Sistemi Distribuiti 1 2003-2004

Esercitazione Sistemi Distribuiti 1

(Consegna: Lunedì 27 Settembre 2004)

L'esercitazione prevede l'implementazione dell'algoritmo sistolico per il prodotto di matrici quadrate di dimensione n in ambiente distribuito usando `Java`. Scopo dell'esercitazione è impratichirsi con i concetti di thread, socket e concorrenza (sincronizzazione, comunicazione, etc.).

Nonostante l'algoritmo non sia pensato per essere realizzato in `Java`, ha il pregio di permettere di affrontare l'utilizzo della maggior parte dei concetti presentati a lezione.

Una volta che l'algoritmo è stato implementato, si dovrà essere testare su una coppia di matrici arbitrarie di dimensione $n = 16$ e si dovrà misurare l'andamento dei tempi di esecuzione in relazione alla scalabilità del sistema (cioè all'aumentare del numero di calcolatori coinvolti). Riepilogando, viene richiesto:

- implementaziozione in `Java` dell'algoritmo sistolico
- misure sperimentali relative alla scalabilità

Prodotto di Matrici

Il prodotto di due matrici $A = (a_{ij})$ e $B = (b_{ij})$ con $i, j = 1, \ldots, n$ è un problema di facile soluzione. A partire dalla definizione matematica:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

si ricava facilmente il ben noto algoritmo sequenziale. L'algoritmo sequenziale ha complessità temporale in $O(n^3)$ ed è facile immaginare che per matrici di grandi dimensioni la durata dell'esecuzione aumenti drasticamente rendendo l'approccio sequenziale impraticabile. Con l'introduzione della possibilità di suddividere il carico computativo tra più *processing element* sono stati proposti diversi algoritmi distribuiti che risolvono questo problema riducendone la complessità temporale e quindi il tempo di calcolo.

Prodotto di Matrice: Algoritmo Sistolico

L'algoritmo sistolico è un algoritmo per la moltiplicazione di matrici in ambiente parallelo-distribuito. La parola "sistolico" è legata alla parola "sistole" e si riferisce

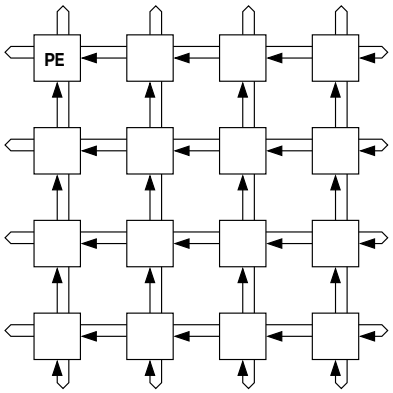

Figura 1: **Architetture adottate dai due algoritmi.**

al battito ritmico del cuore che l'algoritmo imita. Il prodotto delle matrici $(n \times n)$ A e B è calcolato in passi incrementali chiamati "battiti". Le matrici A e B ed il loro prodotto C sono memorizzate in un toroide (vedere figura 1) composto da $n \times n$ processing element; PE_{ij} inizialmente immagazzina gli elementi di posizione (i, j) di ogni matrice. L'algoritmo consta di due fasi:

- preskewing, ed
- esecuzione sistolica.

Il preskewing è la fase di inizializzazione, in cui gli elementi dell'i-sima riga di A e dell'i-sima colonna di B sono spostati di i posizioni, rispettivamente, ad ovest ed a nord. Inoltre gli elementi di C sono inizializzati come segue:

$$c_{ij} = a_{ij} \cdot b_{ij}$$

L'esecuzione sistolica consiste di $n-1$ passi (battiti). Ad ogni passo, gli elementi di A e di B sono spostati attraverso il toroide. L'elemento di A memorizzato nel processing element PE_{ij} è inviato al processing element collegato ad ovest di PE_{ij}; gli elementi di B, invece, sono inviati a nord. Tutti questi spostamenti sono fatti contemporaneamente. Avvenuti gli spostamenti, il processing element PE_{ij} calcola il valore di c_{ij} moltiplicando gli elementi che ha appena ricevuto di A e di B e sommando il risultato al valore precedentemente calcolato di c_{ij}.

Traccia dell'Architettura del Sistema

In questa sezione introdurremo alcune note sulla realizzazione del progetto che dovrete seguire, tradotto sappiamo che può essere sviluppato o progettato diversamente e sicuramente meglio ma la strada proposta è quella che permette di impratichirsi con i concetti presentati a lezione.

I *processing element* descritti sono caratterizzati da tre attributi: il valore che si sta calcolando, un elemento della matrice A ed un elemento della matrice B. Tutti questi attributi cambiano durante l'esecuzione.

I *processing element* devono poter risiedere su macchine diverse oppure essere sulla stessa macchina. Se risiedono sulla stessa macchina saranno thread generati dalla stessa istanza di processing element, processing element su macchine diverse sono istanze distinte della classe rappresentante i processing element.

Ovviamente la natura dei processing element (o thread o processi separati) implicherà la necessità di supportare meccanismi di comunicazione distinti. I processing element PE_{ij} e PE_{kl} si scambieranno i valori di A e di B come segue:

- tramite variabile condivisa (con i soliti problemi di mutua esclusione) se PE_{ij} e PE_{kl} sono thread generati dalla stessa istanza;
- tramite socket altrimenti.

Dovreste predisporre un processo factory (o più se usate più macchine) che generi ed inizializzi i vari processing element e dia il battito di inizio.