Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Relazione di CALCOLO PARALLELO

BitonicSort VS QuickSort

Autori

Matteo Perin ******

Davide Pistilli ******

Flavio Spanò 1197697

Anno Accademico 2018-2019

Indice

Capitolo 1

Descrizione del Problema

L'algoritmo analizzato effettua un pattern matching, ovvero, ricevendo in input un stringa e un testo in cui volerla ricercare, ne calcola il numero totale di occorrenze. Il problema può essere formalizzato in questo modo: definito un insieme Σ di elementi, detto alfabeto, trovare il numero di occorrenza di una sequenza di elementi $\{x_0\, x_1\, ...\, x_{n-1} \;\mid\; x_i \in \Sigma \;\; 0 \leq i \leq n-1\}$, detta stringa, all'interno di un più ampio testo formato sempre da elementi appartenenti all'alfabeto.
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 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 dell'iterazione precedente. Ad ogni iterazione confronta gli elementi del testo e quelli della stringa per vedere se è presente un'occorrenza di quest'ultima.
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), sviluppato da Knuth e Pratt e indipendentemente da Morris nel 1975. Questo cerca di diminuire il numero di confronti necessari alla ricerca. Infatti, grazie ad una tabella che viene salvata all'inizio del procedimento, sono note 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, andando a diminuire 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, fornito in input, 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 (anche questa fornita in input). 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