Corso di Algoritmi Avanzati—modulo 2

Terzo progetto di programmazione—Anno Accademico 2014/2015

Moreno Marzolla

Versione 1.0 del 20/7/2015

Prima versione

Descrizione del progetto

Scopo del progetto è l'implementazione di un algoritmo parallelo per determinare lo **skyline** di un insieme di punti nello spazio su architetture a memoria distribuita utilizzando il linguaggio C e MPI.

Consideriamo un insieme $P = \{p_0, p_2, \ldots p_{N-1}\}$ di N punti nello spazio a tre dimensioni; il punto p_i ha coordinate (x_i, y_i, z_i). Diciamo che p_i *domina* p_j se si verificano entrambe le condizioni seguenti:

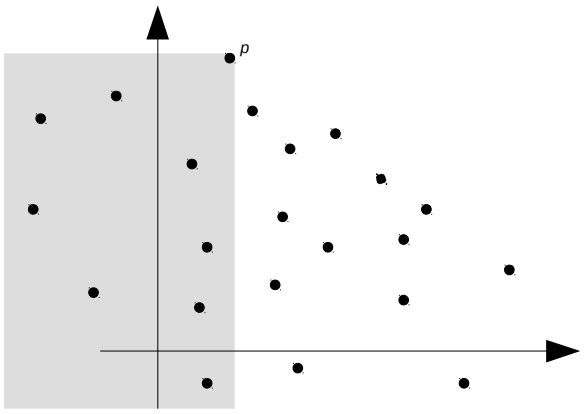
1. $x_i \geq x_j$, $y_i \geq y_j$ e $z_i \geq z_j$
2. Esiste almeno una coordinata (x, y, oppure z) per cui la disuguaglianza valga in senso stretto

(Si presti attenzione al fatto che in letteratura la relazione di dominanza viene talvolta definita con l'operatore $\leq$ anziché $\geq$ come facciamo noi). Detto in altri termini, p_i domina p_j se e solo se ogni coordinata di p_i ha valore maggiore o uguale alla corrispondente coordinata di p_j, ed esiste almeno una coordinata di p_i che abbia valore strettamente maggiore della corrispondente di p_j. Ad esempio, (3, 5, 4) domina (3, 4, 4); (3, 5, 4) domina (2, -1, 3). Si noti che data una coppia di punti, non è detto che uno dei due domini l'altro. Ad esempio, se s = (3, 5, 4) e t = (3, 6, 3), si ha the s non domina t e t non domina s.

Lo skyline $Sk(P)$ dell'insieme P è composto da tutti i punti che non sono dominati da alcun altro punto (si noti che per definizione un punto non domina se stesso):

$$Sk(P) = \{s \in P : \text{non esiste alcun punto } t \text{ in } P \text{ tale che } t \text{ domini } s\}$$

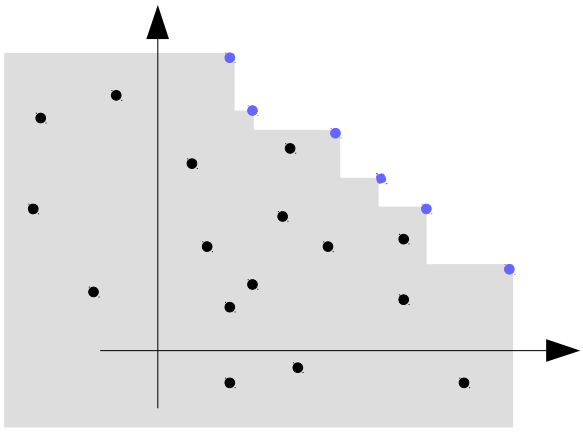
Per capire in cosa consiste l'operatore skyline, è utile considerare un semplice esempio in due dimensioni. La figura seguente rappresenta un insieme di punti sul piano:



I punti dominati da p sono quelli che si trovano all'interno dell'area grigia "proiettata" dal punto p verso il basso e verso sinistra; per definizione, nessuno dei punti nell'area grigia fa parte dello skyline, in quanto sono tutti dominati da un altro punto (p in questo caso).

Esistono diversi algoritmi per determinare lo skyline. Noi useremo l'algoritmo banale che procede per eliminazione: si considerano uno alla volta tutti i punti dell'insieme P, e per ogni punto p si eliminano da P tutti i punti da esso dominati. I punti che rimangono alla fine di questo processo di eliminazione sono tutti e soli quelli che fanno parte dello skyline. Nell'esempio in due dimensioni,

lo skyline è composta dai punti in blu:



L'area grigia è composta dall'unione delle proiezioni di tutti i punti presenti; quindi, i punti che cadono all'interno dell'area grigia sono dominati da qualche altro punto (in particolare, da almeno uno dei punti blu), e non possono far parte dello skyline.

L'input del programma da sviluppare consiste in tre array $x[0..N-1]$, $y[0..N-1]$ e $z[0..N-1]$ che rappresentano le coordinate di N punti nello spazio; lo pseudocodice seguente determina lo skyline applicando il meccanismo appena descritto:

```
SKYLINE( real x[0..N - 1], real y[0..N - 1], real z[0..N - 1])
        bool S[0..N - 1];
        integer i, j;
        for i ← 0 to N - 1 do
                S[i] ← true;
        endfor
        for i ← 0 to N - 1 do
                for j ← 0 to N - 1 do
                        if (x[i], y[i], z[i]) domina (x[j], y[j], z[j]) then
                                S[j] ← false;
                        endif
                endfor
        endfor
        for i ← 0 to N - 1 do
                if S[i] then
                        print x[i], y[i], z[i];
                endif
        endfor
```

Al termine dell'esecuzione, $S[i]$ vale *true* se e solo se il punto i-esimo fa parte dello skyline.

Scopo del progetto è implementare una versione parallela dell'algoritmo SKYLINE su architetture a memoria distribuita usando il linguaggio C e le librerie MPI. Si noti che lo pseudocodice dell'algoritmo sequenziale deve essere inteso come uno schema di massima, e non deve necessariamente essere implementato così com'è; in particolare, la scelta della strategia di partizionamento dei dati e le primitive MPI da utilizzare fanno parte integrante del progetto. Nella relazione è possibile motivare le scelte implementative, indicando i pro e contro dell'implementazione consegnata e le eventuali limitazioni.

Il processo master deve leggere le coordinate dei punti da un file di testo `in.txt` che si trova nella stessa directory dell'eseguibile. Il file `in.txt` è composto da $N+1$ righe di testo: la prima riga contiene il valore N (di tipo `long int`), e le successive N righe contengono le tre coordinate reali $x[i]$ $y[i]$ $z[i]$, separate da spazi (si puo' usare il tipo `float` o `double`, a scelta). Un esempio di file di input è presente nella pagina Web del modulo 2.

Al termine dell'esecuzione, il master deve salvare le coordinate dei punti dello skyline, in un ordine qualsiasi, in un file `out.txt` con lo stesso formato del file di input. Se lo skyline è composto da $K \leq N$ punti, il file `out.txt` conterrà $K + 1$ righe di testo; nella prima riga è presente il valore K, nelle successive K righe sono elencate le coordinate $x[i]$ $y[i]$ $z[i]$, separate da spazi, dei punti che fanno parte dello skyline.

Modalità di svolgimento del progetto

- Il progetto deve essere svolto individualmente. Non è consentito condividere codice o relazione con altri studenti, né utilizzare software disponibile in rete. Non è consentito discutere il progetto con altri.
- Il progetto deve essere realizzato in un singolo file sorgente chiamato `skyline.c`. Il sorgente deve essere adeguatamente commentato. All'inizio del file sorgente deve essere incluso un commento che riporta cognome, nome e numero di matricola dell'autore/autrice, nonché il comando da usare per la compilazione (vedi punto successivo).
- Il progetto deve essere implementato in linguaggio C come applicazione a riga di comando, facendo uso delle librerie MPI. Il progetto deve essere compilabile ed eseguibile senza errori su una delle macchine Linux dei laboratori studenti (sistema operativo Debian GNU/Linux 7.8), oppure sull'immagine dei laboratori virtuali (http://www.virtlab.unibo.it/index.html) mediante il comando

  ```
  mpicc -Wall skyline.c -o skyline [eventuali flag di compilazione]
  ```

  Indicare nel commento iniziale del sorgente e/o nella relazione la riga di comando corretta per la compilazione nel caso siano richiesti altri flag, ad esempio nel caso in cui sia necessario includere librerie di sistema. Nota: sulle macchine del laboratorio è installato OpenMPI (pacchetti `openmpi-bin`, `libopenmpi-dev` ed `openmpi-doc` per la documentazione).
- Per verificarne la correttezza, il programma verrà eseguito mediante il comando

  ```
  mpirun -n proc ./skyline
  ```

  dove *proc* indica il numero di processi MPI da creare. Tutte le istanze saranno lanciate localmente, in modo da evitare ogni problema con la rete.
- Si puo' assumere che il numero N di punti sia sempre (molto) maggiore del numero di processi MPI (ad esempio, $1{,}000 \leq N \leq 100{,}000$). Il programma deve funzionare correttamente anche se il numero di punti non è multiplo esatto del numero di processi MPI. Si puo' assumere che sia sempre possibile mantenere nella memoria locale di ciascun processo MPI le coordinate di tutti i nodi.
- Assieme al sorgente deve essere consegnare una **relazione** in formato PDF in un file chiamato `relazione.pdf`. La relazione non deve superare la lunghezza di **cinque facciate** in formato A4 (font non inferiore a 10 punti). **Nel computo delle cinque facciate rientrano TUTTE le facciate del documento consegnato** (pagina del titolo, eventuale indice o altro); suggerisco di non sprecare spazio dedicando una facciata al titolo o all'indice, che in un documento così corto sarebbero comunque superflui. Indicare il proprio cognome, nome, e numero di matricola all'inizio della relazione.
- La relazione deve descrivere ad alto livello l'implementazione con particolare riguardo alla strategia di parallelizzazione adottata e gli eventuali limiti e/o potenzialità di tale strategia.
- La relazione **deve obbligatoriamente includere una sintetica analisi sperimentale della**

scalabilità dell'algoritmo realizzato, mostrando i grafici di speedup ed efficienza. Non è richiesto che l'analisi di scalabilità venga effettuata sulle macchine del laboratorio. È possibile effettuare i test lanciando tutte le istanze localmente (anche in numero superiore al numero di core disponibili); ovviamente non terrò conto dei tempi di esecuzione in termini assoluti, ma solo del fatto che l'analisi di speedup ed efficienza sia stata svolta correttamente.

Scadenza e modalità di consegna

E' necessario consegnare:

1. Il file sorgente del programma realizzato;
2. La relazione in formato PDF, secondo le specifiche indicate sopra.

Il progetto e la relazione devono essere consegnati entro le ore **12:00 (mezzogiorno)** di **martedì 1 settembre 2015**.

Ricordo che è necessario ottenere una valutazione positiva del progetto per poter partecipare alla prova scritta del modulo 1. La valutazione verrà comunicata via mail, e se positiva resterà valida per l'intero anno accademico 2014/2015, ossia fino all'appello di esami di gennaio/febbraio 2016 (compreso).

La consegna deve avvenire inviando una mail dal proprio indirizzo istituzionale @studio.unibo.it a moreno.marzolla@unibo.it con subject

Consegna Progetto Algoritmi Avanzati 2014/2015

Nella mail vanno indicati cognome, nome, numero di matricola del mittente. Alla mail deve essere allegato un archivio in formato `.zip` oppure `.tar.gz` contenente il sorgente e la relazione in formato PDF; chi lo desidera può includere altri files, ad esempio un `Makefile` per la compilazione (non obbligatorio). L'allegato sarà denominato con il cognome dell'autore (es., `Rossi.zip` oppure `Rossi.tar.gz`), e conterrà una directory con lo stesso nome (es., `Rossi/`) contenente a sua volta il sorgente e la relazione. Riceverete una mail di conferma entro pochi giorni.

Nota per gli utenti Apple: gli archivi .zip prodotti sotto MacOSX spesso includono una directory `.DS_Store/`. L'archivio che viene consegnato **deve essere privo** di tale directory.

Valutazione dei progetti

Per ottenere una valutazione positiva è necessario che il programma compili sulle macchine Linux del laboratorio studenti e funzioni correttamente secondo le specifiche indicate in questo documento. Ulteriori elementi di cui si terrà conto:

- Efficienza della strategia di parallelizzazione adottata;
- Generalità della soluzione implementata;
- Qualità della relazione, in termini di chiarezza, correttezza e presentazione del contenuto;
- Qualità del codice, in termini di correttezza e chiarezza. Verrà penalizzato il ricorso a micro-ottimizzazioni inutili, nonché codice ridondante o inutilmente complesso. Verranno altresì penalizzati errori di programmazione (come ad esempio, accessi out-of-bound, memory leak, uso errato di variabili non inizializzate, eccetera).