Algoritmi paralleli e distribuiti

Indice

1. Introduzione

1.1. Definizione

Un **algoritmo** è una sequenza finita di istruzioni che non sono ambigue e che terminano, ovvero restituiscono un risultato

Gi **algoritmi sequenziali** avevano un solo esecutore, mentre gli algoritmi di questo corso utilizzano un **pool di esecutori**

Le problematiche da risolvere negli algoritmi sequenziali si ripropongono anche qua, ovvero:
- **progettazione**: utilizzo di tecniche per la risoluzione, come *Divide et Impera*, *programmazione dinamica* o *greedy*
- **valutazione delle prestazioni**: complessità spaziale e temporale
- **codifica**: implementare con opportuni linguaggi di programmazione i vari algoritmi presentati

I programmi diventano quindi una *sequenza di righe*, ognuna delle quali contiene *una o più* istruzioni

1.2. Algoritmi paralleli

Un **algoritmo parallelo** è un algoritmo **sincrono** che risponde al motto *"una squadra in cui batte un solo cuore"*, ovvero si hanno più entità che obbediscono ad un clock centrale, che va a coordinare tutto il sistema

Abbiamo la possibilità di condividere le risorse in due modi:
- memoria, formando le architetture
  - **a memoria condivisa**, ovvero celle di memoria fisicamente condivisa
  - **a memoria distribuita**, ovvero ogni entità salva parte dei risultati parziali sul proprio nodo
- uso di opportuni collegamenti

Qualche esempio di architettura parallela:
- **supercomputer**: cluster di processori con altissime prestazioni
- **GPU**: usate in ambienti grafici, molto utili anche in ambito vettoriale
- **processori multicore**
- **circuiti integrati**: insieme di gate opportunamente connessi

1.3. Algoritmi distribuiti

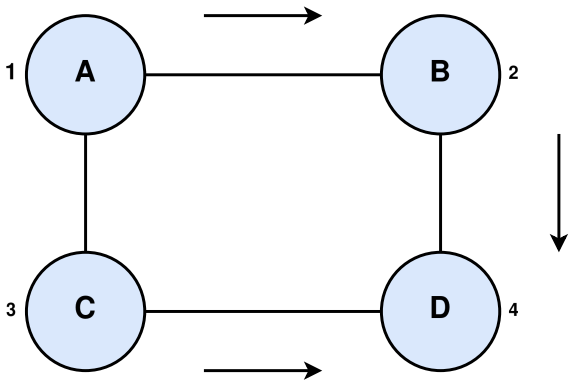
Un **algoritmo distribuito** è un algoritmo **asincrono** che risponde al motto *"ogni membro del pool è un mondo a parte"*, ovvero si hanno più entità che obbediscono al proprio clock personale

Abbiamo anche in questo caso dei collegamenti ma non dobbiamo supporre una memoria condivisa o qualche tipo di sincronizzazione, quindi dobbiamo utilizzare lo **scambio di messaggi**

Qualche esempio di architettura distribuita:
- **reti di calcolatori**: internet
- **reti mobili**: uso di diverse tipologie di connessione
- **reti di sensori**: sistemi con limitate capacità computazionali che rispondono a messaggi *ack*, *recover*, *wake up*, eccetera

1.4. Differenze

Vediamo un problema semplicissimo: *sommare quattro numeri A,B,C,D*



Usiamo la primitiva `send(sorgente,destinazione)` per l'invio di messaggi

Un approccio parallelo a questo problema è il seguente

SOMMA DI QUATTRO NUMERI(A, B, C, D):

1 send(1,2), send(3,4)
2 A+B, C+D
3 send(2,4)
4 A+B+C+D

Un approccio distribuito invece non può seguire questo pseudocodice, perché le due send iniziali potrebbero avvenire in tempi diversi

Notiamo come negli algoritmi paralleli ciò che conta è il **tempo**, mentre negli algoritmi distribuiti ciò che conta è il **coordinamento**