

並列タスクの依頼は，**図 3** に示すキューのモデルをアプリケーションレベルで実装した．各サイトで起動したスタブは Idle，Busy，Down の 3 つの状態に分けて管理される，Idle キューにあるスタブに対してタスクが依頼され，そのスタブは Busy 状態に移される．Idle キューにスタブがなくなると，Busy 状態のスタブが計算を終了するまで待機し，Idle キューに戻ってきたスタブに対して新たなタスクの実行を依頼するアルゴリズムである．エラーを発生したスタブは Down 状態に移される．このタスクの並列実行にはファーミ

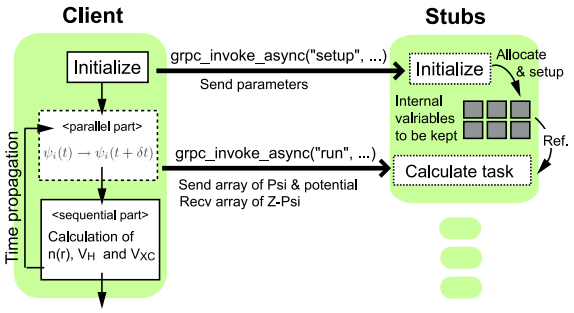

図 2 グリッド化された TDDFT の計算モデル

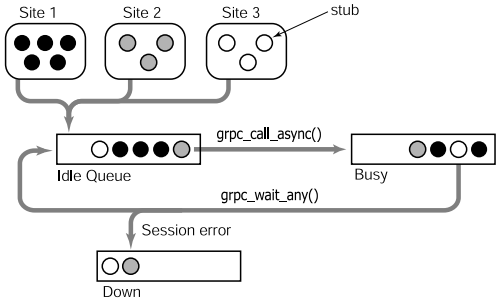

図 3 並列実行のためのキューモデル

ング[8] を利用することもできるが，今回はリモート・オブジェクトを用い，耐故障性についても考えるためファーミング API は利用しなかった．

4.1 リモート・オブジェクトを用いた記述

TDDFT のグリッド化に際して，時間発展の計算をスタブで行うためには全スタブで初期化処理が必要となる．この初期化時に作成される配列データは後のメイン計算にて参照されるが，メイン計算を繰り返し行うにあたり，データがスタブ側で保持されることが望ましい．しかし，従来の関数ハンドルでは，スタブが呼び出しセッションをまたがってデータを保持することが保証されなかったため，クライアントに毎回データを送信するかファイルに保存しなければならなかった．前者は通信のオーバーヘッドを招き，後者は開発者の手間を増やすことになる．

この問題は TDDFT 以外のアプリケーションでも起こりうると考えられ，効率的な通信とプログラムの記述性の向上に対応すべく，Ninf-G2 ではリモート・オブジェクトの機能が加えられた．リモート・オブジェクトは，スタブを複数のメソッドを持つことができるオブジェクトと定義したもので，内部にデータを保持することができる．これを用いて TDDFT を実装すると，図 2 のように HPsi_setup() において時間発展計算で参照するデータを初期化し，HPsi_run() においてクライアントから動的に渡す引数データと合わせ

て，時間発展計算を行うようプログラム可能となる．

リモート・オブジェクトの API は．同期呼び出しが grpc_invoke()，非同期呼び出しが grpc_invoke_async() となっている．これは呼び出しの際にメソッド名を指定する以外，従来の関数ハンドルとの違いはない．

4.2 長時間実行に要求される機能

TDDFT は時間ステップの更新計算を数千回必要とし，問題規模に応じて数日の計算を行う．しかし，グリッド環境では長い間スタブを起動して多数のタスクを処理させる過程で，ネットワークが切断されたり，計算機の停止が起きることは避けがたい．こうした障害に対して，アプリケーションの実行をやり直すことなく継続できることが望まれる．ミドルウェアとして自動復旧のサポートも考えられるが，Ninf-G では必要に応じてアプリケーションが耐故障性を実装できるように設計され，適切に障害や異常を検知してアプリケーションにエラーを返す機能が実装されている．これを踏まえて，本 TDDFT プログラムではタスクとスタブの対応を動的に決定するモデルを採用し，異常スタブを検出すると，それを利用可能リストから取り除くように実装した．

障害の要因は様々であるが，grpc_call_async() を発行してタスクの実行依頼をする際にスタブが動いていないとスタブエラーが返る．データ転送中にスタブとの接続が切断されると，grpc_wait_any() はセッションエラーを得る．しかし，状況によってはクライアント側で障害を検出できないことも考えられ，そのために Ninf-G2 ではハートビート機能が提供された．これは，ハートビートのタイムアウトを通じてスタブの異常を検出し，grpc_wait_any() にセッションエラーを返す仕組みである．本 TDDFT プログラムでは，以降の実験においてハートビートを 60 秒から 180 秒の間隔で送信するように設定し，障害検知に利用した．

一方，取り除いたスタブを再利用するために，サイト内の全てのスタブが障害リストに入った場合，該当サイトの全スタブを再起動するように TDDFT のクライアントプログラムを実装した．しかし，障害が長引く場合，この再起動も失敗する．この時，ネットワークや計算機の状況によってはハンドル生成が失敗にならず，起動したはずのスタブからの応答を永久に待ってしまうこともある．これに対しても Ninf-G2 では job_startTimeout 属性を設けて，スタブの応答待ちをタイムアウトさせる機能を提供している．本 TDDFT プログラムでもこれを利用した．

障害以外では，Globus のプロキシ証明書の更新問題がある．現状，証明書の作成時に有効期限を十分に大きく設定すれば，アプリケーションを長時間動作させることができる．しかし，セキュリティ上好まれないことから，ユーザが定期的に証明書を更新することが望ましい．更新をリモートにも伝えるため，Ninf-G2 では refresh_credential の属性を提供しており，本 TDDFT プログラムでもこれを利用した．

表 1 実験環境

Site	#CPU	System	Throughput
AIST	34	PentiumIII 1.4GHz	116 MB/s
KISTI[1]	16	Pentium4 1.7GHz	0.28 MB/s
SDSC[2]	12	Xeon 2.4GHz	0.044 MB/s
KU[3]	4	Athlon 1GHz	0.050 MB/s
DU[4]	1	PentiumIII 1.1GHz	0.79 MB/s

[1] Korea Institute of Science and Technology Information
[2] San Diego Supercomputer Center
[3] Kasetsart University
[4] Doshisha University

5. 評 価 実 験

本研究では前章で述べたように実装した TDDFT を用いて，Ninf-G を評価した．本稿では，アプリケーションによっては重要である経路を暗号化した際の通信性能，Version 2 で加えられた引数の同時転送の性能，4 サイト 64CPU を用いて 18 時間の連続実行を行った結果を示す．実験環境は**表 1** の通りである．スループットは，AIST の計算機から 1MB のメッセージを TCP で送信することで計測した．一方，UDP で計測すると，AIST から KU へ 1Mbits より大きなメッセージを送ると次第にパケットの損失割合が大きくなり，2Mbits で 4.7%，4Mbits で 52%，8Mbits で 75%であった. これは KU 以外のサイトとの計測に比べて特別であったことに注意されたい．これに対して，対象とした分子はクライアントからスタブへ 4.87MB のデータを入力し，出力として 3.25MB を得る．

5.1 通信経路の暗号化

表 2 は，通信経路を暗号化した時の入力データの転送時間を暗号化しない時と比較している．LAN 環境の実験は AIST のクラスタ内で行い，インターネット環境の実験は DU をクライアントとして AIST のクラスタに対してタスクの実行を依頼する形で行った.Plain は暗号化がない場合，GSI と SSL では 128bit の RC4 で暗号化されている．

結果より，データサイズが 2.5KB と小さい場合には，暗号化による転送速度の劣化はほとんど見られない．しかし，転送量が増加すると著しく性能が低下した．暗号化により増加する転送量は 20B のダイジェ

表 2 経路を暗号化した際の入力データの転送時間 [sec]

	LAN			Internet		
	Plain	GSI	SSL	Plain	GSI	SSL
2.5KB	0.038	0.041	0.038	0.060	0.059	0.059
2.5MB	0.16	0.85	0.87	0.95	2.7	2.7
4.87MB	0.23	1.3	1.3	1.5	4.2	3.2

表 3 並列タスクの実行時間（walltime [sec]）

	LAN		Internet	
# stubs	Wait	Nowait	Wait	Nowait
1	70.48	70.48	72.52	72.52
16	74.20	73.56	94.59	90.27
32	79.10	76.76	118.57	109.93

スト分であるため，性能を低下させた要因は暗号化と復号化に要した時間と考えられる．

5.2 非同期呼び出しにおける引数の同時転送

従来の grpc_call_async() ではタスクの実行を非同期に依頼しても，その引数の転送が完了するまで GridRPC 関数が返ることはなかった．Ninf-G2 では，transfer_argument 属性を nowait に設定することで転送完了を待たずに関数が返るため，これを利用して連続的にタスクを依頼し，呼び出しの遅延を隠蔽できる．本稿ではこれを同時転送と呼んでいる．引数の同時転送を有効にした時と無効にした時の並列タスクの実行時間を，前項と同様に，LAN 環境とインターネット環境で計測した．引数のサイズは入出力合わせて 8.12MB とし，入力と出力の転送が互いに重ならないよう，各タスクの実行時間を 70 秒と設定した．

表 3 に示すように，同時転送（nowait）は呼び出しの遅延を隠蔽し，結果として逐次転送（wait）より速くタスクを処理できた．スタブ数が多い時や，ネットワークのホップ数が多くなり通信のレイテンシが大きくなるインターネット環境では，同時転送により大きな時間を削減できた．この結果は，TDDFT のようにタスクを並列実行するアプリケーションにとって，引数の同時転送が有効であることを示している．

5.3 長時間の実行

Ninf-G が障害を適切に検出し，実装モデルが障害に対応しながら長時間実行可能であるか検証するために，本 TDDFT プログラムを日本時間午後 10 時から翌 16 時までの 18 時間にわたり実行した．クライアントは AIST の計算機で実行し，DU を除く 4 サイト 64CPU でスタブを実行した．また，5.1，5.2 節を踏まえて通信性能を重視して経路の暗号化はせず，同時転送を有効にして実験を行った．シミュレーションに

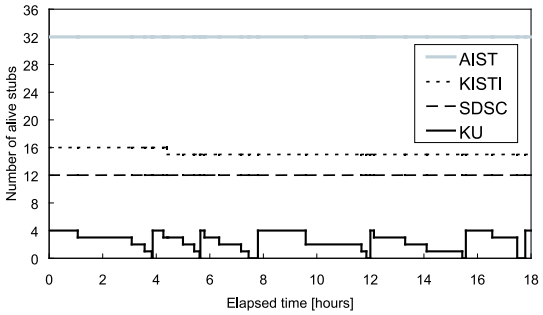

図 4 サイト別生存スタブ数の変化

用いた分子は 122 個の占有軌道を有するため，122 タスクに分割して並列処理した．18 時間後，プログラムが計算した時間ステップの更新ループは 330 であった．ただし，現状のプログラムは計算精度や通信データのチューニングが十分でないため，グリッドを用いた際の並列処理の効率は検討しない．

18 時間の実験中，約 1800 タスクを処理する毎に，生存するスタブ数を記録した結果が**図 4** である．KU で動くスタブは途中で異常終了する回数が大きかった．原因はいずれもクライアントとスタブ間の通信の寸断であり，Ninf-G はセッションエラーを返した．また 1 度に数スタブが終了する場合と，あるスタブだけが終了する場合があった．KU の 4 スタブが全て終了すると，実装した復旧アルゴリズムによりスタブが再起動された．ただし，今回の実験ではスタブの再起動にも失敗するような中長期的な障害は発生しなかった．

同様に，サイト別の処理タスク数を**図 5** に示す．タスクの実行は生存する全スタブに対して，まず 1 つ依頼し，その後は処理の終えたスタブに対して依頼するアルゴリズムである．今回は通信コストの問題から，半分以上のタスクはローカル資源である AIST のスタブで処理され，サイト間の通信状況にも依存して若干の変化はあるものの，KISTI や SDSC ではほぼ一定量のタスクが処理された．KU は 122 タスクのうち，せいぜい 1〜2 個程度しか割り当てられなかったが，データ通信に失敗してスタブが異常終了することも多く，処理タスク数は一定しなかった．

図 6 は，サイト別の入力データの転送速度である．SDSC へのネットワーク・スループットはそれほど高くないが KU に比べて安定しており，スタブが終了することもなかった．図 4 と図 6 を合わせると，KU の生存スタブ数が少ない時に KU への転送速度も高いことが分かる．これは KU への転送過多を表した結果であるとともに，ネットワークのバンド幅の不足や不安定さに起因する問題が考えられ，詳細なネットワーク

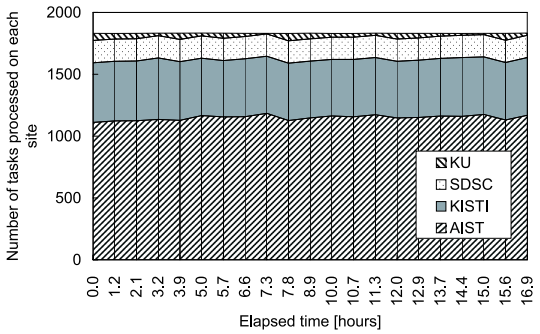

図 5 時間毎のサイト別処理タスク数

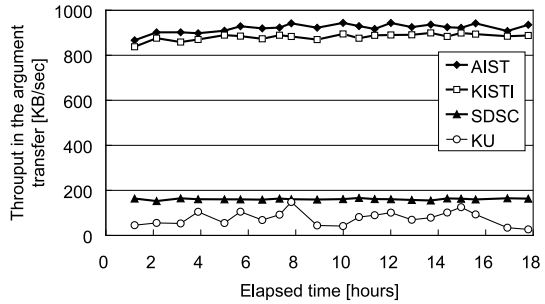

図 6 時間毎のサイト別転送速度

の調査が必要である．

これらの実験結果より，Ninf-G2 は 1 日程度の長時間実行に耐えることができ，ネットワークの切断をアプリケーションに通知し，アプリケーションがスタブを再起動しながら計算を継続できることを確認した．本実験ではネットワーク障害についての検証だけであったが，Ninf-G2 では，クライアントとスタブのセッションを監視することでスタブの状態を知るため，計算機の障害であっても同様の対応が可能であると考える．一方，本実験で経験した障害はネットワークの一時的な過負荷による通信の途絶が主であった．これに対して，スタブを再起動させるのではなく，Ninf-G が再接続で対応できれば復旧がスムーズに行われ，アプリケーション側での対応が不要になると考える．

6. ま と め

本稿では，科学計算アプリケーションの 1 つとして TDDFT を用いて Ninf-G Version 2 を評価した．科学計算アプリケーションにとって有用であると考えられる評価項目において，Ninf-G Version 2 の機能および性能を示すことができた．今後は計算精度を上げたり，電子数を増やしたりすることで計算規模を拡大し，大規模環境でさらに長期間の実験を行うなど，より実用に近い検証が必要だと思われる．

謝辞 本研究を行うにあたり，TDDFT のプログラムを提供して下さった信定（分子研），矢花（筑波大）両氏，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 KISTI，KU，SDSC，同志社大に感謝いたします．

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) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *Supercomputing Applications and High Performance Computing*, Vol. 11, No. 2, pp. 115–128 (1997).
2) 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).
3) 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).
4) 池上努, 武宮博, 長嶋雲兵, 田中良夫, 関口智嗣: 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–C_{20} 分子のレプリカ交換モンテカルロ, 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. 14–22 (2003).
5) Takemiya, H., Shudo, K., Tanaka, Y. and Sekiguchi, S.: Constructing Grid Applications Using Standard Grid Middleware, *Grid Computing*, Vol. 1, pp. 117–131.
6) 武宮博, 田中良夫, 中田秀基, 関口智嗣: 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, 情報処理学会論文誌 コンピューティングシステム (2004).
7) Yabana, K. and Bertsch, G. F.: Time-Dependent Local-Density Approximation in Real Time: Application to Conjugated Molecules, *Quantum Chemistry*, Vol. 75, pp. 55–66 (1999).
8) 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, 情報処理学会研究報告 HPC-96 (2003).