第1回 NetSolveゼミ

ゼミ担当者 ： 輪湖純也，谷口義樹，斉藤宏樹
指導院生 ： 青井桂子
開催日 ： 2002 年 10 月 21 日

ゼミ内容: 本ゼミでは，Grid RPC System である NetSolve について説明する．ここでは，NetSolve とは何か，NetSolve の構成，さらに NetSolve の可能性について触れ，最後に NetSolve の利用例を示す．

1 はじめに

本ゼミでは，Grid RPC System の一つである NetSolve[1]について説明する．NetSolve を使うとどのような利点があるのか，NetSolve はどのような構成になっていて，実際に利用するためにはどのような手順が必要かを説明する．

2 NetSolve とは

NetSolve とは，Tennessee 大学の Innovative Computing Laboratory の Jack Dongarra 等によって開発された Grid RPC System である．Grid RPC とは，Grid Computing に対する remoto procedure call(RPC) の仕組みを標準化・実装しているものである．NetSolve は遠隔地にある計算資源にアクセスすることができ，遠隔地のコンピュータにある最適化されたライブラリを使用することができる仕組みを兼ね備えている．

NetSolve により，ユーザは最新のソフトウェアのインストールの手間や，遠隔地のコンピュータへの login の手間が省けるようになる．例えば，LAPACK[2](Linear Algebra PACKage) などの数値ライブラリを利用して，プログラムを実行させる場合，通常は以下の二つの方法をとる．

- LAPACK のソースファイルあるいはオブジェクトファイルを手元のコンピュータにインストールし，それらを利用するユーザプログラムとリンクし実行する．
- 計算センタなどのスーパーコンピュータに login し，既にインストールされているライブラリを利用する．

このような二つの方法ではユーザに手間がかかる．まず手元のコンピュータにライブラリをインストールする場合は，ライブラリのバージョンが上がるたび，あるいはソフトウェアのバグが見つかるたびに，ライブラリを新しくする必要がある．また，計算センタの場合は，利用するためにわざわざ計算センタのスーパーコンピュータに login し，実行する必要がある．このような手間を省くことができる以外に，NetSolve は，いつでも誰でもどこからでも実験が行える環境をつくりだす可能性を潜めている．

3 NetSolve の構成

NetSolve システムは,「ゆるく」結合されたマシンの集合からなる．ここでいう「ゆるく」の意味とは，これらのマシンがローカル，ワイド，グローバルなネットワークでつながれ，異なる団体や組織によって運営されているということを意味している．さらに，NetSolve システムはヘテロな環境での連係をサポートしている．つまり，異なるアーキテクチャ，OS の違いをもつマシン同士が，同時にこのシステムに参加できる．

NetSolve は，以下に示す 3 つのコンポーネントで構成されている（Fig. 1 参照）．

- NetSolve クライアント
- NetSolve エージェント
- NetSolve サーバ（リソース）

NetSolve クライアントは，手元のコンピュータで動作し，NetSolve エージェントを通じて実際の計算を NetSolve サーバで実行させることができる．

実行の流れは，次のようになる．

[1]バージョンは 1.4.1(2002/10/10 現在)
[2]線形計算ライブラリで， 連立一次方程式，線形最小二乗問題，固有値問題などを解くことができる．

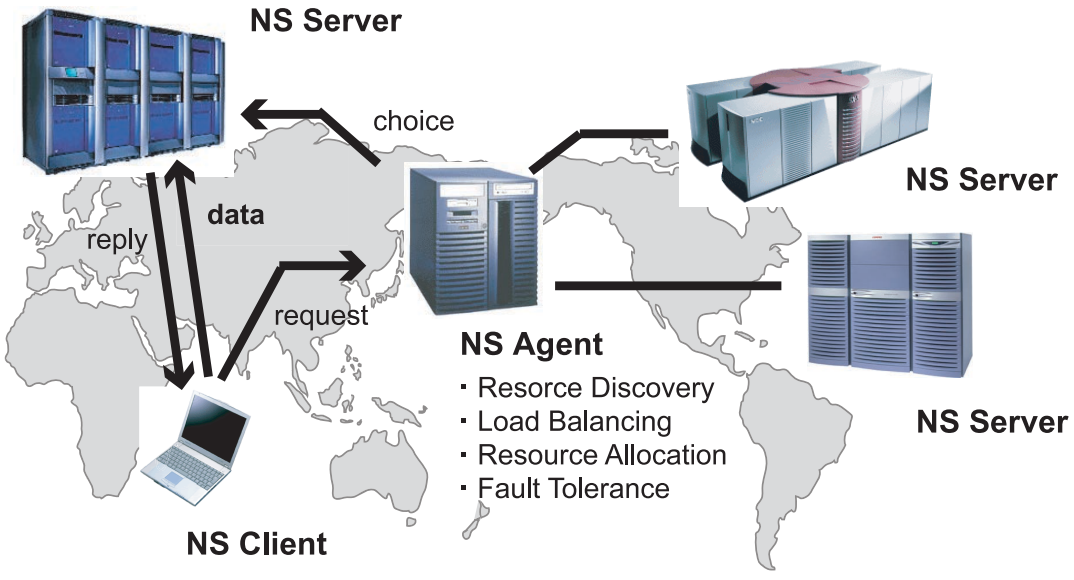

Fig. 1　NetSolve の構成

1. クライアントがエージェントに対して実行要求を行う.
2. エージェントは要求を受け取り，登録されているサーバから，ロードバランスを考慮して適切な計算リソース（サーバ）を選ぶ.
3. クライアントは，選ばれた計算リソースにデータを送る.
4. サーバでは，クライアントからデータを受け取り，数値ライブラリを実行し，結果をクライアントに返す.

4　NetSolve の可能性

NetSolve は単にインストールの手間や login の手間を省くだけではない．ライブラリの実行をするコンピュータ (以下，計算リソース) はネットワークに接続され，NetSolve サーバが起動していさえすればよいので，世界中のコンピュータで実行できる可能性を持っている．現在，NetSolve のインターネットを用いた環境におけるテストベッドとして，デンマークコンピューティングセンタや NCSA(National Center for Supercomputing Applications)，テネシー大学ノックスビル校など十数か所で NetSolve サーバが起動し，いつでも誰でもどこからでも実験が行える環境が整っている[3].

計算リソースは，ワークステーション，スーパーコンピュータ，計算機クラスタなど，さまざまな計算機であり，エージェントはその時々のネットワーク状況，計算機の負荷をみながら最適な計算リソースを選んでくれる.

[3]http://icl.cs.utk.edu/netsolve/people/partners.html

5　NetSolve の install

この章については，添付資料『NetSolve 簡易マニュアル』を参照されたい．

6　NetSolve の動作実験

実際に NetSolve を使用し，遠隔地にあるライブラリを利用した動作実験を行う．簡単なライブラリとプログラムを作成し，クライアント，サーバー，エージェントを同一 LAN に接続されたマシンで設定する．

6.1　実験の構成

Fig. 2 のようにクライアント，サーバー，エージェントを構成し，実験を行う．

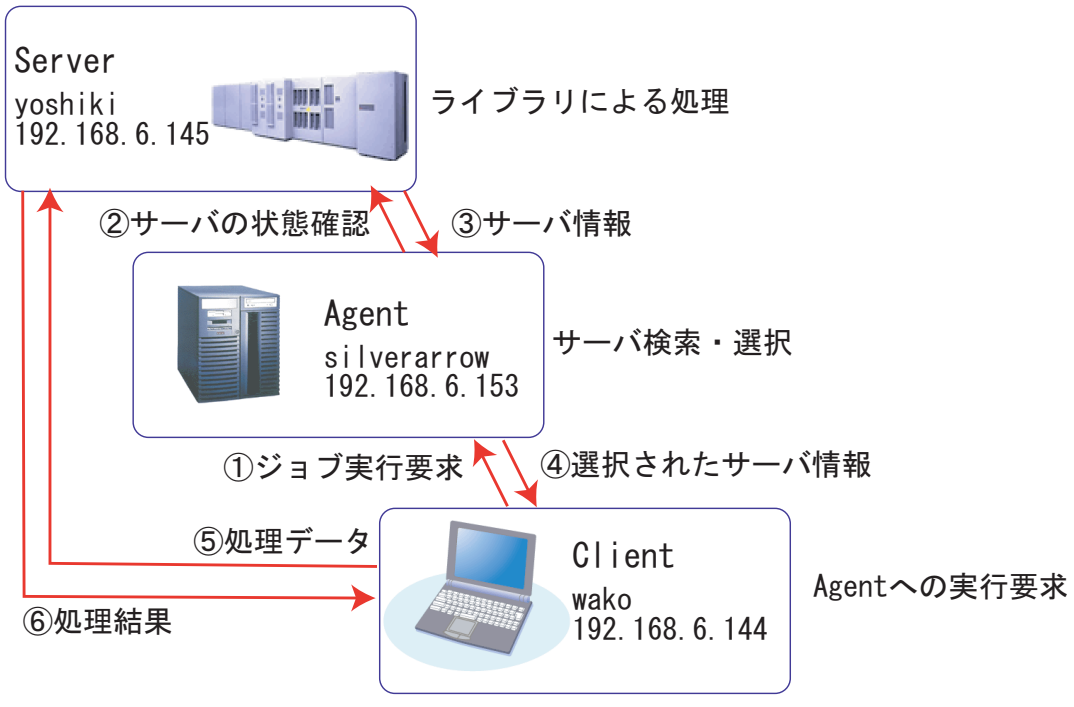

Fig. 2　NetSolve の実行環境

Fig. 2 の環境を設定するために，各マシンでにおいて/etc/hosts にクライアント，サーバー，エージェントの IP アドレスを以下のように加える (/etc/hosts のファイルは root 権限でしか書き込めないので，root 権限で設定する)．

```
192.168.6.144 wako.work.isl.doshisha.ac.jp wako
192.168.6.145 yoshiki.work.isl.doshisha.ac.jp yoshiki
192.168.6.153 silverarrow.work.isl.doshisha.ac.jp silverarrow
```

6.2　NS Server の設定

サーバ側では，ユーザが定義した関数を使用できるように PDF(Problem Description File) を作る必要がある．今回の例では，以下の Fig. 3 のような PDF ファイルを作成し，~/NetSolve-1.4.1/problems/func という名前で保存した．記述内容については，添付資料『NetSolve 簡易マニュアル』を参照されたい．実際に実行される部分は，@CODE〜@ENDCODE に記述される．今回の例では，別に用意した func() 関数をコールしている．

```
@PROBLEM func
@INCLUDE <unistd.h>
@LIB /home/yoshiki/ns_semi/func.o
@DASHI /home/yoshiki/ns_semi
@INCLUDE "func.h"
@LANGUAGE C
@MAJOR COL
@PATH /semi/func/
@DESCRIPTION
Basic test
Adds 1 to an integer
@INPUT 1
@OBJECT SCALAR D x
integer in input
@OUTPUT 1
@OBJECT SCALAR D OUTPUT
integer in output
@COMPLEXITY 1000000,20000

@CALLINGSEQUENCE
@ARG I0
@ARG O0

@CODE
   @O0@ = (double *)malloc(1 * sizeof(double));
   *(@O0@) = func(*(@I0@));
  sleep(5);
@END_CODE
```

Fig. 3　PDF ファイル (func)

実際にコールされる func() 関数の内容 (func.c) を Fig. 4 に，参照されるヘッダファイル (func.h) の内容を Fig. 5 に示す．この func.c をコンパイルし，func.h とともに~/ns_semi というディレクトリ内に置いておく．

```
#include "func.h"

double func(double data){
  return data+1.0;
}
```

Fig. 4　func.c

次に，server_config ファイルを編集し，サーバで利用できる PDF のリスト (@PROBLEMS:) に，作成した PDF を追加する．今回の sever_config ファイルは以下の Fig. 6 のような内容になっている．個々の記述に意味については，添付資料を参照されたい．編集後，“make standard” する．

```
extern double func(double data);
```

Fig. 5　func.h

```
@PROC:1
@AGENT:silverarrow.work.isl.doshisha.ac.jp *
@WORKLOADMAX:-1
@SCRATCH:/tmp/
@MPIHOSTS ./MPImachines 4
@PROBLEMS:
./problems/testing
./problems/qsort
./problems/func
#./problems/mandelbrot
#./problems/blas_subset
#./problems/lapack_subset
#./problems/lapack
#./problems/scalapack
#./problems/sparse_iterative_solve
#./problems/sparse_direct_solve
#./problems/arpack
#./problems/testingglobus
@RESTRICTIONS:
* 10
```

Fig. 6　server_config

6.3　NS Client の設定

クライアント側では，NS Server に登録されている関数を呼び出すためのインタフェースが必要となる．本ゼミでは，各種あるインタフェースの中から C のブロッキング呼び出しである **netsl** を用いる．**netsl** は次のように定義されている．

```
int netsl(char *problem_name, , <argument list>, )
```

以下の Fig. 7 にクライアント側のプログラムを示す．このプログラムは，先ほどサーバ側で定義した関数 func() をクライアント側から呼び出し，indata に対して 1 を足して outdata に返すというものである．

コンパイルは，netsl を用いているためライブラリを指定する必要がある．

```
gcc client.c -I /home/junya/NetSolve-1.4.1/include -L /home/junya/NetSolve-1.4.1/lib/i686_pc_linux_gnu -lnetsolve
```

```
#include "netsolve.h"
#include <stdio.h>

int main(){
  double indata = 1.0;
  double outdata = 0.0;

  netsl("func()",&indata,&outdata);

  printf("Input  Data:  %f\n",indata);
  printf("Output Data:  %f\n",outdata);

  return 0;
}
```

Fig. 7　クライアントプログラム (client.c)

6.4　実行結果

以上の実行環境を用いて，実際に実験を行う．先ほど指定した，NS Server，NS Agent をそれぞれ起動し，実行する．以下の Fig. 8 に実行結果を示す．

```
junya@wako:~/netsol_test$ ./a.out
Initializing NetSolve...
Initializing NetSolve Complete
Sending Input to Server yoshiki.work.isl.doshisha.ac.jp
Downloading Output from Server yoshiki.work.isl.doshisha.ac.jp
Input  Data:  1.000000
Output Data:  2.000000
```

Fig. 8　実行結果

参考文献

1) NetSolve，http://icl.cs.utk.edu/netsolve/

2) 建部 修見,「グローバルコンピューティング [4] 米国から見た GCI/Network servers」，Computer Today 2000.7 No.98