数理解析研究所講究録 1362 巻 2004 年 133-141

Ninf-Gによる ネットワーク数値計算ライブラリーの提供

スイミーソフトウェア株式会社　藤田有哉 (Ariya Fujita)
Swimmy Software,Inc.
日本ビジュアルニューメリックス株式会社　寺内和也 (Kazuya Terauchi)
Visual Numerics Japan,Inc.
産業技術総合研究所　田中良夫 (Yoshio Tanaka)
National Institute of Advanced Industrial Science and Technology
産業技術総合研究所　関口智嗣 (Satoshi Sekiguchi)
National Institute of Advanced Industrial Science and Technology

1　はじめに

本稿は、ネットワークによる遠隔地計算機利用技術の一つであるGridRPCを利用して、遠隔地の高性能計算機をサーバーとして科学技術計算を行う新しいアプリケーション開発法について論じている。

GridRPCは、手元の端末から広域ネットワークに散在する計算資源を活用することを可能にするミドルウェアである。GridRPCによって、スーパーコンピューターの高性能な計算能力があたかも自分の端末に備わっているかのようなアプリケーションを開発することができる。

またサービス提供側で利用者のリクエストを複数の高性能計算機に資源分散的に割り振ることにより、ネットワーク上に仮想的な計算機センターのサービスを構築することができる。

Ninf-Gは数値計算ライブラリーをネットワークライブラリーとして提供することに適したGridRPCの実装である。我々はNinf-Gを使って、LAPACK, IMSL® などの、いくつかの汎用的な数値計算ライブラリーをネットワークライブラリー化(「グリッド化」)した。

これら数値計算ライブラリーのグリッド化への移植作業は、ユーザーがアプリケーション中でネットワーク上のライブラリーをオンデマンドで使うことのできる、数値計算のアプリケーションサービスプロバイダーを構築する第一歩である。

本稿の構成は、2. にGridRPCを利用したネットワーク上の仮想計算機センターの構想について語る; 3. にNinf-Gの概要とアプリケーション開発法を記す; 4. に各数値計算ライブラリーのナイーブなグリッド化を阻む要因とその対応のテクニックについて; 5. では商用の数値計算ライブラリー、特にVisual Numerics® 社のIMSL® のグリッド化について述べる。

2　ネットワーク数値計算ライブラリーによる仮想計算機センター構想

我々は、数値計算ライブラリーをネットワーク上のサービスとして提供し、利用者が自由にこのサービスを利用できるための技術を研究している。

従来大規模な数値計算や複雑な計算を行う場合には、スーパーコンピューターや並列計算機などの高性能計算機を利用する技量に熟達する必要があり、効果的に利用できるまでに少なからぬ時間を必要とした。

我々が構想している数値計算ライブラリーのネットワークサービスでは、利用者は実際にネットワークの背後で数値計算を行っている高性能計算機群を意識すること無く、あたかも自分の手元の端末に高性能計算機の能力が備わっているかのようにシームレスに計算を行わせることができる。

本稿で対象としている計算機の利用法は、利用者が自分専用のプログラムを開発し、その中で汎用的な数値計算関数を使用している場合である。本稿では、利用者が従来自分のPCにインストールされた汎用ライブラリーの関数を利用していたものを、ネットワークライブラリーの関数に置き換えるだけで高性能計算機を利用できるような、新しいネットワークアプリケーションの開発方法を提示している。

この新しいネットワークアプリケーション開発法を支える基盤技術は、GridRPCである。「グリッド遠隔手続き呼び出し」GridRPCは、ネットワークコンピューティングを容易に行うためのミドルウェアのソフトウェア開発技術であり、グリッドコンピューティングの中核技術の一つである。

GridRPCは、本稿で論点とするサービスプロバイダーとしての利用法以外に、手元の端末から遠隔地のスーパーコンピューターの高性能の計算能力を利用したり、地理的に分散配置された複数の計算機を利用して高帯域の計算を行うことができる、自由度と応用性の高い技術である。

GridRPCによるサービスプロバイダーのメリットの一つには、バックエンドで動作する実際の計算機は利用者からは隠蔽され、利用者は自分にとって必要な機能だけを得ることができるということが挙げられる。

サービスプロバイダーのシステムでは、実際に動作しているサーバー上のライブラリーのインストール・保守作業はサーバーの側の管理者に委ねらている。したがって、利用者はこれらの面倒な作業に煩わされる必要はなく、高性能で高品質なライブラリーを即時使用することが可能なのである。

またもう一つのメリットとして、バックエンドで動作する高性能計算機群に対して適切な資源管理によるジョブの負荷分散が行われるならば、計算機資源を無駄にするような待ち時間を減じることができ、利用者に快適なライブラリーサービスを提供することができるということがある。

適切な資源管理のもとで計算サービスがネットワークで提供されることで、あたかもネットワークに接続するだけで使用可能なネットワーク上の仮想的な計算機センターを構築することができるのである。

我々は、このようなGridRPCを利用した仮想計算機センター構築の一環として、代表的な数値計算ライブラリーをNinf-G[1]のライブラリーとして提供する開発作業を進めている。

Ninf-Gは、GridRPCの実装の一つでありサーバー・クライアント型のネットワークアプリケーションの開発環境である。サーバー側・クライアント側でのネットワークアプリケーションの記述・開発法が非常に理解しやすい仕様であるため、特にネットワークプログラマーの層が薄い科学技術計算の分野においてネットワークアプリケーションを開発する環境として適している。

我々はすでに線形演算のライブラリーLAPACK[2], その並列計算機版ScaLAPACK[3], 離散フーリエ変換のFFTW[4]について開発が終了しており、Ninf-Gのインターフェースとしていつでも提供できる状態にある。またVisualNumerics® 社の数値・統計計算ライブラリーIMSL®[5]についてもネットワークライブラリーとして提供する研究を行っている。

我々の開発したネットワークライブラリーを高性能計算機群にインストールし、利用者にはNinf-Gクライアントの開発環境を配布することで、利用者はどこでもすぐに利用可能な仮想計算機センターのサービスを入手することができる。

2.1 ネットワーク数値計算ライブラリーとは

従来、汎用ライブラリーを利用してアプリケーションを開発する場合には、手引き書にしたがってソースコード中で目的関数の呼び出しを記述し、コンパイル時に目的関数を含んだライブラ

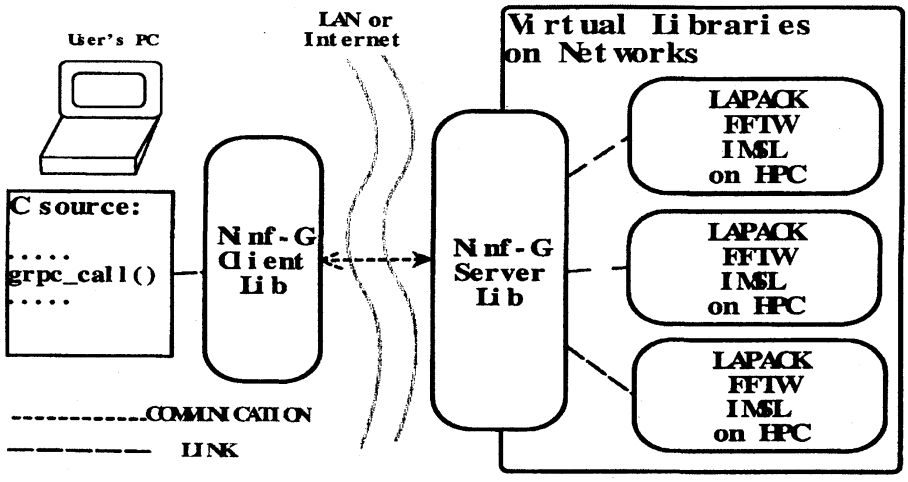

図 1. ネットワーク数値計算ライブラリー

リーと自分のプログラムを結合させて実行ファイルを生成するという作業を行ってきた。

ネットワークライブラリーのシステムでは、目的関数のライブラリーはネットワーク上の別の計算機上に置かれ、クライアント側のプログラムはクライアント用の通信ライブラリーに結合されるだけである。サーバー側では、数値計算ライブラリーはサーバー機能をもつ実行ファイルに結合される。目的関数に渡す引数などのデータは、サーバー・クライアントそれぞれの通信ライブラリーに仲介されて送受信されることで、クライアントプログラムは目的の計算結果を得ることができる。(図 1.)

通信の仕組みは利用者には隠蔽されているため、利用者は自分のアプリケーションであたかも自分の PC にライブラリーがインストールされているように、ネットワーク上の計算機資源を使用することができる。

2.2 Ninf-G によるライブラリープロバイダー

ネットワーク上のサービスとして提供されている数値計算ライブラリーを利用することにはどのようなメリットがあるのだろうか。以下にネットワークライブラリーのメリットを列挙する。

(1) ネットワークライブラリーの導入が容易

ユーザープログラムを実行ファイルにするためには、目的のライブラリーと結合する必要があるが、ライブラリーによってはこの結合の作業が非常に煩雑なものになる。ネットワークライブラリーを利用する場合、クライアント側のユーザープログラムは、通信ライブラリーと結合しさえすればいいので、実行ファイルの構築が容易になる。

(2) アプリケーション開発が特定の計算機環境から開放される

HPC の世界では、アーキテクチャーによって得手とする数値計算が異なる。通常の結合では、使用したい複数のライブラリーのアーキテクチャーが統一されていないと結合することができない。ネットワークライブラリーの場合は、数値計算ライブラリーの動作環境が 64bit で、グラフ化アプリケーションが 32bit であるような場合でも、容易に組み合わせて利用することができる。

(3) 仮想計算機センター

代表的な数値計算ライブラリーをネットワーク上に構築されたセンターで提供することで、サービスに登録するだけで数値計算関数を必要なときにすぐに利用することができる仮想計算機センターを構築できる。端末上のプログラムに高性能計算機の計算性能がそなわっているかのように使用することができる。

これらのメリットは、今後汎用ライブラリーを利用するアプリケーション開発においてネットワークライブラリーを利用することを推奨する十分な理由になると我々は考える。

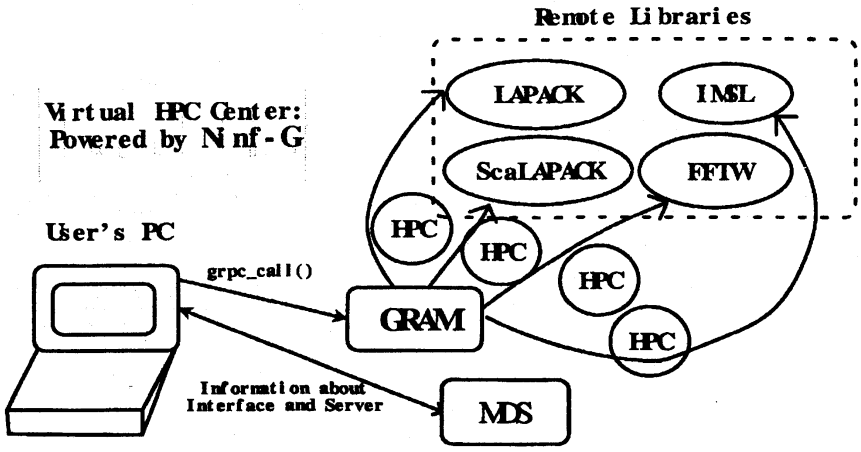

図 2. Ninf-G による仮想計算機センター

2.3 Ninf-G による仮想計算機センター構想

図 2. に本稿の主題である Ninf-G による仮想計算機センターの構想を示す。利用者の端末で動作するクライアントプログラムは、ネットワークライブラリー中の関数を呼び出すときに、初めに MDS(Meta Directory Service) という情報サーバーにアクセスする。情報サーバーは、目的関数と結合されたサーバーの実行ファイルのネットワーク上の位置と引数についての情報をクライアントに返す。

クライアントプログラムは MDS から得た情報からあらためて GRAM(Grid Resource Authorization Manager) というジョブマネージャーにサーバープログラムの実行を要求する。GRAM は実行を要求されたサーバープログラムをバックエンドの高性能計算機で実行し、サーバー・クライアント間で通信の接続が行われる。

利用者は、GRAM の背後で動作する高性能計算機そのものについて知る必要が無く、GRAM から適切な資源分散ができるバッチシステムを利用すれば、複数の高性能計算機を使用する快適なサービスを提供することができる。

3 グリッド遠隔呼び出し手続き (GridRPC) Ninf-G 概要

本節では Ninf-G の概要について説明する。はじめに Ninf-G の開発の歴史と動作環境について概説する; つづいて Ninf-G によって既存のアプリケーションをグリッドアプリケーション化するための工程と、クライアント側・サーバー側それぞれのグリッドアプリケーション化に必要な開発作業について説明する。

ネットワークライブラリーとして提供する関数の例として LAPACK の代表的なソルバー dgesv を使用する。

3.1 Ninf-G 略歴

Ninf-G は産業技術総合研究所 (旧電子技術総合研究所) で開発されたグリッドアプリケーション開発環境である。

1994 年、GridRPC の実装として前進の Ninf がリリースされた。このときに既存のアプリケーションをグリッド化する作業を簡略化するための、ネットワークライブラリーの IDL ファイル、IDL コンパイラ、クライアントプログラム用コンパイラなどの規格が既に定められている。

2001 年、Globus Toolkit を利用した Ninf の実装として Ninf-G がリリースされ、Globus の機能を組み合わせることによって、より汎用的なグリッド機能を提供できるようになった。また、以降認証や通信の隠蔽化は Globus の機能を利用するようになった。

2003 年には、Ninf-G2 が発表され、大規模ネットワーク上の多くの高性能計算機上で同時に計算を行わせるためのジョブ管理機能の充実、サーバープログラムに内部状態を保持させより高度な

計算能力をもたせるためのリモートオブジェクト機能が追加されている。

Ninf-G が動作する環境は以下の通りである。

```
Globus Toolkit:
       Globus Ver2 (2.0 2.2)
       Globus Ver1 (1.1.3 1.1.4)
OS: RedHat 6.1,6.2,,7.1,7.2,7.3
    Solaris 7,8
    IRIX 6.5
対象ライブラリー (サーバー側): C 言語
                Fortran77, Fortran90
クライアント:   C 言語
```

3.2 Ninf-G によるネットワークアプリケーション開発の工程

既存のアプリケーションを Ninf-G によってグリッドアプリケーション化する工程について述べる。

今までローカルな環境で動作することを前提に開発されたアプリケーションをネットワークライブラリーを利用して動作するプログラムに移植することを「グリッド化」–"Gridify"という。

この移植の工程は、開発の流れとして大別すると、サーバー側のネットワークライブラリーのインターフェース設計とネットワークライブラリーを利用するクライアントプログラムの 2 つの流れになる。

クライアント側では、ソースコード中の目的関数を呼び出している箇所を、GridRPC 呼び出し grpc_call() に書き換えるだけでよい。改定したソースコードはクライアントプログラム用のコンパイラに通す。このコンパイラは通常使用する (gcc のような)C コンパイラのラッパーであり、必要なクライアント用の通信ライブラリーを結合し、実行ファイルを生成する。

サーバー側でのネットワークライブラリー作成は、初めに関数の引数やデータサイズを記した Interface Description Language を記述する。この IDL ファイルを IDL コンパイラーに通すことによって、MAKE ファイルとサーバー実行プログラムのソースコードが作成される。make することでソースコードから目的関数のライブラリーと通信ライブラリーが結合されたサーバー実行ファイルが生成される。

Ninf-G ではグリッドアプリケーション開発が、サーバー側・クライアント側ともに定型化されているため、ネットワークプログラムの初心者でも容易にグリッドアプリケーション開発を習得することができる。

3.3 Ninf-G クライアントプログラム

Ninf-G のネットワークライブラリーを使用するクライアントプログラムの製作について解説する。

クライアントプログラムは、ネットワークライブラリーの目的関数を次のように呼び出す。(図 4.)

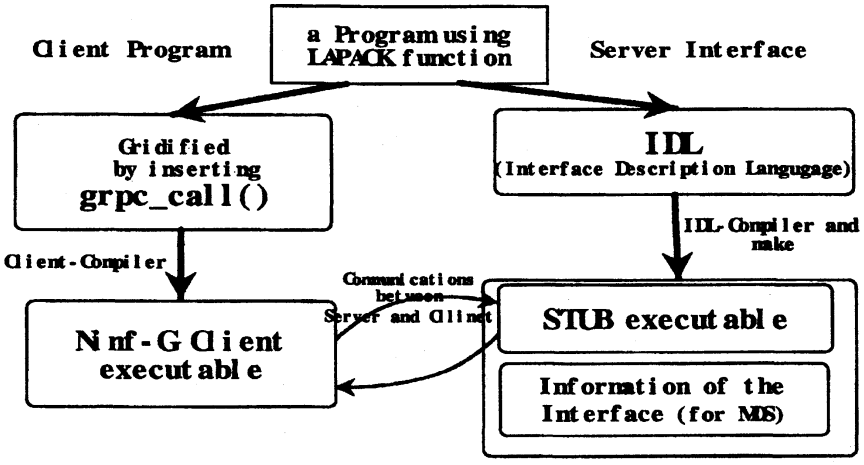

図 3. Ninf-G によるグリッドアプリケーション開発

```
....
grpc_function_handle_init( &handle,
                           host, port,
                      "lapack/dgesv" );
grpc_call( handle,
           n, nrhs, a, lda,
           ipiv,
           b, ldb, &info );
....
```

図 4. ネットワーク上の LAPACK を使用するクライアントプログラム

図 4. のクライアントプログラムで行っていることは、目的関数の handle を獲得し、GridRPC コール grpc_call() を使用しているだけである。このプログラムのコンパイルは Ninf-G クライアント用コンパイラ (cc のラッパー) で行う。

利用者はわずかな改変だけで自分のアプリケーションをグリッド化することができる。

3.4 Ninf-G によるネットワークライブラリー製作

サーバーが提供するネットワークライブラリーは IDL(Interface Description Language) ファイルから生成される。IDL ファイルには、対象関数への引数の渡し方など関数の取り扱いの他に、対象ライブラリーの結合の仕方、コンパイルオプションなどを記述する。(図 5.)

Ninf-G の関数の引数として使用できる型は、整数型、文字型、単精度・倍精度実数型、単精度・倍精度複素数型、およびこれらの変数の配列である。それ以外の構造体やポインター型は利用することができない。

対象ライブラリーの結合の仕方やコンパイルオプションは、実行ファイルを生成するための MAKE ファイル中のコンパイル用のフラグに使用される。これらのオプションによってコンパイル方法が決定される。

IDL ファイルを IDL コンパイラーに通すと、サーバープログラムのソースコードと MAKE ファイルが生成される。make を実行するとサーバープログラムのオブジェクトファイル、目的関数を含む数値計算ライブラリーと Ninf-G の通信ライブラリーが結合され、サーバー実行ファイルが生成される。同時に IDL に記述された関数の引数をもとに情報ファイルが作成され、MDS という情報サーバーに登録される。

IDL ファイル 1 つから、Ninf-G のサーバーに必要なすべてのファイルが生成されるため, ネットワークライブラリー開発者は定型的な工程を踏むだけで製作を進めることができる。

```
Define dgesv( IN int n,
              IN int nrhs,
              INOUT double a[lda][n],
              IN int lda,
              OUT int ipiv[n],
              INOUT double b[ldb][nrhs],
              IN int ldb,
              OUT int *info
)
Calls "Fortran" dgesv( n, nrhs,
                       a, lda,
                         ipiv,
                    b, ldb, info );
```

図 5. LAPACK 関数の IDL(サーバーインターフェース)

IDL コンパイラによって生成されるサーバプログラムのソースコードは、終了が要求されるまで永久に続くループの中で、必要な関数の引数を通信によって獲得し、目的関数に渡すという構造になっている。(図 6.)

数値計算ライブラリーを利用するためには、実行ファイルの中で目的関数の呼び出しを行わなければならないが、Ninf-G ではサーバー側の実行ファイルがこの呼び出しを行い続けながらクライアントに結果のデータを返信することで、あたかも関数そのものをネットワーク上で提供しているように見せかけている。

```
_stub_dgesv.c:

main() {
  nsstb_INIT();
  while(1) {
  ...
  nsstb_BEGIN();
  {
     dgesv( n, nrhs,
            a, lda,
              ipiv,
       b, ldb, info );
  }
  nsstb_END();
  ...
  }
  nsstb_EXIT();
}
```

図 6. サーバー実行プログラムのソースコード

4 ネットワークライブラリー開発のテクニック

前述の LAPACK のような単純な引数のインターフェースをもった関数に対しては、逐次 Ninf-G 用の IDL を作成することでネットワークライブラリー化することが可能である。しかしながら、ある種の関数はナイーブな Ninf-G の IDL 製作法ではうまくグリッド化できない場合がある。

またサービスの利用者に使いやすいインターフェースを提供する目的や並列計算のフロントエンドを提供する目的から、工夫された中継関数を用意する必要が生じる。

本節では、今回開発した個別の数値計算ライブラリーのインターフェースの特徴から、中継関数を製作するテクニックについて解説する。また我々がネットワークライブラリーの提供について目指すところから、開発上注意した点について述べる。

4.1 LAPACK - ワークバッファの自動確保機能

線形ソルバー LAPACK の関数のインターフェースについては、前節の例のように多くの関数がナイーブなグリッド化を許す。Ninf は 1994 年の最初のリリースから LAPACK の関数をグリッド化できるように仕様が定められていた。

一部の関数では、ワークバッファを確保するサイズをユーザー定義に委ね、サイズに応じて記憶容量と計算時間のトレードオフが生じるように作成されている。Ninf-G 版のインターフェースでは、最低限度必要なワークバッファはサーバープログラム側で確保し、それ以上のワークバッファは利用者が決定できるように改訂してある。それゆえ、初心導入者は使用に際してワークバッファのサイズに煩わされる必要が無い。

4.2 ScaLAPACK - 並列計算のシリアルインターフェース

LAPACK の並列計算機版というべき ScaLAPACK をグリッド化するにあたり、本来の並列計算は隠蔽化し LAPACK のように手軽に使用できるインターフェースを提供した。したがって、提供しているインターフェースは利用者から見ると、ScaLAPACK ではなく LAPACK にみえることになるが、利用目的が並列計算そのものより並列計算による計算速度の向上や大規模計算であると想定されるので不都合は生じないものと判断している。

ScaLAPACK による並列計算を LAPACK のようなシリアル関数に見せかけるための、行列データの各ノードへの分配・計算終了後の各ノードからのデータ収集は、C 言語マクロを利用して簡略化して表現できるように開発した。このことにより、600 個近くある ScaLAPACK 関数のほとんど全てにシリアル計算のインターフェースを与え、グリッド化することを達成している。

4.3 FFTW - 最適計算の実現

FFTW2.1.5 をグリッド化する作業には 2 つの大きな問題があった:

(1) FFTW のアルゴリズムの中核である"fftw_plan"が構造体であるため、Ninf-G 関数の引数には使用できない。

(2) 初期化関数と主計算関数が分離されていて、初期化によって最適化を、主計算はこの最適化のデータを繰り返し用いる構造になっている。

対処 (1) 構造体"fftw_plan"については、FFTW に備えられた構造体と文字列を変換する関数を利用して対応した。Ninf-G では fftw_plan に対応する文字列を最適化データとしてやりとりし、主計算の高速化に活用するようにした。

対処 (2) 離散フーリエ変換の最適化は、Ninf-G のサーバープログラムの特徴である「通信のループの中で目的関数を呼び出していること」を利用して、最適化された内部状態をサーバー側で保持し続けることで解決した。主計算部にあたる離散フーリエ変換の繰り返し中では、サーバープログラムも継続して実行され続けるようにし、最適計算の環境を崩さないようにした。

サーバープロセスの内部状態の維持による計算の高速化により FFTW の最適化の有無による速

度差をグリッド化しても実現しているものと考えられる。(表 1.) 詳細な計算時間の測定については、現在検証中である。

表 1. FFTW の計算時間: ローカル版とグリッド版
ローカルなテストとそのグリッド版の実行時間の実時間測定
ESTIMATE モードと MEASURE(最適化) モードを比較

	ローカルテスト	グリッド版
(A) ESTIMATE	144.352	874.189
(B) MEASURE	93.572	833.569
(A)-(B)	44.780	40.620

(単位 秒)

Pentium4 2.0AGHz 512MB
サイズ=1234567 繰り返し回数 40 回
グリッド版は localhost に対する通信で実験
グリッド版は一台の PC でサーバー・クライアント両役を行っているので実行時間が大きいが、最適化の有無による時間差は同程度生じている。

4.4 IDL 製作の上でのポイント

以上、ユーザーインターフェースを提供する上で中継関数を提供する例をみてきたが、我々がこれらネットワークライブラリー製作に際して、特に押えておくべきと判断したポイントについて以下 3 点を列挙する。

1. **互換性**
   既存のアプリケーションを容易にグリッド化できるように引数の互換性を維持する。

2. **導入コストの削減**
   オプションの自動調整や並列計算の隠蔽化など初心導入者に優しい設計。

3. **パフォーマンス**
   繰り返し使用される関数は、オーバーヘッドの低減について考慮した。

これらのポイントを押えることにより、初心導入者にとっては導入が簡単で、既利用経験者にとっては妥当な計算速度のネットワークライブラリーを提供できることになる。

5 商用ライセンスされた数値計算ライブラリーのグリッド化

我々は商用ライセンスされた数値計算ライブラリーがネットワークサービスで提供される可能性についても検討を行っており、Visual Numerics 社の商品である IMSL についてもグリッド化の試験を行った。

本節では、IMSL のグリッド化とその精度評価、および IMSL も含め商用ライセンスされたライブラリーをグリッド化する問題点について述べる。

5.1 IMSL のグリッド化

IMSL のグリッド化の試験を日本ビジュアルニューメリックス社の協力と試用ライセンスの提供によって行った。

今回のインターフェース製作に関して利用したものは、IMSL のマニュアルに記述された関数の使用法のみであり、ソースコードの閲覧は行っていない。Ninf-G では、構築されたライブラリーに直接結合してネットワークサービスを構築するので、ソースコードを閲覧すること無くある程度の性能・機能のインターフェースを作成することができる。

ヘテロ環境での精度評価

アーキテクチャーの異なる計算機環境でサーバー・クライアントの関係を構築し、IMSL のグリッド化の計算精度を測定した。計算精度について計算の誤差ノルムをローカル計算とネットワーク計算で算出して比較した。

VisualNumerics 社によれば、IMSL は計算機のアーキテクチャーによらずに等しい計算結果を返すことができるとのことである。今回の結果では最小精度の範囲で相違が生じており、この相違の原因については今後の研究を必要とする。

```
Ninf-G サーバー:
 PentiumIII 1.4GHz Dual
 OS RedHat 7.3
 コンパイラー PGI 3.2-4
 ライブラリー IMSL Fortran95 V4.0

Ninf-G クライアント: Sun-Blade-1000
 UltraSPARC-III SMP 150MHz
 Memory 2GB
 OS SUN Solaris8
 コンパイラー SunWorkshop 6.2
```

5.2 商用ライセンスされたライブラリーのグリッド化

本論の最後に、商用ライセンスされたライブラリーをグリッド化する際に発生するビジネス上のポイントをメリット・デメリット両側面から考察する。

商用ライセンスの数値計算ライブラリーは、特定のライセンスされた高性能計算機でしか実行できない。そのため、これらのライブラリーをグリッド環境で利用できるということは利用者の間口を拡大し裾野を拡げることになる。この点についてベンダーのメリットを見出すことができる。

一方、ネットワーク上で商用ライセンスのライブラリーを使用することの違約性の有無を考察すると、現状のNinf-Gではサーバープロセスのアカウントはそのサーバー上でのユーザーアカウントであり、ネットワークライブラリーを利用することとサーバーマシンにログインして目的のライブラリーを利用することに大差を見出せず、違約性は発生しないと我々は考えている。

ただし、ベンダーは、堅牢な認証・回線の隠蔽化など十分なセキュリティが用意されているグリッド環境での利用にかかわらず、ネットワークに自分たちの商品であるライブラリーを晒すことには抵抗感があるかもしれない。しかし、曖昧な抵抗感からビジネスのチャンスを逃すべきではない。

グリッドアプリケーションの開発は、高性能計算機に閉じ込められていた使い勝手が良いとはいえない資源を幅広く利用者に開放し、利用者人口の拡大に貢献するものであり、長期的にベンダーを資するものであると、我々は確信する。ネットワーク活用に対する新しいライセンスを開拓し、積極的なビジネス展開をすることを各ベンダーに期待したい。

6 まとめ

本稿では、Ninfチームが公開を予定しているネットワークライブラリーとしての各種数値計算ライブラリーの開発状況について報告した。我々はネットワークで数値計算ライブラリーを提供する仮想計算機センターの構想を抱いており、手始めにLAPACK,ScaLAPACK,FFTWのNinf-Gのインターフェースの開発を完了した。

Visual Numerics社の著名な商用の数値・統計計算ライブラリーであるIMSLについても、ネットワークライブラリー化のための研究を進めている。

これら開発状況についての情報は、Ninf WEBページ

http://ninf.apgrid.org

から入手することができ、ライブラリーのIDLファイルもここから提供する予定である。

計算機とそれをとりまく環境の発展とともに、それを利用するための応用技術も発展していく。現在のところ、GridRPCが使用されている環境は計算機科学の専門的な応用分野に限られているが、将来的には一般的に広く利用される技術になると考えられる。

ダイナミックリンクライブラリーが現在のOS上では基本的な技術として当り前に使用されていてなおかつ利用者が普段意識することはないように、ネットワークライブラリーもネットワークによる情報インフラストラクチャーを縁の下の力持ち的に支えるようになるだろう。

どのような形でGridRPCの規格が標準化されるにせよ、GridRPCが広く使用される時代には、我々の開発した数値計算ライブラリーのグリッド化インターフェース群が非常に有効に利用されるはずである。

謝　辞

日頃お世話になっております、Ninfチームの皆様に感謝いたします。

本研究の一部は科学技術振興機構「計算科学技術活用型特定研究開発推進事業」の一環として実施している「仮想スーパーコンピュータセンタ利用環境GridLibの構築」によるものである。

参考文献

1) 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), or http://ninf.apgrid.org

2) http://www.netlib.org/lapack

3) http://www.netlib.org/scalapack

4) http://www.fftw.org

5) http://www.vni.com