Progetto di High Performance Computing 2024/2025

Enrico Marchionni, matr. 0001032322

17 dicembre 2024

1 Introduzione

...

2 Versione Seriale

L'algoritmo implementato è:

```c
int skyline(const points_t *points, int *s)
{
  const int D = points->D;
  const int N = points->N;
  const float *P = points->P;
  int r = N;

  for (int i = 0; i < N; i++) {
    s[i] = 1;
  }

  for (int i = 0; i < N; i++) {
    if (s[i]) {
      for (int j = 0; j < N; j++) {
        if (s[j] && dominates(&(P[i * D]), &(P[j * D]), D)) {
          s[j] = 0;
          r--;
        }
      }
    }
  }
  return r;
}
```

Listing 1: Algoritmo per il calcolo dello skyline in C

È l'algoritmo seriale proposto è tipico per soluzioni brute-force del problema dello skyline. Il costo computazionale dell'algoritmo in Listing 1 è in generale $O(N^2D)$. Infatti il ciclo esterno viene percorso N volte, nelle quali, nel caso peggiore, vengono eseguite altrettante N operazioni di confronto tra D elementi (da cui $N \times N \times D$). Si può notare che D può influire

anche molto nella complessità, in particolare se $D >> N^2$. Nel caso in cui $N^2 >> D$, si considera $O(N^2)$, nel caso in cui $D >> N^2$, $O(D)$, nell'ultimo caso in cui $D \approx N^2$ allora $O(N^2 D)$.

Caso	Complessità	Range
Pessimo	$\Theta(N^2 D)$	Tutti i punti sono dello skyline.
Medio	$\Theta(N^2 D)$	La metà dei punti fanno parte dello skyline.
Ottimo	$\Theta(ND)$	Il primo punto analizzato domina tutti gli altri.

Tabella 1: Input data.

Come si può vedere dalla Tabella 1 la complessità in realtà varia anche di molto in base all'input fornito. Fattore importante nella misura della weak scaling efficiency per esempio.

3 Versione OpenMP

...

4 Versione MPI/CUDA

...

5 Conclusioni

...