PARALLEL HIGH
PERFORMANCE COMPUTING
CdS magistrale in informatica

11 – Netsolve

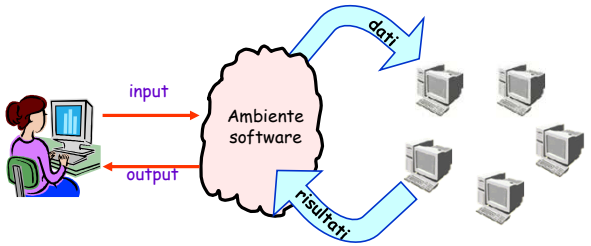
Dipartimento di Matematica e Applicazioni
Universita' degli Studi di Napoli Federico II

wpage.unina.it/lapegna

1

ruolo di un ambiente per il c.d.

- gestire gli accessi alle risorse
- gestire l'eterogeneita'
- gestire la dinamicita'



La scelta delle risorse hardware e software
e' completamente a carico dell'ambiente software

2

NetSolve : obiettivi primari

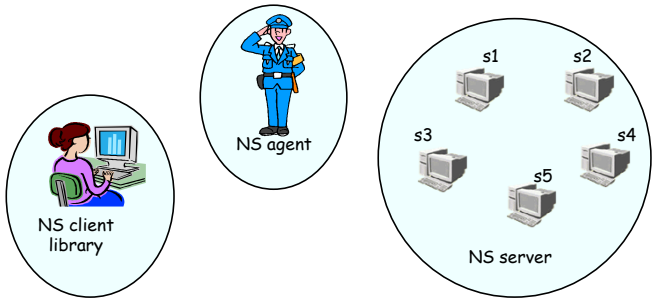
- permettere l'utilizzo "a distanza" di software matematico di alta qualita'
- mettere a disposizione l'hardware per la relativa esecuzione

Possibilmente in maniera facile…

Network Computational Server

3

NetSolve: componenti fondamentali



4

NetSolve agent

E' la porta di accesso a un sistema NetSolve

Esso gestisce un database con informazioni sulle risorse computazionali del sistema:

- performance hardware dei server
- software disponibile sui server
- statistiche sull'uso dinamico dei server

Il NS agent usa le informazioni del database per

- trovare e allocare le risorse di calcolo,
- bilanciare il carico tra i server
- tenere traccia di eventuali fallimenti

NetSolve client library

Al momento della compilazione, l'applicazione viene linkata alla NS client library

Tramite le chiamate a tale libreria l'applicazione accede alle risorse di calcolo di NetSolve senza alcuna conoscenza delle reti e dell' hardware coinvolto

La NS client library e' disponibile con interfacce a:

- Fortran
- C
- Matlab
- Mathematica
- Excel (in corso di sviluppo)
- Octave
- Condor-G

NetSolve server (1)

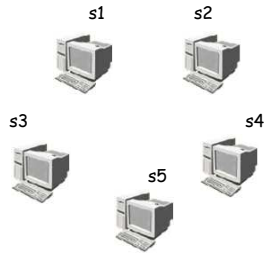
Sono le risorse di calcolo dell'intero sistema

Essi possono essere:

- workstation
- cluster
- sistemi paralleli

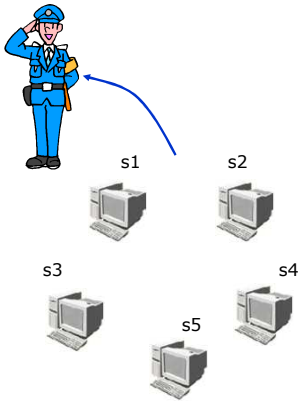
Un NS server puo' essere:

- un server software (repository)
  (viene messo a disposizione solo il software residente sul server)
- un server hardware
  (viene messo a disposizione solo l'hardware per l'esecuzione di software residente sui server software)
- un server hardware e software
  (viene messo a disposizione sia l'hardware che il software residente)



NetSolve server (2)

server	P	t_0	B	P1	P2
s1	xxx	yyy	zzz	aaa	bbb
s2					
s3					



Per fare parte di un sistema NetSolve, il server deve registrare sul database dell'agente le proprie caratteristiche statiche:

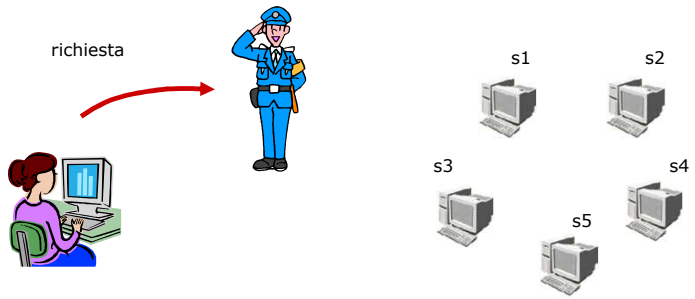
- Performance P
- Latenza t_0 e bandwidth B della rete
- Software disponibile p1, p2,…

NetSolve server (3)

Il software presente sui server deve essere
incluso in un wrapper
che ha lo scopo di fornire una interfaccia uniforme verso
NetSolve e che contiene una serie di informazioni quali

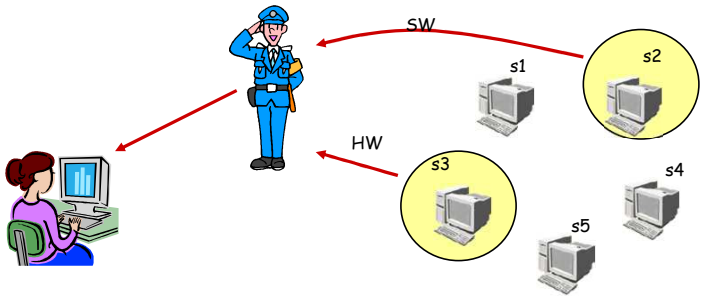
- documentazione
- caratteristiche del problema
- complessita' computazionale dell'algoritmo utilizzato

Overview del funzionamento: richiesta



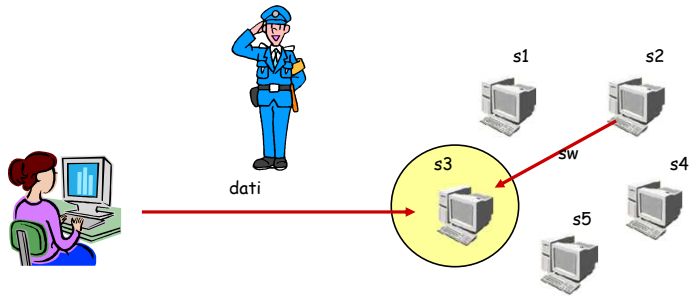
Il client invia la richiesta all'agent con le caratteristiche del problema da svolgere

Overview del funzionamento: scelta



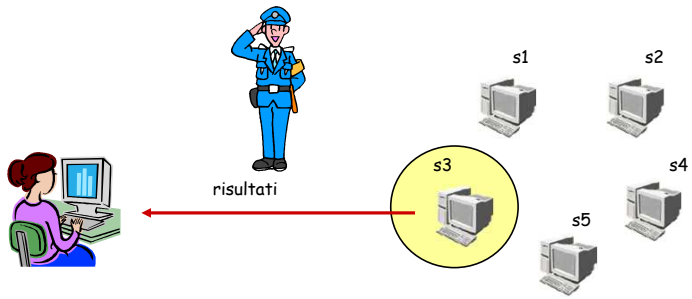
L'agent effettua la scelta del SW server (s2) e dell'HW server (s3), e le comunica al client

Overview del funzionamento: input



Il client invia direttamente al server HW i dati, dove viene inviato anche il software necessario dal server SW

Overview del funzionamento: output



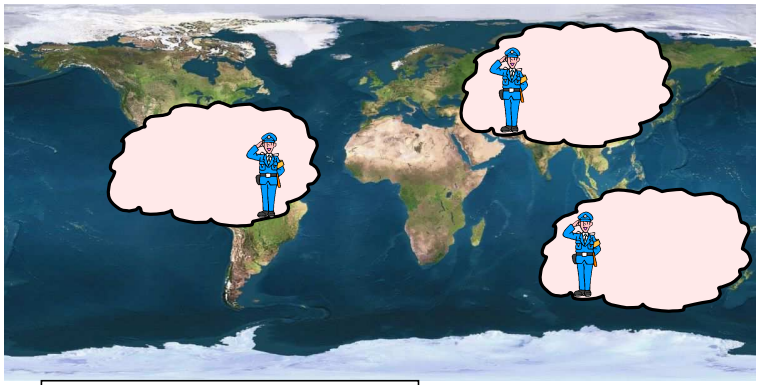
Il server HW invia direttamente
al client i risultati

Osservazione

Client, agent e server sono entita' distinte

possono risiedere in tre siti differenti
e formano un sistema NetSolve

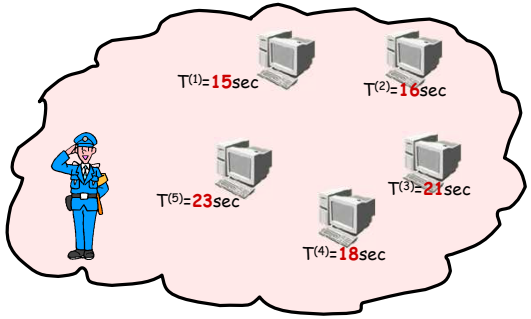
Osservazione (2)



- Esistono numerosi NetSolve system
- Sono indipendenti l'uno dall'altro
- Sono identificati dall'agent

il server migliore

Sia $T^{(i)}$ il tempo necessario stimato per risolvere un problema P sul server $S^{(i)}$



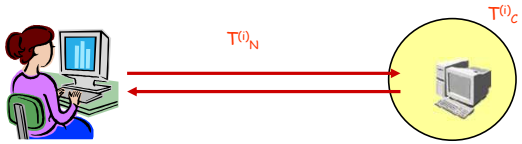
Il server migliore S^* e' quello che
minimizza una stima tempo totale di esecuzione

($T^* = \min T^{(i)}$)

Come viene calcolato $T^{(i)}$?

$T^{(i)}$ e' la somma di

$T^{(i)}_C$ tempo di calcolo su S (i)
$T^{(i)}_N$ tempo per spedire i dati dal client a $S^{(i)}$ e viceversa



Calcolo di $T^{(i)}_N$

Siano:
- t_0 latenza della rete tra client e server
- B bandwidth della rete (in bytes/sec)
- D_S input dal client al server (in bytes)
- D_R output dal server al client (in bytes)

$$T^{(i)}_N = \underbrace{t_0 + D_S/B}_{\text{input}} + \underbrace{t_0 + D_R/B}_{\text{output}}$$

Calcolo di $T^{(i)}_N$

- t_0 latenza della rete tra client e server
- B bandwidth della rete (in bytes/sec)

Sono informazioni disponibili
all'interno del database dell'agente

- D_S input dal client al server (in bytes)
- D_R output dal server al client (in bytes)

Sono informazioni disponibili
al momento della richiesta dal client

Calcolo di $T^{(i)}_C$

Siano:
- N dimensione del problema
- C(N) complessita' dell'algoritmo
- R performance effettiva del server

Dipende da
- performance "di riferimento" P
- workload w

$$T^{(i)}_C = C(N) / R$$

Calcolo di $T^{(i)}_C$

- N dimensione del problema

Disponibile al momento della richiesta dal client

- C(N) complessita' dell'algoritmo

Disponibile nel database dell'agente

- P performance di riferimento

Disponibile nel database dell'agente

- w workload

Comunicato periodicamente dal server all'agente

Perf. di riferimento e perf. effettiva

Nel database dell'agente e' conservata, per ogni server, una performance di riferimento P

(LINPACK benchmarck)

Al variare del workload w, la performance effettiva R puo' essere molto diversa

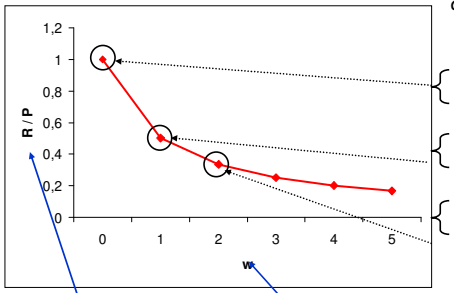
Necessita' di un modello per la performance effettiva R

Il modello di performance effettiva R

Viene utilizzato il modello

R = P/(1+w)

Motivazione: la CPU e' condivisa con altre applicazioni



con 0 applicaz. estranee si ha R = P,

con 1 applicaz. estranea si ha R = P /2,

con 2 applicaz. estranee si ha R = P /3

frazione di P ottenibile

Numero di applicazioni "estranee" (oltre alla mia in esecuzione)

Come viene calcolato $T^{(i)}$?

Ad ogni richiesta del client, il NS agent calcola per ogni server $S^{(i)}$ il tempo stimato di esecuzione $T^{(i)}$

$$T^{(i)} = T^{(i)}_C + T^{(i)}_N =$$

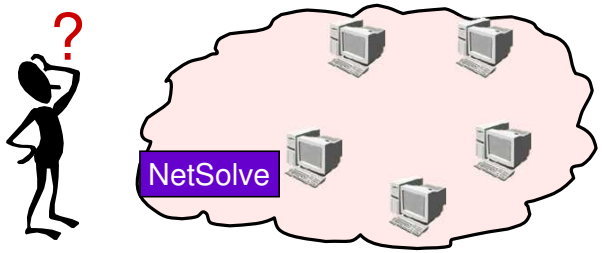
$$C(N)*P/(1+w) + t_0 + D_S/B + t_0 + D_R/B$$

E calcola il minimo dei $T^{(i)}$

25

26

Risorse computazionali di un NS system



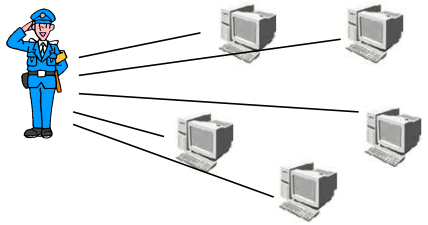
Dato un NS system
- qual e' l'hardware disponibile?
- qual e' il software disponibile?

27

osservazione

Ogni server, sia esso hw o sw e' registrato sul database del NS agent

Interrogare il NS agent!



Il comando

>> NS_config *agente*

fornisce le informazioni sulle risorse hardware disponibili nel sistema identificato da "*agente*"

"*agente*" e' il nome della macchina su cui e' in esecuzione l'agente da interrogare

28

esempio

Per interrogare l'agente in esecuzione su
renato.dma.unina.it

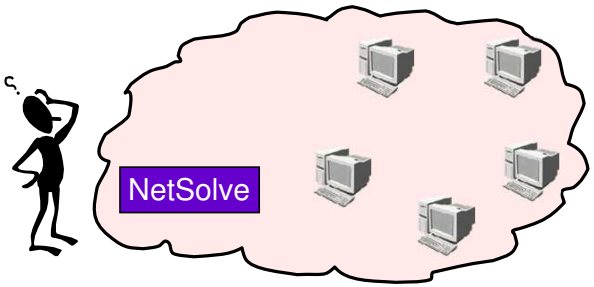
server registrati sull'agente
renato.dma.unina.it

```
>> NS_config  renato.dma.unina.it
AGENT: renato.dma.unina.it (192.167.11.200)
SERVER: pclab12.dma.unina.it (192.167.11.6) (0 failures)
SERVER: pclab14.dma.unina.it (192.167.11.4) (0 failures)
SERVER: pclab16.dma.unina.it (192.167.11.3) (0 failures)
>>
```


Esempio: uso di blas

Problema: si vuole effettuare il prodotto di due matrici di ordine n=4

C'e' qualcosa su NetSolve?



Informazioni sul software disponibile

Il comando

>> NS_problems *agente*

fornisce le informazioni sulle risorse software disponibili nel sistema identificato da "*agente*"

"*agente*" e' il nome della macchina su cui e' in esecuzione l'agente da interrogare

```
>> NS_problems  renato.dma.unina.it
/BLAS-wrappers/Level3/dmatmul
/BLAS-wrappers/Level3/zmatmul
/librerie/operazioni/somma
/BLAS/Level1/daxpy
/BLAS/Level1/ddot
/BLAS/Level1/zaxpy
/BLAS/Level2/dgemv
/BLAS/Level3/dgemm
/BLAS/Level3/zgemm
```

Routine DGEMM di BLAS per il prodotto di matrici

Informazioni su uno specifico sw

Il comando

>> NS_probdesc *agente* *problema*

fornisce le informazioni sul "*problema* " disponibile nel sistema identificato da "*agente*"

Come funziona dgemm? (1)

Comando >> NS_probdesc renato.dma.unina.it dgemm

```
. . .
* 7 objects in INPUT
 - input 0: Scalar Character.
 TRANSA -
TRANSA specifies the form of op( A ) to be used in
the matrix multiplication as follows:
 TRANSA = 'N' or 'n',  op( A ) = A.
 TRANSA = 'T' or 't',  op( A ) = A'.
 TRANSA = 'C' or 'c',  op( A ) = A'.
 - input 1: Scalar Character.
 TRANSB -
TRANSB specifies the form of op( A ) to be used in
the matrix multiplication as follows:
 TRANSB = 'N' or 'n',  op( A ) = A.
 TRANSB = 'T' or 't',  op( A ) = A'.
 TRANSB = 'C' or 'c',  op( A ) = A'.
 . . .
```


Come funziona dgemm? (2)

```
. . .
- input 2: Scalar Double Precision Real.
 alpha
 - input 3: Matrix Double Precision Real.
 Matrix A
 - input 4: Matrix Double Precision Real.
 Matrix B
 - input 5: Scalar Double Precision Real.
 beta
 - input 6: Matrix Double Precision Real.
 Matrix C
* 1 objects in OUTPUT
 - output 0: Matrix Double Precision Real.
 Matrix C after performing the computation
* Calling sequence from C or Fortran
13 arguments
 - Argument #0:
   - pointer to input object #0 (transa)
```


Come funziona dgemm? (3)

```
. . .
 - Argument #1:
   - pointer to input object #1 (transb)
 - Argument #2:
   - number of rows of input object #3 (a)
   - number of rows of input object #6 (c)
 - Argument #3:
   - number of columns of input object #4 (b)
   - number of columns of input object #6 (c)
 - Argument #4:
   - number of columns of input object #3 (a)
   - number of rows of input object #4 (b)
 - Argument #5:
   - pointer to input object #2 (alpha)
 - Argument #6:
   - pointer to input object #3 (a)
 - Argument #7:
   - leading dimension of input object #3 (a)
 - Argument #8:
   - pointer to input object #4 (b)
```


Come funziona dgemm? (4)

```
. . .
 - Argument #9:
   - leading dimension of input object #4 (b)
 - Argument #10:
   - pointer to input object #5 (beta)
 - Argument #11:
   - pointer to input object #6 (c)
   - pointer to output object #0 (c)
 - Argument #12:
   - leading dimension of input object #6 (c)
   - leading dimension of output object #0 (c)
>>
```

problema

Il comando NS_probdesc fornisce le informazioni sul "*problema* "

Come sottomettere il problema a NetSolve?

```
int netsl("problema ( )" , arg1 , arg2 , ... )
                                    rit.: =0 se ok, <0 se errore
```

La funzione netsl invia una richiesta a NetSolve

Ha come argomenti:
- il nome del problema
- gli argomenti cosi' come specificato da NS_probdesc

Prodotto di matrici con NetSolve

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

main(int argc, char **argv){

int i, j, n, info;
double A[10][10], B[10][10], C[10][10], alpha, beta;
char flag;

n=2;

for (i=0; i< n; i++){
for (j=0; j< n; j++){
      A[i][j] = 1+(int) (10.0*rand()/(RAND_MAX+1.0));
      B[i][j] = 1+(int) (10.0*rand()/(RAND_MAX+1.0));
}}

alpha=1.e0;
beta=0.e0;
flag='N';
```

Definizione matrici A e B

Inizializzazione argomenti DGEMM

Prodotto matrici con NetSolve

```
info = netsl("dgemm()", &flag, &flag, n, n, n, &alpha, A, 10, B, 10, &beta, C, 10);

printf("MATRICE A\n");
for (i=0; i< n; i++){
for (j=0; j< n; j++){
printf(" %f ", A[i][j] );
} printf("\n"); }

printf("MATRICE B\n");
for (i=0; i< n; i++){
for (j=0; j< n; j++){
printf(" %f ", B[i][j] );
} printf("\n"); }

printf("MATRICE C\n");
for (i=0; i< n; i++){
for (j=0; j< n; j++){
printf(" %f ", C[i][j] );
} printf("\n"); }

}
```

Chiamata a netsl per uso DGEMM

stampe

Esempio: Compilazione del programma

La funzione netsl ( ) fa parte della NetSolve client library

Link alla NetSolve client library

```
>> cc  -o  prodmatrici  prodmatrici.c  -I$(NETSOLVE_ROOT)/include
          -L$(NETSOLVE_ROOT)/lib/$(NETSOLVE_ARCH)   -lnetsolve
>>
```

Attraverso le funzioni della NetSolve client library il nostro programma e' in grado di accedere a NetSolve

Scelta del sistema NetSolve

Esistono numerosi sistemi NetSolve

Prima dell'esecuzione e' necessario sceglierne uno

Scelta dell'agente

Definizione della variabile di ambiente NETSOLVE_AGENT

A secondo della shell utilizzata. . .

csh

oppure

bash

```
setenv  NETSOLVE_AGENT renato.dma.unina.it

NETSOLVE_AGENT  renato.dma.unina.it
export  NETSOLVE_AGENT
```


esecuzione

```
>> prodmatrici
Initializing NetSolve...
Initializing NetSolve Complete
Sending Input of Task 20 to Server pclab16.dma.unina.it
Downloading Output from Server pclab16.dma.unina.it
MATRICE A
 9.000000  8.000000
 10.000000  4.000000
MATRICE B
 4.000000  8.000000
 2.000000  8.000000
MATRICE C
 52.000000  136.000000
 48.000000  112.000000
>>
```


ESEMPIO 2

IL PARALLELISMO IN NETSOLVE

problema

Come sommare due array di
10 elementi?

$$c_i = a_i + b_i \qquad i = 1,..,10$$

Prima soluzione

```
#include <stdio.h>
#include "netsolve.h"
int main( ){
float a[10], b[10], c[10];
int  i,  info;
for( i=0; i<10; i++){
       a[i]=1+(int) (10.0*rand()/(RAND_MAX+1.0));
       b[i]=1+(int) (10.0*rand()/(RAND_MAX+1.0));
       netsl("somma( )",&a[i],&b[i],&c[i]);
       printf("la somma di %f e %f  =  %f \n", a,b,c);
}
}
```

Chiamare 10 volte netsl ( )

supponiamo sia disponibile un programma "somma"

esecuzione

```
Initializing NetSolve...
Initializing NetSolve Complete
Sending Input of Task 5 to Server pclab14.dma.unina.it
Downloading Output from Server pclab14.dma.unina.it
la somma di 9.000000 e 4.000000  =  13.000000
Sending Input of Task 6 to Server pclab16.dma.unina.it
Downloading Output from Server pclab16.dma.unina.it
la somma di 8.000000 e 8.000000  =  16.000000
Sending Input of Task 7 to Server pclab14.dma.unina.it
Downloading Output from Server pclab14.dma.unina.it
la somma di 10.000000 e 2.000000  =  12.000000
......
```

Esecuzione prima somma
Esecuzione seconda somma
Esecuzione terza somma
...

Funzioni bloccanti

```
Initializing NetSolve...
Initializing NetSolve Complete
Sending Input of Task 5 to Server pclab14.dma.unina.it
Downloading Output from Server pclab14.dma.unina.it
la somma di 9.000000 e 4.000000  =  13.000000
Sending Input of Task 6 to Server pclab16.dma.unina.it
Downloading Output from Server pclab16.dma.unina.it
la somma di 8.000000 e 8.000000  =  16.000000
```

L'input della seconda somma non viene inviato prima della ricezione del risultato della prima somma

La funzione netsl(...) e' bloccante

osservazione

Le somme sono indipendenti

E' possibile eseguirle "in parallelo" su differenti server

int netsl_nb("problema ( )" , arg1 , arg2 , ...)

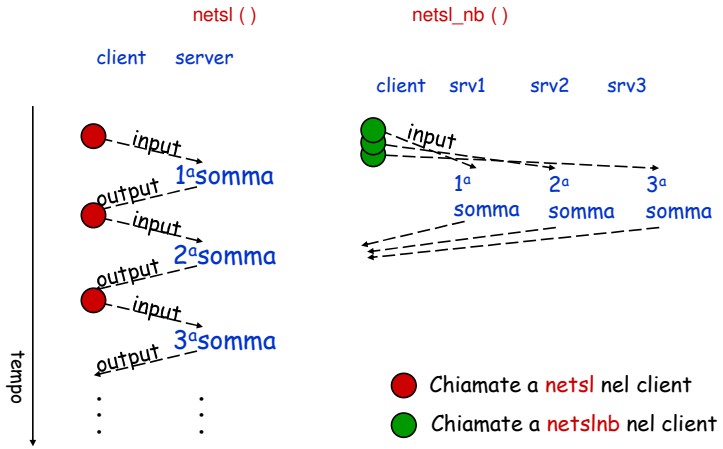
rit.: >0 se ok, <0 se errore

La funzione netsl_nb ( ) invia una richiesta a NetSolve senza attendere il risultato

Ha gli stessi argomenti di netsl ( )

netsl_nb ( ) ritorna un identificativo del problema inviato

Funzione netsl ( ) vs netslnb ( )



Esempio: somma di 10 coppie di numeri

```
#include <stdio.h>
#include "netsolve.h"
int main( ){
        float a[10], b[10], c[10];
        int i;
for( i=0; i< 10; i++){
        a[i]=1+(int) (10.0*rand()/(RAND_MAX+1.0));
        b[i]=1+(int) (10.0*rand()/(RAND_MAX+1.0));
        netsl_nb("somma( )",&a[i],&b[i],&c[i]);
}
.......
}
```


Esempio: esecuzione

```
>>
Initializing NetSolve...
Initializing NetSolve Complete
Sending Input of Task 5 to Server pclab16.dma.unina.it
Sending Input of Task 6 to Server pclab16.dma.unina.it
Sending Input of Task 7 to Server pclab16.dma.unina.it
Sending Input of Task 8 to Server pclab16.dma.unina.it
Sending Input of Task 9 to Server pclab16.dma.unina.it
Sending Input of Task 10 to Server pclab16.dma.unina.it
Sending Input of Task 11 to Server pclab16.dma.unina.it
Sending Input of Task 12 to Server pclab16.dma.unina.it
Sending Input of Task 13 to Server pclab16.dma.unina.it
Sending Input of Task 14 to Server pclab16.dma.unina.it
>>
```

E i risultati?

problema

Mediante netsl_nb ( ) vengono inviati i dati ad un server e non si attende il risultato.

Per sapere se l'esecuzione di un problema e' terminata

int netsl_pr(identificativo)

rit.: = 0 se ok, <0 se errore

La funzione netslpr ( ) verifica se il risultato del problema identificato dall'identificativo e' pronto

Ritorna:
0 se il risultato e' pronto
-1 se il risultato non e' pronto
-17 se l'identificativo non e' valido

problema

Mediante netslpr ( ) si sa solo se l'esecuzione e' terminata e il risultato e' pronto sul server.

Per ricevere il risultato dal server

int netsl_wt(identificativo)

rit.: = 0 se ok, <0 se errore

La funzione netslwt ( ) attende la fine dell'esecuzione del problema identificato dall'identificativo e fa ritornare la soluzione dal server

osservazioni

netsl_nb ( ) seguita immediatamente da netsl_wt ( ) equivale a netsl ( )

netsl_wt ( ) chiamata dopo netsl_pr ( ) che ritorna 0 ritorna subito

Esempio: somma di 10 coppie di numeri

```
. . .
for (i=0; i<10; i++){
        a[i]=1+(int) (10.0*rand()/(RAND_MAX+1.0));
        b[i]=1+(int) (10.0*rand()/(RAND_MAX+1.0));
        info[i] = netsl_nb("somma()", &a[i] , &b[i] , &c[i] );
}
fatti = 0;
while (fatti < 10){
        for (i=0; i<10; i++){
                if (netsl_pr (info[i]) ==0){
                        netsl_wt(info[i]);
                        fatti++;
                        printf("risultato di somma = %f \n", c[i]);
                }
        }
}
. . .
```

Si inviano i 10 problemi a NetSolve

Risultato i-mo pronto?
Se si, ricevi il risultato

Esecuzione (1)

Vengono inviati i dati di tutti i problemi

```
Initializing NetSolve...
Initializing NetSolve Complete
Sending Input of Task 130 to Server pclab16.dma.unina.it
Sending Input of Task 131 to Server pclab16.dma.unina.it
Sending Input of Task 132 to Server pclab14.dma.unina.it
Sending Input of Task 133 to Server pclab16.dma.unina.it
Sending Input of Task 134 to Server pclab14.dma.unina.it
Sending Input of Task 135 to Server pclab16.dma.unina.it
Sending Input of Task 136 to Server pclab14.dma.unina.it
Sending Input of Task 137 to Server pclab16.dma.unina.it
Sending Input of Task 138 to Server pclab14.dma.unina.it
Sending Input of Task 139 to Server pclab16.dma.unina.it
```

Esecuzione (2)

Downloading Output from Server pclab16.dma.unina.it
la somma di 9.000000 e 4.000000 = 13.000000
Downloading Output from Server pclab16.dma.unina.it
la somma di 8.000000 e 8.000000 = 16.000000
Downloading Output from Server pclab14.dma.unina.it
la somma di 10.000000 e 2.000000 = 12.000000
Downloading Output from Server pclab16.dma.unina.it
la somma di 4.000000 e 8.000000 = 12.000000
Downloading Output from Server pclab14.dma.unina.it
la somma di 3.000000 e 6.000000 = 9.000000
. . .
. . .

Vengono ricevuti i risultati di tutti i problemi

osservazione

L' insieme delle funzione
netsl_nb, netsl_pr e netsl_wt
permette di introdurre il parallelismo
in NetSolve in maniera flessibile, ma bisogna verificare in maniera esplicita se il risultato e' pronto

Esiste un meccanismo altrettanto asincrono
ma che lascia a NetSolve la gestione completa
del parallelismo?

Farming

Il farming e' un meccanismo per gestire numerose richieste dello stesso problema con una sola chiamata a NetSolve

E' NetSolve che controlla quando le soluzioni sono disponibili e le scarica dai server

meno generale di netslnb ma piu' efficiente

```
netsl_farm ( range, problema, arg1, arg2, . . .)
                                    rit: 0 se ok, -1 se errore
```

Numero di problemi da risolvere. E' una stringa del tipo "i=0,9"

Nome del problema da risolvere. E' una stringa analoga a quella usata in netsl( )