Esercizi MPI-Fortran

Davide Giacca -:- *dave@davethewave.it*

0.1 Questioni di base

Ping Pong
Ogni processo deve mandare il proprio id al processo successivo e stampare a video

```
Sono il processo numero X e ho ricevuto il messaggio Y
```

Difficoltà: per bambini

Provare ed evitare il deadlock
Provare con due processi:

```
Se il mio id=0 allora
 manda la variabile a al proc 1
 ricevi la variabile b dal proc 1
Se il mio id=1 allora
 ricevi la variabile a dal proc 0
 manda la variabile b al proc 0
```

Difficoltà: per tutti

0.2 Riduzioni e alberi

Scrivere un programma che:

- per ogni processo prenda a caso un numero tra 0 e 100 nella variabile A
- sommi e moltiplichi tutte le variabili A locali e le memorizzi nelle variabili SOMMA e PRODOTTO locali del processo 0
- stampa a video di SOMMA e PRODOTTO

usando solo send e receive bloccanti.
Domanda: quale è il metodo migliore per fare la riduzione? La somma e la moltiplicazione sono associative: la strategia SOMMA=A_0+A_1+... si completa in N passi.
Invece usando una riduzione ad albero binario (sommando a coppie...) il calcolo si completa in $log_2 N$ passi. Provare con 16 processi

```
mpirun -np16 nomeprogramma
```

Difficoltà: simpatico

0.3 Function Decomposition Vs. Domain Decomposition

Scrivere un programma che:

- crei una matrice A(50x50) di dati casuali di tipo REAL compresi tra 0 e π

- calcoli
$$\forall n \;\; C[i] = cos(A[n][i])$$
e
$$\forall n \;\; S[i] = sin(A[i][n])$$
dove C e S sono vettori di dimensione 50.

- Verificare che a ogni passo
$$C[i]^2 + S[i]^2 = 1$$

usando solo send e receive bloccanti.
Domanda: Quale è il migliore metodo per suddividere il lavoro tra i processi: separare i dati o separare le funzioni? e se la matrice fosse 50000x50000?
Difficoltà: eh... mica male...

0.4 Chiamate punto-punto non bloccanti

Nel programma precedente, inserite al posto delle chiamate bloccanti quelle non bloccanti
Domanda: è variato il tempo di calcolo? e se le funzioni da calcolare fossero più complicate?
Difficoltà: beh... fatto quello sopra...

0.5 Legge di Amdhal

Ora che potete vedere il tempo di calcolo, provate a eseguire il programma con

```
mpirun -npX nome_programma
```

con X che varia da 1 a 6. Salvate il risultato del tempo di calcolo su un file.
Domanda: Il programma segue la legge di Amdhal? Al variare del numero di processori, come si comporta?
Difficoltà: beh... fatto quello sopra... e quello sopra ancora... è facile!

0.6 Chiamate uno-molti, molti-uno, molti-molti

- Scrivere un programma che, data una matrice A(50x50) ne calcoli la trasposta ${}^{T}A$

- Scrivere un programma che calcola il prodotto di due matrici REAL A(50x50) e B(50x50) e lo memorizzi nella matrice C(50x50).

Usate chiamate punto-punto e collettive.
Domanda: quale tipo di chiamate semplifica *molto* il programma? quali sono più veloci?
Difficoltà: per jedi...