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

2003－HPC－96 (12)
2003／10／16

Ninf-Gによるグリッド数値計算ライブラリーの構築

藤田有哉* 田中良夫†
関口智嗣†

我々はNinf-G用の数値計算ライブラリーのインターフェースを製作した。このインターフェースを使用することで、LAPACK, FFTW, IMSL ® 等の数値計算ライブラリーをネットワーク経由で利用することができる。Ninf-Gクライアントの利用者がこれらのライブラリーを利用する上で必要なものはNinf-G環境のみである。

Development of numerical libraries for GridRPC by using Ninf-G

ARIYA FUJITA*, YOSHIO TANAKA†
and SATOSHI SEKIGUCHI†

We have developed numerical libraries for Ninf-G. These are supplied as interface description files (IDL) of Ninf-G, and you can execute remote functions of LAPACK, FFTW and IMSL®, etc. via the Internet by using these Grid libraries. What users of Ninf-G clients require for using these libraries is just a Ninf-G system.

1 はじめに

広域ネットワークに散在する計算機資源を有効に活用するためのグリッド技術として、アプリケーション開発者にはGridRPCが提供されている。GridRPCは遠隔の計算資源をローカルライブラリーのようにアプリケーションの内部で使用することを可能にする、グリッド技術のミドルウェアである。GridRPCの環境として、NetSolve [1] ,Ninf1 [2] , Ninf-G [3] が挙げられる。

我々は、代表的な数値計算ライブラリーであるLAPACK [4] , ScaLAPACK [5] , FFTW [6] , IMSL ®[7] をGridRPCで使用するグリッドライブラリーを開発した。グリッドライブラリーがインストールされたサーバーに遠隔呼び出しをかけるだけで、クライアントは特別な設定を行うこと無くローカルマシンの上の関数を使っているようにサーバー上の関数から実行結果を得ることができる。

これらグリッドライブラリーの環境にはGridRPCの実装Ninf-Gを採用した。

Ninf-Gによるグリッドライブラリー作成作業のほとんどは、目的の関数に対するIDL(Interface Description Language)を記述することである。

Ninf-GのIDLファイルは、C言語のように記述され作成が容易なものであるため、上記で挙げた数値計算ライブラリーに含まれる関数の個数が膨大なものであるにもかかわらず、グリッド化の作業は比較的短期で八ヶ月程度で済んだ。

[1]スイミーソフトウェア
Swimmy Software,Inc.
[2]産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

Ninf-G の利用者はクライアントからサーバーのリモートライブラリーを呼び出す方法を短期間で習得することが可能であり、ローカルマシン上で動作するアプリケーションを容易にグリッドアプリケーションに改変することができる。

代表的な数値計算ライブラリーを Ninf-G 用のグリッドライブラリーとして提供することは、グリッドアプリケーションの利用者および開発者の間口を拡大することにつながる、と我々は確信している。

本稿の構成は、次節で今回のグリッドライブラリー製作の指針について述べる; 続く 3 (LAPACK),4 (ScaLAPACK),5 (FFTW),6 (IMSL) の各節は各ライブラリーのグリッド化における仕様と特徴を示す。結びに (7 節) 今後の展望を挙げる。

2 ライブラリー製作の指針

Ninf-G 用のグリッドライブラリーは IDL(Interface Dexcription Language) の形で提供される。IDL ファイルには、対象ライブラリーのリンクの仕方、対象関数への引数の渡し方などを記述する。

今回我々が開発したグリッドライブラリーには、ライブラリーの位置やコンパイルオプションなどアーキテクチャーに依存する部分をインストール時に調整できるように Makefile が用意されている。

我々は Ninf-G 用のリモートライブラリー IDL を次の方針で製作した。

(1) 互換性

上記のライブラリーの関数を使用するアプリケーションを容易にグリッドアプリケーションに移植できるように、引数の渡し方をオリジナルの関数に近づけた。返値を返す関数については、引数の最後尾に返すようにした。

(2) 導入の簡易化

初心導入者が容易に目的のリモートホスト上の関数を使用できるように、次の 2 点を念頭に置いて開発した。

(a) クライアントに専用の特別な設定・リンクを要求しない。

利用者は目的のホストに grpc_call() するだけで、期待する結果を得ることができる。クライアントに必要な環境は Ninf-G ready であればよく、特定のオブジェクト・ライブラリーファイルを必要としない。

(b) オリジナルの関数の引数の複雑さを低減した。

とくにワークバッファは、互換性のために形式的に渡す必要はあるが、メモリーを確保して渡す必要は無い。

(3) パフォーマンス

何度も繰り返し呼び出されることが想定される関数、特に FFTW の関数については、なるべくオリジナルの関数のパフォーマンスを損なわないようにオーバーヘッドを低減した。

我々は、数値計算ライブラリーを Ninf-G のグリッドライブラリー化する際, 支障が無い限り Ninf-G と同じくオリジナル名末尾に-G を付加して呼称する。

以降の各節で我々がグリッドライブラリー化した数値計算ライブラリーについて述べる。

3 LAPACK-G

LAPACK は多くの OS ベンダーから最適化されたものが提供されている線形演算のためのライブラリーであり、ソースコードは Netlib [8] から自由に取得できる。

LAPACK は単精度, 倍精度、実数, 複素数型の二次元の密行列と帯行列の線形ソルバーを提供している。線形ソルバーの種類は、

LE: 連立方程式
SEP,NEP: 固有値問題
SVD: 特異値分解

と分類される。

LAPACKは、Ninf1初リリース時からIDLファイルが提供されているNinfのIDLファイルの仕様と相性の良いライブラリーである。(図1.)

Ninf1リリース時のIDLは倍精度実数型の関数を扱っていたが、今回我々は、LAPACK-Gで1000以上あるオリジナルのLAPACKのすべての関数、すなわち単精度実数型、倍精度実数型、単精度複素数型、倍精度複素数型の関数のIDLファイルを開発した。

```
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 1. LAPACK 関数の IDL

以降の副節より、利用者の利便性から付加されたLAPACK-Gの機能を述べる。

3.1 Cライブラリー化と配列

Fortranの関数であるLAPACKは参照渡し(Cのポインター)で引数を渡さなければならないが、LAPACK-Gでは値渡しで引数を渡すようにした。(図2.) Ninf-GはC言語ベースであり、利用者もC言語でプログラムを開発すると考えられるからである。

2次元行列は行優先で1次元の配列に格納されなければならない。2次元行列の取り扱いは、Fortranの関数をC言語から呼び出す使い方と同様である。密行列の場合はLAPACKのマニュアル[9]に従って、余計な0部分が省略された配列で渡す。

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

図 2. LAPACK を使用するアプリケーションのグリッド化

3.2 ワークバッファの自動アロケーション

LAPACKの関数は必要なワークバッファのアロケーションを利用者に委ねている場合が多い。LAPACKでは、ワークバッファのサイズに応じて、メモリーと計算時間のトレードオフが生じるように設計されている関数が多い割合で存在するからである。

LAPACK-Gではgrpc_call()時に渡されるバッファサイズlworkとソース中に記述された最小限必要な量を比較し、大きい方の量を採用してメモリーを確保する。

初心利用者は、ワークバッファのサイズをマニュアルから調べること無くLAPACK-Gの関数を利用することができる。

3.3 固有値ソート

LAPACK-Gの内16個の関数は、利用者が定義したSELECT関数を利用して、固有値に対するソートを自由に行うことができる。(図2.)

```
/* 複素数を選択しソートするための関数 (TRUE/FALSE) */
long select(float *wr, float *wi) {
/* wr, wi: A の固有値 W の実部, 虚部 */
  long ret = 0;
  if ((*wi==0.)) ret=1;
/* 虚部が 0 の固有値にたいして優先的に */
/* ソートする。*/
  return ret;
}

  ...
/* main() 中の呼び出し */
sgees( "V", "S", select, &n,
       a, &lda, &sdim, wr, wi,
       ..., &info);

/* w[0..] が純虚数であれば優先的に */
/* w[] の先頭に配置する。*/
```

図 2. ユーザー定義関数で固有値ソートする。

LAPACK-Gでは、ユーザー定義関数は文字列の引数として渡し、解釈にYACCを用いることで、この機能を実現している。

我々は単一マシン上でローカルな計算の計算時間を測定することで、YACCを利用することのオーバーヘッドを測定した。(表1.)

サイズの増大に対して、行列データの送受信の通信量及び演算は2次元的に増大するが、固有値ソートは1次元的なものなので、YACCで作成したインタープリターであって

も速度的な問題は生じない、と我々は考えている。

表 1. 主要計算部、ネイティブコードと YACC インタープリターの比較。ネイティブコード,YACC ともに虚部==0 を試した。(%) は実行時間全体に占める割合

	行列サイズ 250x250	行列サイズ 1000x1000
VOID	0.93	51.49
NATIVE	0.09(8.8%)	4.51(8.1%)
YYLEX	0.71(70%)	7.05(12%)

COUNT 150 回
(単位 秒)

4 ScaLAPACK-G

ScaLAPACK は NetLib で提供されている分散メモリー型並列計算機のための並列化された LAPACK である。ScaLAPACK の関数は LAPACK と同等の関数の並列化版が提供され、分散的に配置された密行列・帯行列を処理することができる。

並列計算機で動作する ScaLAPACK は、利用者が並列処理の記述子を作成し関数に渡す必要があり、また ScaLAPACK 関数の呼び出し前後で配列の分配・収集の処理を行う必要がある。

我々の開発した ScaLAPACK-G では、これらの並列計算の処理はラッピング関数が行うことで隠蔽され、ScaLAPACK の関数を対応する LAPACK の関数のように使えるインターフェースを提供している。

したがって、利用者はバックエンドの並列処理 (分散・並列計算・収集) を意識すること無く、バックエンドの並列計算機で動作している ScaLAPACK 関数を LAPACK のように使うことができる。

LAPACK-G と同様、ScaLAPACK-G もワークバッファを自動的に確保するようにしている。ScaLAPACK のワークバッファの必要サイズは LAPACK よりも複雑なものであるため、この機能は初心利用者の導入を容易にする。

ScaLAPACK-G の関数への引数の渡し方は LAPACK-G と同様の規則に従う。ただしピボットつきの密行列に限り、LAPACK と配列の格納形式が異なる [10]。

我々の、ScaLAPACK-G のラッピング関数の製作は、C 言語のマクロによる記述で行われた。このマクロを使うことで、分散メモリーに置けるメモリ確保、分配、収集を簡潔に記述することができる。(図 3.) 我々は ScaLAPACK のほとんどすべての関数約 600 個の IDL を作成した。

```
Define psgesv(  IN int n,
        IN int nrhs,
        INOUT float global_a[ROW_a][COL_a],
        IN int lda,
        OUT int global_ipiv[n],
        INOUT float global_b[ROW_b][COL_b],
        IN int ldb,
        OUT int *info
)
OPTIONS
{
        INITIALIZE();

        SCALAR( "int", n);
        SCALAR( "int", nrhs);
        SCALAR( "int", lda);
        SCALAR( "int", ldb);

        MATRIX( "float", a, ROW_a, COL_a);
        MATRIX( "float", b, ROW_b, COL_b);
        VECTOR( "r", "int", ipiv, ROW_a);

        DISTRIBUTE( "float", a, ROW_a, COL_a);
        DISTRIBUTE( "float", b, ROW_b, COL_b);

        psgesv( &n, &nrhs,
         loca, &one, &one, desca,
         locipiv,
         locb, &one, &one, descb, &linfo);

        GATHER( "float", a, ROW_a, COL_a);
        GATHER( "float", b, ROW_b, COL_b);
        vGATHER( "r", "int", ipiv, ROW_a);

        RETRIEVE("int", &linfo, 1);

        FREE_MATRIX(a);
        FREE_MATRIX(b);
        FREE_VECTOR(ipiv);
}
```

図 3. ScaLAPACK の LAPACK 化するラッピング関数はマクロで記述する。

5 FFTW2-G

FFTW は GPL に従って配布されている離散フーリエ変換のライブラリーである。このライブラリーを使うことで、1 次元・他次元、複素数・実数、任意のサイズで、離散フーリエ変換を行うことができる。

我々は FFTW 2.1.5 を対象に FFTW-G の IDL を開発した。FFTW の最新のバージョ

ンはFFTW3であるが、FFTW2との互換性が全く失われているため、今回開発したIDLはFFTW3には対応していない。

FFTW2の特徴として、2のベキ乗以外のサイズの離散フーリエ変換に対して独特のアルゴリズムを用いて計算することが挙げられる。

離散フーリエ変換は、サイズが2の冪乗でない場合は素因数分解法とRader法を用いることで計算量を低減することができる。このとき、与えられた離散フーリエ変換をどういう順番で分解するかで計算時間に相違が生じ、この相違は実行する計算機に依存する。

FFTW2にはこの分解を実測によって最適化する、fftw_plan構造体が用意されている。fftw_plan構造体は、より小さいサイズのfftw_planへの連結リストになっていて、関数"fftw_create_plan"を呼び出したときに最適化の結果が蓄えられる。

Ninf-Gの仕様では、連結リストのような構造体を送受信する機能がない。そこでFFTW2-Gでは、FFTW2でfftw_planのファイルへの保存に使用されるwisdomという文字列を代わりにサーバー クライアント間で送受信することで、計算を高速化する。(図4.)

指定したサイズによっては、wisdom作成による最適化を行った場合、行わなかった場合に比べて、FFTW2-Gの主要計算部の計算速度が向上する。しかしながら、wisdomの作成は何回かの離散フーリエ変換の実測という計算量をともなうので、wisdomの機能を使用することが有効か否かはwisdom作成に必要な時間の追加と主要計算部の時間の短縮のトレードオフで決定されなければならない。

オリジナルのFFTW2は必ずfftw_planを作成してから計算本体を行わなければならないが、FFTW2-Gは初心導入者のために計算本体をただ呼び出すだけでフーリエ変換を行うことができる。

```
char wisdom[WISDOMLEN];  /* WISDOMLEN = 8000 */

Ninf_call( "fftw/fftw_create_plan",
           n,
           FFTW_FORWARD, FFTW_MEASURE,
           wisdom );

/* wisdom 作成を省略して、'fftw_one' を */
/* 実行することも可能である。 */
```

```
grpc_function_handle_init( &handle,
                           host, port,
                           "fftw/fftw_one" );

for(count=0;count<howmany;count++ ) {
   grpc_call( &handle,
              wisdom,
              in, out,
              isize, osize);
}

grpc_function_destruct(&handle);
```

図4. FFTW2-Gのクライアントの例

6 IMSLのグリッド化

IMSLはVisual Numerics ® 社の商用の数値計算および統計計算ライブラリーである。

我々は、日本ビジュアルニューメリックス社の好意でIMSLを試用させてもらい、ソースコードを見ること無く、マニュアルをもとに10数個の関数についてNinf-Gのリモートライブラリーにすることに成功した。

今回リモートライブラリー化した関数は、オリジナル関数の引数の扱いの相違から次の2種類に分類される。

(a) Fortarn77の固定引数の関数

Fortran77用に用意されたIMSLの関数は、引数の個数が関数毎に固定されている。LAPACKの関数の引数の渡し方と同様なので、これらの関数のIDLはLAPACK-G用のIDLを作成した方法と同じ要領で作成できた。

(b) Fortran90, C言語用の可変引数の関数

C言語のマクロva_argで実装されているような、引数の個数が可変になっている関数がIMSLの現在のトレンドである。

Ninf-Gの仕様は、可変長引数の関数に対応していないため、サーバー側、クライアント側のいずれかもしくは両者に可変長引数と固定長引数の関数の間をラッピングする関数が必要となる。

もしクライアント側にラッピング関数のオブジェクトを置く必要があるならば、利用者に必要な環境はNinf-Gのみという今回の開発の趣旨からは少し外れることになる。

今回のテストでは、可変引数の関数に対しサーバー側にのみラッピング関数を用意して対応した。

7 今後の展開

LAPACK-G,FFTW2-G,ScaLAPACK-G については、配布可能なアーカイブがすでに作成されており、パフォーマンスや精度の検討を通じて今後改良されていくことになる。

現在、開発作業を検討中である 2 項目を示す。

(1) IMSL のグリッドライブラリー化

IMSL のグリッドライブラリー化については今回はテストのみで、すべての関数をサポートする IDL 製作までには至らなかった。IMSL のすべての関数をサポートする IDL の製作を研究中である。

(2) FFTW3-G

新しくリリースされた FFTW Version 3 は、より抽象化されたインターフェースの関数に改変され、Version 2 とは全く違うものになっている。そのため、Version 3 用の IDL については現在グリッドライブラリーの仕様を検討中である。

謝　辞

さまざまな面でお世話になっております, 産業技術総合研究所グリッド研究センターの皆様、それから Ninf チームの皆様に感謝致します。いつも御指導いただきどうもありがとうございます。

IMSL のグリッド化について御協力いただいた日本ビジュアルニューメリックス社に感謝致します。

参考文献

1) Casanova,H. and Dongarra,J.: NetSolve: A Network Server for Solving Computational Science Problems, *Proceedings of Supercomputing '96* (1996).

2) http://www.ninf.apgrid.org

3) 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)

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

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

6) http://www.fftw.org

7) http://www.vni.com

8) http://www.netlib.org

9) Anderson,E., Bai,Z., Bischof,C., Balckford,S., Demmel,J., Du Croz,J., Greenbaum,A., Hammarling,S., McKenney,A. and Sorensen,D. : LAPACK Users's Guide Third Edition, the Society for Industrial and Applied Mathematics, 1999

10) Balckford,L.S., Choi,J., Cleary,A., Demmel,J., Dhillon,I., Dongrra,J., Hammarling,S., Henry,G., Petitet,A., Stanley,K., Walker,D.and Whaley,R.C.: ScaLAPACK User's Guide, the Society for Industrial and Applied Mathematics, 1997