Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Relazione di CALCOLO PARALLELO

Algoritmo Knuth-Morris-Pratt

Autori

Raffaele Di Nardo Di Maio 1204879

Cristina Fabris 1205722

Anno Accademico 2018-2019

Indice

Capitolo 1

Descrizione del Problema

L'algoritmo analizzato si occupa di effettuare un pattern matching. In forma generale questo problema può essere formalizzato nel seguente modo: definito un insieme Σ di elementi, detto *alfabeto*, trovare un'occorrenza di una sequenza di elementi

$$\{x_0\, x_1 \ldots x_{n-1} \mid x_i \in \Sigma \;\; 0 \leq i \leq n-1\}$$

detta *stringa*, all'interno di un più ampio testo formato sempre da elementi appartenenti all'alfabeto.
Nel nostro caso l'algoritmo riceve in input la stringa e un testo e restituisce come output l'indice corrispondente alla posizione del primo carattere della prima occorrenza in cui è possibile trovare, all'interno del testo, la stringa data.
Negli anni sono stati studiati ed implementati diversi algoritmi che si occupano di questa ricerca poiché si tratta di un problema che trova applicazioni negli ambiti più vari. Per citare solo un esempio, nella bioinformatica è applicato nella ricerca di una determinata sequenza di basi all'interno del DNA.
L'algoritmo brute force per la ricerca di stringhe scandisce ripetutamente il testo, ogni volta iniziando dall'elemento successivo a quello utilizzato all'inizio dell'iterazione precedente. Ad ognuna di queste confronta gli elementi del testo e quelli della stringa per vedere se è presente un'occorrenza. Al primo confronto che restituisce esito negativo passa all'iterazione successiva.
Quest'algoritmo è poco efficiente, in quanto non tiene in nessun modo conto delle informazioni acquisite dai confronti nelle iterazioni precedenti a quella attualmente in esecuzione.
Un algoritmo più complesso ed efficiente è, invece, quello di Knuth Morris Pratt (KMP). Questo è stato sviluppato da Knuth e Pratt e indipendentemente da Morris nel 1975. La maggior efficienza si verifica grazie al fatto che cerca di diminuire il numero di confronti necessari alla ricerca. Infatti, sfrutta una tabella che viene computata all'inizio del procedimento, nella quale sono salvate le posizioni successive a quella corrente, in cui è possibile trovare un'occorrenza della sequenza cercata.
In questo modo gli elementi del testo precedentemente verificati non vengono ricontrollati, sfruttando così tutte le informazioni acquisite precedentemente e diminuendo di molto il numero di confronti rispetto all'algoritmo brute force.
L'algoritmo che andremo ad analizzare è una particolare implementazione dell'algoritmo KMP che sfrutta la ricerca in parallelo su più sezioni del testo iniziale.

Infatti, divide quest'ultimo in più parti (pari al numero di processi che si vogliono attuare in parallelo) e all'interno di ogni set di elementi dell'alfabeto che si vengono a creare cerca la stringa richiesta. Inoltre, per la ricerca di occorrenze a cavallo tra più set, applica una particolare accortezza, sfruttando anche in questo caso il lavoro di più processi in parallelo.

Capitolo 2

Descrizione dell'algoritmo

Capitolo 3

Analisi della complessità computazionale

Capitolo 4

Analisi sperimentale

4.1 Tempi totali d'esecuzione e speed up

4.2 Tempi d'esecuzione al variare della taglia dell'input

4.3 Impatto delle comunicazioni

4.4 Test con parametri di input asimmetrici

Capitolo 5

Conclusioni

Bibliografia