Il calcolo di Pi greco

Luca Simon e Roberto Carrer

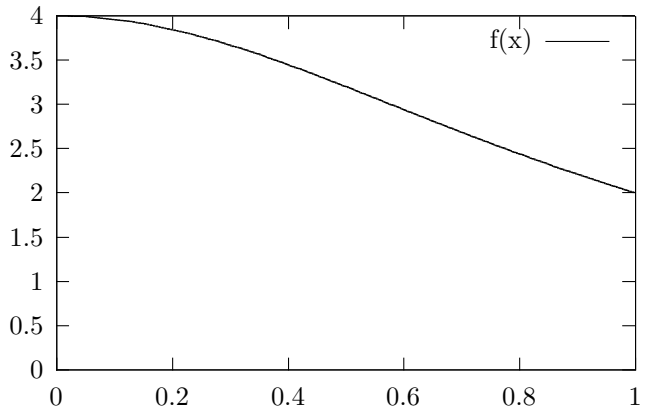
Questo esempio si propone di calcolare il valore di π usando il metodo di integrazione numerica. Partendo dalla formula

$$\int_0^1 \frac{1}{1+x^2}dx = arctan(x)\Big|_0^1 = arctan(1) - arctan(0) = arctan(1) = \frac{\pi}{4}$$

possiamo ottenere il valore di π integrando numericamente.

$$\pi = 4\int_0^1 \frac{1}{1+x^2}dx$$

Utilizzando il metodo dei rettangoli,si divide l'intervallo $[0,1]$ in n sottointervalli di ampiezza $\frac{1}{n}$ e si somma il valore delle aree dei rettangoli di base $\frac{1}{n}$ e altezza $f(x)$.



Ecco il codice:

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f(double a)
{
return (4.0 / (1.0 + a*a));
}
```

La funzione da integrare $f(x) = \frac{4}{1+x^2}$

```
int main(int argc,char *argv[])
{
 int done = 0, n, myid, numprocs, i;
 double PI25DT = 3.14159265358979323846264 3;
 double mypi, pi, h, sum, x;
 double startwtime = 0.0, endwtime;
 int  namelen;
 char processor_name[MPI_MAX_PROCESSOR_NAME];
```

La costante PI25DT contiene un valore di π di riferimento che sarà usato per valutare l'errore di approssimazione.

```
 MPI_Init(&argc,&argv);
 MPI_Comm_rank(MPI_COMM_WORLD,&myid);
```

Usuali chiamate di inizializzazione dei processi.

```
if (argc<=1) {
printf("Insert the parametres\n");
fflush(stdout);
MPI_Finalize;
return 0;
}
else
 n = atoi(argv[1]);
 fprintf(stdout,"Process %d of %d is on %s\n",myid,
  numprocs, processor_name);
 fflush(stdout);
```

Controlla che l'utente abbia inserito il numero di suddivisioni dell'intervallo richieste e poi stampa messaggi sul numero e nome dei processi coinvolti nel calcolo.

```
if (myid == 0)
startwtime = MPI_Wtime();
```

Il processo 0 fa partire il tempo.

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

Il processo 0 manda a tutti gli altri il numero di rettangoli (n).

```
h   = 1.0 / (double) n;
sum = 0.0;
```

Tutti calcolano la base dei rettangoli e inizializzano l'accumulatore dell'area.

```
for (i = myid + 1; i <= n; i += numprocs)
{
x = h * ((double)i - 0.5);
sum += f(x);
}
```

Il ciclo di calcolo dell'area dei rettangoli viene eseguito da tutti i processi rispettivamente nei propri segmenti di competenza che sono determinati dal numero di processo (myid); l'incremento del contatore di ciclo (i) è funzione del numero di processi attivi (numprocs) e quindi ciascuno sommerà n elementi distribuiti alternativamente in numprocs fasce; questa metodologia di assegnazione dei compiti ai singoli processi viene detta di *interleaving*.

```
mypi = h * sum;
```

Il singollo processo calcola l'area di sua competenza.

```
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

Questa funzione raccoglie tutti i valori parziali (anche del processo 0) contenuti nella variabile mypi, li somma e li invia nella variabile pi del processo 0.

```
if (myid == 0) {
endwtime = MPI_Wtime();
printf("pi is approximately %.16f, Error is %.16f\n",
pi, fabs(pi - PI25DT));
printf("wall clock time = %f\n", endwtime-startwtime);
fflush(stdout);
}
```

Il proceso 0 ferma il tempo e stampa i risultati.

```
MPI_Finalize();
return 0;
}
```

Funzione di chiusura del calcolo parallelo e ritorno del main().

Il programma può essere provato collegandosi al cluster ed eseguendo la demo corrispondente.