Implementazione operazione "saxpy" in ambiente MPI-Docker ($c = \alpha \cdot a + b$)

Denny Caruso[1]

1: denny.caruso001@studenti.uniparthenope.it

1. Definizione e analisi del problema

Si vuole realizzare un prodotto software che implementi in *parallelo* l'operazione nota come *saxpy* in ambiente MIMD-DM mediante l'utilizzo della libreria MPI. In particolare, si vuole determinare il vettore risultato c ottenuto come la somma fra il vettore b e il vettore a che viene moltiplicato per uno scalare α. Le operazioni sono elementari dal punto di vista dell'algebra lineare e fondamentalmente consistono nell'effettuare il prodotto di un vettore per uno scalare e successivamente il vettore risultante da questa operazione viene sommato a un altro vettore. Ovviamente i due vettori di input ed il vettore di output sono delle stesse dimensioni.

L'algoritmo sequenziale può essere banalmente realizzato con un singolo *loop* che effettua tante iterazioni quanti sono gli elementi in input in uno dei due vettori in input, all'interno di ognuna delle quali si eseguirà la somma di un vettore con l'altro vettore che viene anticipatamente moltiplicato per uno scalare.

Keywords: MPI, *saxpy*

2. Descrizione dell'approccio parallelo

Approfondire la scelta della strategia di parallelizzazione: motivare la decisione, descrivere la strategia in questione quanto più precisamente possibile, anche con l'aiuto anche di schemi se occorre. Valutare l'efficienza dell'approccio parallelo considerato con le metriche standard: speed-up, overhead, efficienza, Ware-Amdhal, isoefficienza.

3. Descrizione dell'algoritmo parallelo

Descrivere il proprio algoritmo nel dettaglio, riportando possibilmente i passi salienti in pseudo-codice e spiegando le scelte implementative (es. come assegnare dimensioni del sotto- problema, come gestire i processori/core, se e quando è necessario effettuare controlli, ecc.).

4. Input e Output

Descrivere cosa è necessario dare in input al software al momento dell'utilizzo, come parametri e/o quando richiesto interattivamente dal software stesso. Spiegare che output aspettarsi, per i diversi tipi di input. Per entrambi spiegare la forma in cui i dati devono essere forniti o in cui le informazioni verranno restituite (specificare il tipo dei dati, descrivere i file eventualmente prodotti, spiegare come interpretare l'output a video). Illustrare delle situazioni di errore previste dal software.

5. Routine implementate

Il software potrà essere composto da una o più routine. Alcune implementate dal programmatore, altre predefinite e appartenenti a librerie non standard del C, come API di MPI o OpenMP. È necessario illustrarle.

6. Analisi delle performance del software

Prendere i tempi d'esecuzione e riportarli in tabelle e grafici significativi, al variare della dimensione dell'input e del numero di processori/core impiegati. Corredare lo studio anche con grafici di speed-up ed efficienza.

6. Esempi d'uso

Riportare esempi di esecuzione del software, così come appare a video. Se ci sono casi particolari o casi limite, riportare almeno un esempio.

7. Riferimenti bibliografici

Se si è utilizzato materiale per studiare, o come riferimento per scrivere descrizioni e commenti, riportate in questa sezione libri, appunti di lezione, slide, articoli, siti web da cui questo materiale proviene, dove possibile specificando titolo ed autore.

8. Appendice

Riportare il codice scritto, compresa la DOCUMENTAZIONE INTERNA: commentate opportunamente il codice perché sia di facile lettura e comprensione per chi lo analizza, che ne potrà dare così migliore valutazione. Per un eventuale approfondimento, si consiglia di consultare il sito:
http://www.nag.co.uk/numeric/FD/manual/html/FDlibrarymanual.asp
dove sono disponibili documentazioni esterne delle routine di una libreria (parallela) commerciale.