Implementazione di algoritmi paralleli per la soluzione di Sudoku

Gaetano Conti, Vincenzo Imperati

Sommario

“I computer sono incredibilmente veloci, accurati e stupidi. Gli uomini sono incredibilmente lenti, inaccurati e intelligenti. L’insieme dei due costituisce una forza incalcolabile.”

- ALBERT EINSTEIN -

01

INTRODUZIONE

INTRODUZIONE

Approccio multithreading alla risoluzione di sudoku 9x9

Vengono discusse due soluzioni:

- Pthread: DFS in parallelo su più thread
- OpenMP: BFS parallelizzando alcune parti del codice

INTRODUZIONE

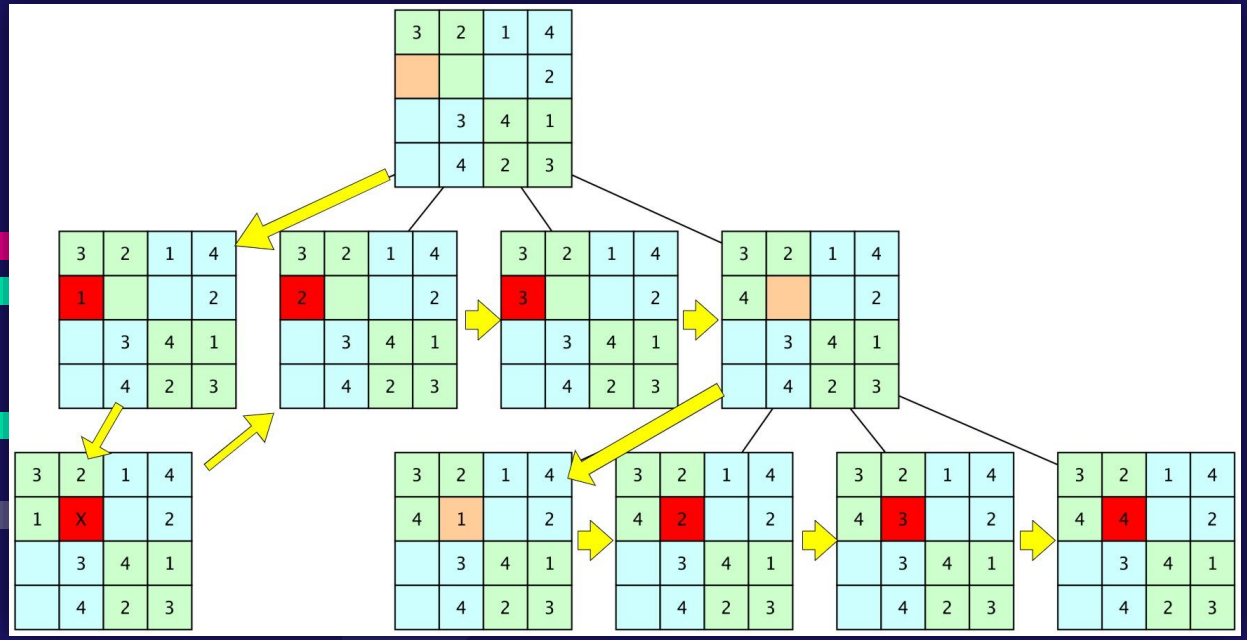
Nel documento vengono descritte:

- le architetture delle due soluzioni
- i problemi riscontrati
- le limitazioni riscontrate
- i test svolti su di un dataset comune di sudoku 9x9 con un'unica soluzione

02

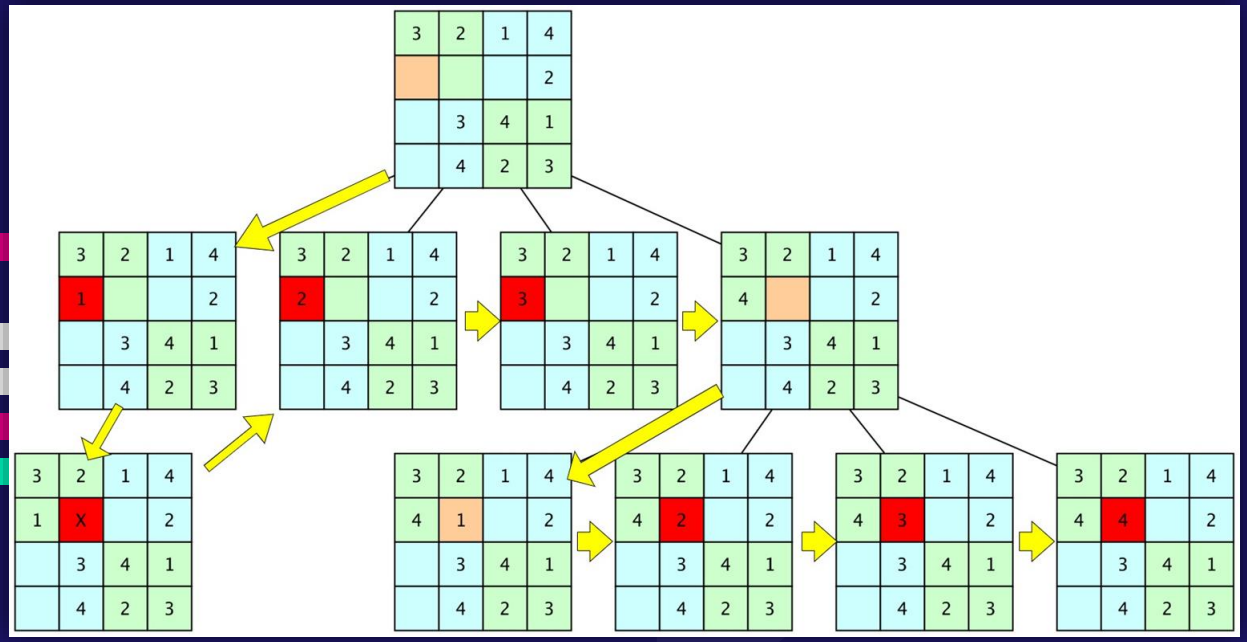
Soluzione Pthread

Soluzione Pthread



DFS - approccio vantaggioso qualora la soluzione si trovasse in profondità, invece che in superficie (dove performa meglio la BFS).

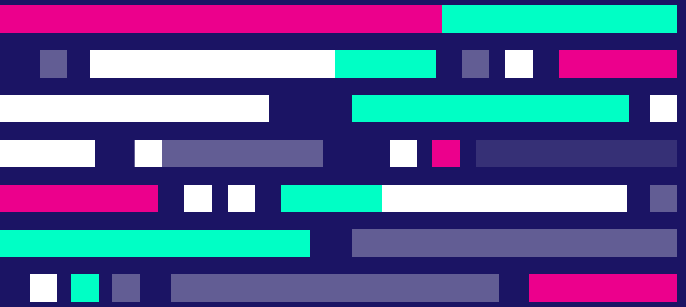
Soluzione Pthread



Lo scopo è trovare l'unica soluzione del sudoku svolgendo la DFS in modo parallelo sfruttando la potenza computazionale di più thread.

L'obiettivo è anche riuscire a distribuire in modo ottimale il lavoro computazionale che ogni thread deve svolgere.

Soluzione Pthread



L'algoritmo acquisisce in input:
- il numero di thread che è chiamato ad utilizzare
- il sudoku da risolvere

Durante l'acquisizione del sudoku viene riempita la griglia di 81 celle e vengono memorizzate in una lista le coordinate delle celle vuote

Questa lista ha lo scopo di identificare le celle vuote senza il bisogno di leggere la griglia

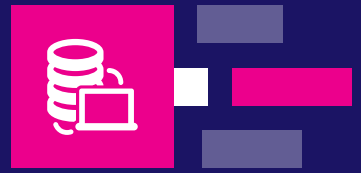
VARIABILI PRINCIPALI

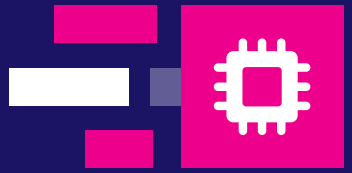
problem_solved

Intero che indica se un sudoku valido è stato trovato

sudokus_to_solve

Lista di sudoku da validare, uno per ogni thread



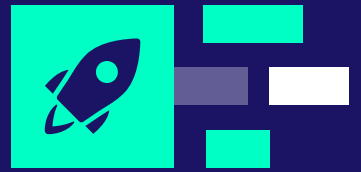


semaphores

Lista di semafori, uno per ogni thread

solved_sudoku

Sudoku che conterrà la grid valida da mandare in output



VARIABILI PRINCIPALI

mutex_shared_data

Mutex per accedere in mutua esclusione alla FIFO

shared_fifo_idle_threads

FIFO condivisa tra i vari thread

mutex_solved_sudoku

Mutex per accedere in mutua esclusione alla scrittura del sudoku che conterrà una grid valida

POSSIBILI STATI DI UN THREAD

Idle validator

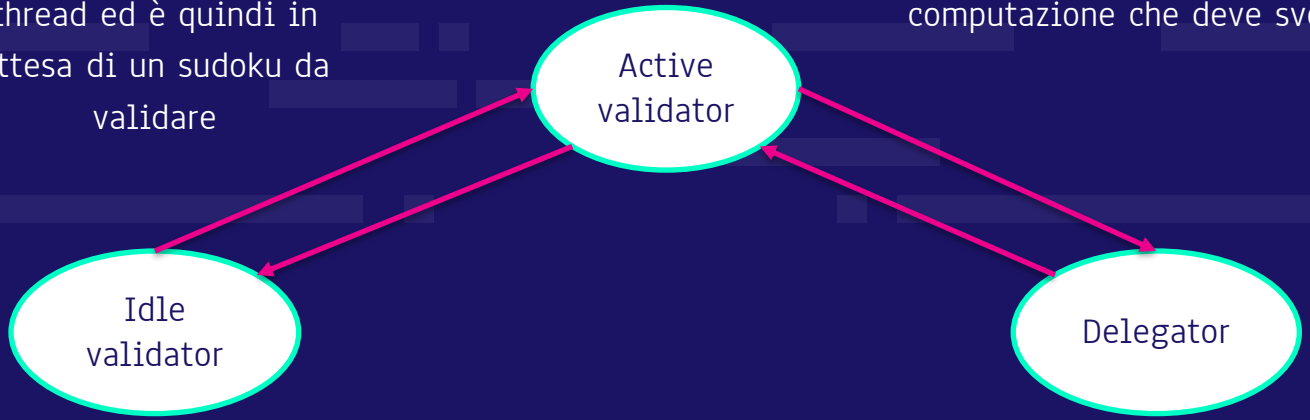
non sta validando nessun sudoku. Esso è visibile nella FIFO dai Delegator thread ed è quindi in attesa di un sudoku da validare

Active validator

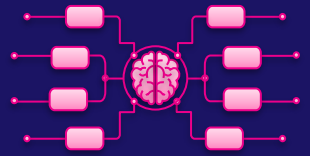
sta validando il sudoku che gli è stato assegnato

Delegator

Active validator che ha la possibilità di delegare ad un Idle Validator una parte di computazione che deve svolgere



FUNZIONAMENTO



L'algoritmo lancia i thread che sono tutti Idle validator

Verrà assegnato ad un thread il sudoku acquisito in input come sudoku da validare

Questo thread è il primo a diventare Active Validator

Il singolo thread si ritrova a dover validare un gruppo di sudoku, così prima di intraprendere qualsiasi validazione controlla se c'è un Idle Validator che attende un sudoku da validare

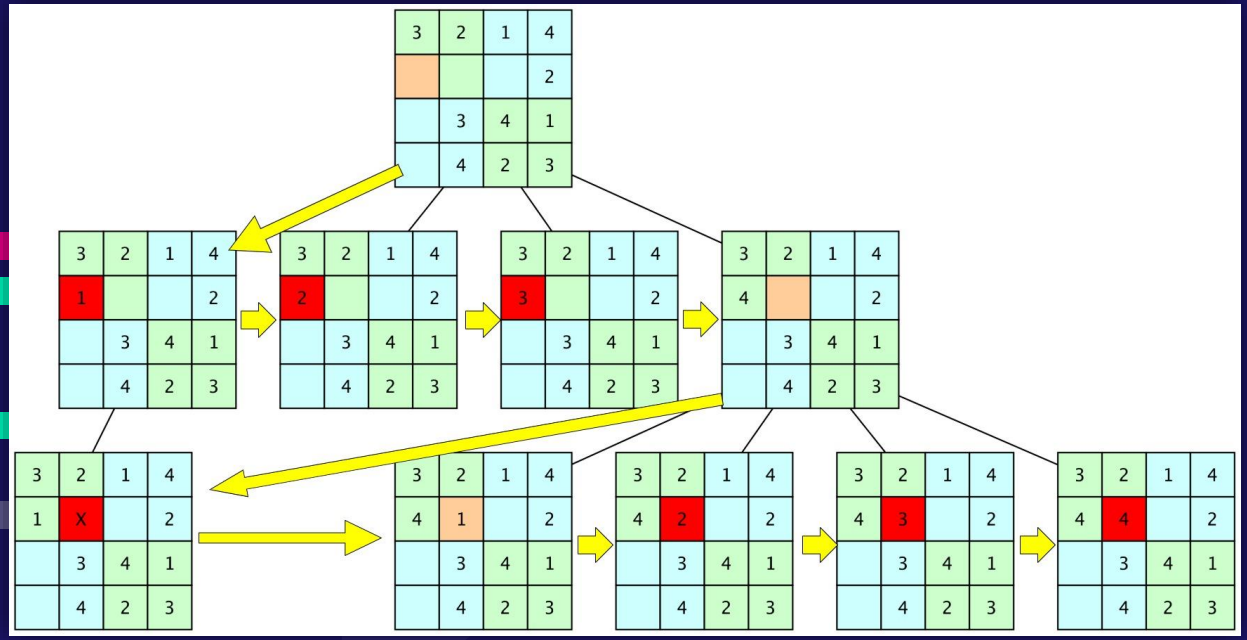
Nel caso questo fosse disponibile, il thread che possiede i sudoku da validare diventa Delegator e delega la validazione di un sudoku al primo Idle Validator in attesa

L'algoritmo termina quando un Active Validator riesce a riempire l'ultima cella vuota di un sudoku, che gli è dato da validare, con un numero valido

03

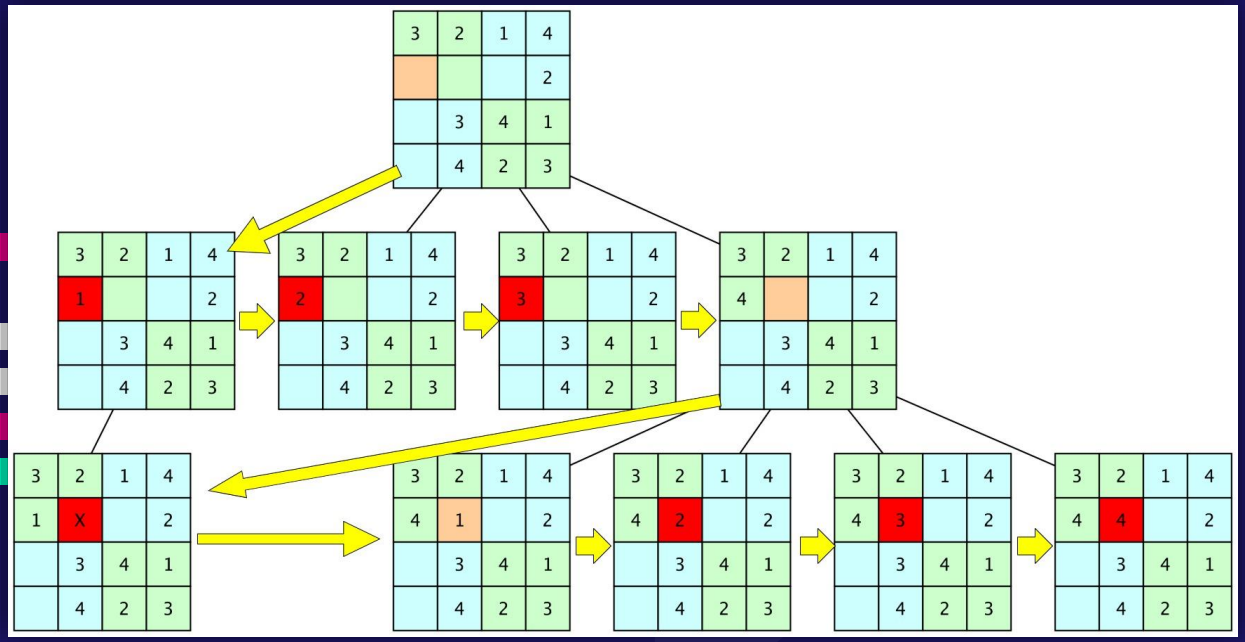
Soluzione OpenMP

Soluzione OpenMP



BFS - approccio vantaggioso qualora la soluzione si trovasse in superficie, invece che in profondità (dove performa meglio la DFS)

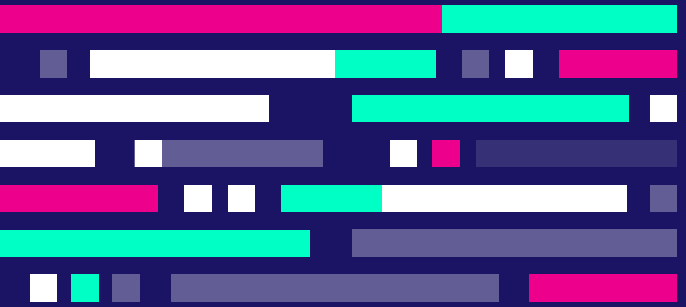
Soluzione OpenMP



Lo scopo è trovare l'unica soluzione del sudoku svolgendo la BFS in modo parallelo sfruttando la potenza computazionale di più thread

L'obiettivo è anche riuscire a distribuire in modo ottimale il lavoro computazionale che ogni thread deve svolgere

Soluzione OpenMP



L'algoritmo acquisisce in input:
- il numero di thread che è chiamato ad utilizzare
- il sudoku da risolvere

Durante l'acquisizione del sudoku viene riempita la griglia di 81 celle e vengono memorizzate in una lista le coordinate delle celle vuote

Questa lista ha lo scopo di identificare le celle vuote senza il bisogno di leggere la griglia

VARIABILI PRINCIPALI

solved_sudoku

Intero che se diverso da zero indica che un sudoku valido è stato trovato

list

Lista contenente tutti i sudoku da validare. Inizialmente contiene solo il sudoku acquisito in input

index_list

Indice di list, ci permette di operare sulla list

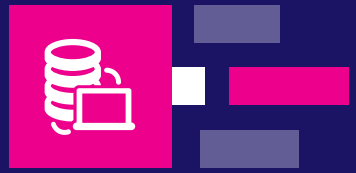
VARIABILI PRINCIPALI

new_list

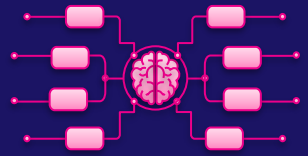
Lista contenente il nuovo livello di sudoku da validare

new_index_list

Indice di new_list, ci permette di operare sulla new_list



FUNZIONAMENTO



Attraverso una BFS che visita il livello successivo dell'albero si crea la new_list tramite l'ultimo livello visitato rappresentato da list

La new_list viene riempita di sudoku da validare inseriti dai thread che parallelizzano questo inserimento

I thread sono chiamati a parallelizzare il doppio ciclo di for che permette la compilazione di new_list

Una volta che un thread riesce a riempire l'ultima cella vuota di un sudoku notifica tramite solved_sudoku che è stato trovato un sudoku valido e scrive questo sudoku nella memoria occupata dal sudoku che rappresenta la soluzione

Alla fine della compilazione di new_list l'algoritmo può terminare

04

Testing

SPECIFICHE

Le due soluzioni proposte sono state testate su uno stesso dataset di 100 sudoku

I test sono stati effettuati su una macchina con una CPU Intel(R) Core(TM) i7-6500U e una RAM di 8 GB

Ogni soluzione è stata eseguita con le seguenti quantità di thread a disposizione: 1, 2, 4, 8, 16, 32

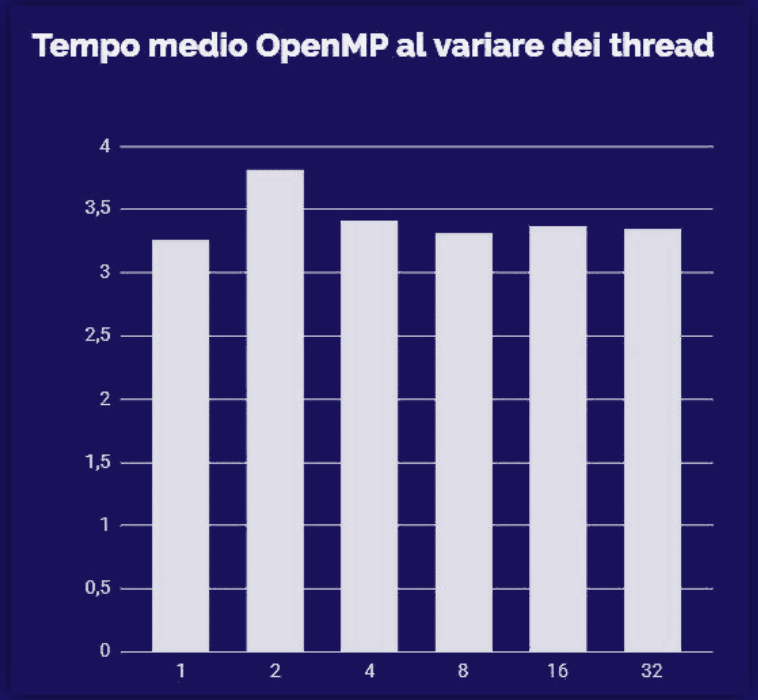
Test Pthread

Prestazioni

Si noti che l'esecuzione con due thread ha una prestazione migliore che con uno solo

All'aumentare del numero dei thread le prestazioni si vanno a deteriorare

Test OpenMP



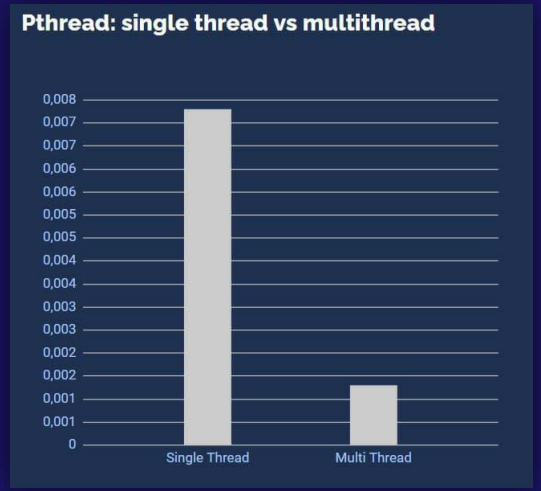
Prestazioni

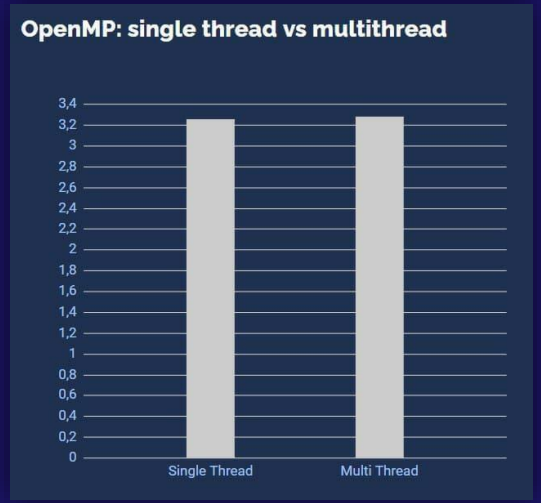
Si noti che non vi è un netto miglioramento prestazionale attraverso l'utilizzo di più thread

SINGLE THREAD VS MULTITHREAD

	Single thread	Multi thread
Pthread	0,0077	0,0017
OpenMP	3,2578	3,2827

Tutti i test sono stati eseguiti sullo stesso dataset





GRAZIE

Progetto a cura di:
Gaetano Conti e Vincenzo Imperati