Università di Bologna, Laurea in Ingegneria e Scienze Informatiche

Corso di High Performance Computing

Progetto d'esame 20242025

prof. Moreno Marzolla

Versione 1.0 del 4/12/2024

Prima versione di questo documento

1. L'operatore SKYLINE

Consideriamo un insieme $\boldsymbol{P} = \{\boldsymbol{P}_0, \boldsymbol{P}_2, \ldots \boldsymbol{P}_{N-1}\}$ di N punti in uno spazio a D dimensioni; il punto $\boldsymbol{P}_i$ ha coordinate $(P_{i,0}, P_{i,1}. \ldots, P_{i,D-1})$. Diciamo che $\boldsymbol{P}_i$ *domina* $\boldsymbol{P}_j$ se si verificano entrambe le condizioni seguenti:

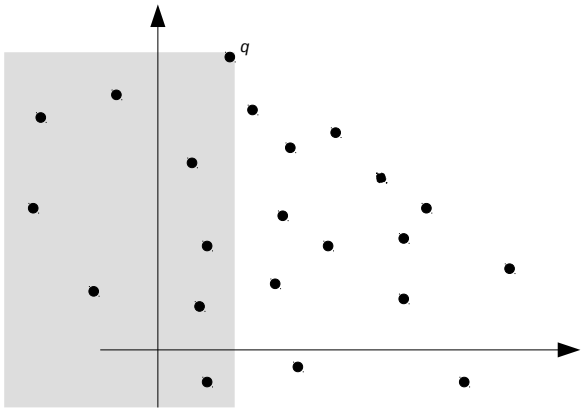
1. Per ogni dimensione $k = 0, \ldots D - 1$, $P_{i,k} \geq P_{j,k}$
2. Esiste almeno una coordinata k per cui la disuguaglianza di cui al punto precedente vale in senso stretto

(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, $\boldsymbol{P}_i$ domina $\boldsymbol{P}_j$ se e solo se ogni coordinata di $\boldsymbol{P}_i$ ha valore maggiore o uguale alla corrispondente coordinata di $\boldsymbol{P}_j$, ed esiste almeno una coordinata di $\boldsymbol{P}_i$ che abbia valore strettamente maggiore della corrispondente coordinate di $\boldsymbol{P}_j$. Ad esempio, nel caso $D = 3$ di ha che (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 $\boldsymbol{s}$ = (3, 5, 4) e $\boldsymbol{t}$ = (3, 6, 3), si ha che $\boldsymbol{s}$ non domina $\boldsymbol{t}$ e $\boldsymbol{t}$ non domina $\boldsymbol{s}$.

Lo *skyline* $Sk(\boldsymbol{P})$ dell'insieme $\boldsymbol{P}$ 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):

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

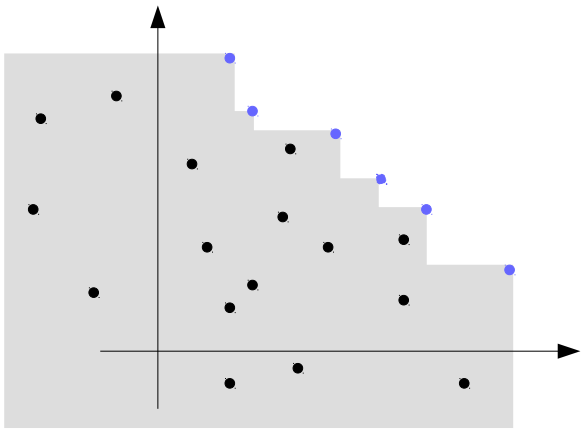
Per capire in cosa consiste l'operatore skyline, è utile considerare un semplice esempio in due dimensioni. La figura seguente rappresenta un insieme di punti sul piano:



I punti dominati da $\boldsymbol{q}$ sono quelli che si trovano all'interno dell'area grigia "proiettata" dal punto $\boldsymbol{q}$ 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 ($\boldsymbol{q}$ in questo caso).

L'operatore skyline viene utilizzato in diversi problemi di ottimizzazione ed è fornito come estensione del linguaggio SQL [1, 2]. Esistono diversi algoritmi per determinare lo skyline. In questo progetto consideriamo l'algoritmo banale che procede per eliminazione: si considerano uno

alla volta tutti i punti dell'insieme ***P***, e per ogni punto ***q*** 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 una matrice $P[0..N - 1.\ 0..D - 1]$, dove ogni riga della matrice rappresenta le D coordinate di un punto. Lo pseudocodice seguente determina e stampa lo skyline applicando il meccanismo appena descritto:

```
SKYLINE( P0, P1, … PN-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
                if S[i] then
                        for j ← 0 to N - 1 do
                                if (S[j] and Pi domina Pj) then
                                        S[j] ← false;
                                endif
                        endfor
                endif
        endfor
        for i ← 0 to N - 1 do
                if S[i] then
                        print Pi
                endif
        endfor
```

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

L'algoritmo proposto non è il più efficiente possibile per calcolare lo skyline, ma ha il vantaggio di essere comprensibile. Lo scopo di questo progetto non è di migliorare lo stato dell'arte, ma di dimostrare la capacità di parallelizzare in modo efficiente un algoritmo utile in pratica.

2. Specifica del progetto

Scopo del progetto è la realizzazione di due implementazioni parallele dell'algoritmo per il calcolo dello skyline visto nella sezione precedente. La prima deve essere realizzata usando OpenMP; la

seconda deve essere realizzata, a scelta, usando MPI oppure CUDA (uno dei due, non entrambi). I nomi degli eseguibili devono essere `omp-skyline` (per la versione OpenMP), `mpi-skyline` (per la versione MPI) e `cuda-skyline` (per la versione CUDA). Il programma viene lanciato dalla riga di comando nel modo seguente (si fa l'esempio della versione OpenMP; le altre versioni sono analoghe):

`./omp-skyline < input_file > output_file`

Il programma deve leggere l'input da standard input e scrivere il risultato su standard output. Nel caso di MPI, il programma verrà lanciato usando `mpirun`; il processo 0 (il master) dovrà leggere l'input e stampare l'output, in quanto è l'unico che ha accesso a stdin/stdour.

L'input è costituito da $N + 2$ righe di testo: la prima riga contiene il numero di dimensioni D, seguito da eventuali altri caratteri che devono essere ignorati fino alla fine della riga; la seconda contiene il numero di punti N; le successive N righe contengono ciascuna le D coordinate reali di un punto, separate da spazi o tab.

D [*eventuali altri caratteri da ignorare*]

N

$\mathbf{P}_{00}$ $\mathbf{P}_{01}$ $\mathbf{P}_{02}$ … $\mathbf{P}_{0\ D-1}$

$\mathbf{P}_{10}$ $\mathbf{P}_{11}$ $\mathbf{P}_{12}$ … $\mathbf{P}_{1\ D-1}$

…

$\mathbf{P}_{N-1\ 0}$ $\mathbf{P}_{N-1\ 1}$ $\mathbf{P}_{N-1\ 2}$ … $\mathbf{P}_{N-1\ D-1}$

Al termine dell'esecuzione, il programma deve stampare su standard output le coordinate dei punti dello skyline, nell'ordine in cui i punti compaiono nel file di input. L'output deve avere lo stesso formato dell'input: se lo skyline è composto da $K \leq N$ punti, l'output del programma conterrà $K + 2$ righe di testo; nella prima riga è presente il numero di dimensioni D; sulla seconda comparirà il valore K; nelle K righe successive sono elencate le coordinate dei punti che fanno parte dello skyline, separate da spazi. I punti dello skyline possono essere elencati in un ordine qualsiasi.

Si tenga inoltre presente quanto segue:

- Si assuma $D \geq 2$
- Si assuma $N \leq 1024 \times 1024$; il valore di N sarà comunque sempre scelto in modo tale da non esaurire la memoria a disposizione (sia nella versione per CPU che in quella per GPU).
- Verranno accettati programmi che impongono vincoli sulla dimensione del dominio (es., che funzionano correttamente solo se il numero di punti è multiplo di...); la relazione deve indicare chiaramente l'esistenza di tali vincoli. Ovviamente programmi soggetti a vincoli riceveranno una valutazione inferiore rispetto a quelli che funzionano correttamente nel caso generale.

3. Cosa consegnare

Viene richiesto di consegnare due versioni parallele del programma:

1. La prima, chiamata `omp-skyline.c`, deve utilizzare OpenMP;
2. La seconda versione deve utilizzare, a scelta, MPI oppure CUDA (uno dei due, non entrambi). Il file deve chiamarsi rispettivamente `mpi-skyline.c` oppure `cuda-skyline.cu`

Oltre ai due programmi di cui sopra, è obbligatorio includere:

3. Una relazione in formato PDF della lunghezza massima di 6 facciate che descriva le strategie di parallelismo adottate e discuta la scalabilità ed efficienza dei programmi realizzati.

Ulteriori requisiti:

- I sorgenti devono essere adeguatamente commentati. All'inizio di ogni file deve essere indicato cognome, nome e numero di matricola dell'autore/autrice.
- Includere un file di testo chiamato README contenente le istruzioni per la compilazione e l'esecuzione dei programmi consegnati; chi lo desidera può usare un makefile per la compilazione (non obbligatorio).
- Includere ogni altro file necessario alla compilazione del software (es., `hpc.h` per chi ne fa uso). È possibile scomporre il proprio codice in più sorgenti da compilare separatamente e riunire in fase di linking; in tal caso occorre documentare la procedura di compilazione nel file README di cui al punto precedente.
- I programmi verranno compilati e testati sul server `isi-raptor03.csr.unibo.it`. Tuttavia, le misure di prestazioni descritte nella relazione (vedi oltre) non devono necessariamente essere condotte sul server, ma possono essere effettuate su proprio hardware.
- Tutti i programmi devono compilare senza *warning*. Per le versioni OpenMP e MPI si usino i flag `-std=c99 -Wall -Wpedantic`, come fatto durante il corso.
- Verranno accettati programmi che impongono vincoli sulla dimensione del dominio (es., numero di punti multiplo di…), tenendo presente che comporteranno una valutazione inferiore. La relazione deve indicare chiaramente l'esistenza di tali vincoli.
- La relazione non deve superare la lunghezza di **sei facciate in formato A4**, contando tutte le pagine (inclusi eventuali frontespizi, indici o altro; suggerisco di evitare frontespizi e indici, che su un documento così corto non servono). Il formato della relazione è libero, ma viene fornito uno schema in formato LibreOffice che può essere usato come base se lo si desidera.
- La relazione non deve descrivere il codice riga per riga (per quello ci sono già i sorgenti), ma le strategie di parallelizzazione adottate, eventualmente con l'ausilio di schemi o diagrammi. La relazione **deve** riportare e commentare le prestazioni delle versioni parallele realizzate utilizzando le metriche che si ritengono più appropriate. Indicare il proprio cognome, nome e numero di matricola.

Invito a prestare la massima attenzione alla qualità della relazione, perché una relazione scadente è il motivo più frequente di penalizzazione della valutazione. Vanno evitati (e saranno fortemente penalizzati) errori grossolani come "~~ghost shell~~" invece di "ghost cell", "~~pull di thread~~" invece di "pool di thread". Va inoltre evitato l'uso dei seguenti termini:

- "tempistica/tempistiche" (→ tempi di esecuzione)
- "prestante" o "performante" ("~~algoritmo più prestante~~" / "~~algoritmo più performante~~"→ algoritmo con prestazioni migliori)
- "randomico" (→ casuale)

Anche se la relazione del progetto non è una tesi di laurea, è utile prendere visione dei consigli per la stesura della tesi, limitatamente alla parte relativa ai consigli di scrittura. Può anche essere utile fare riferimento alla guida di stile di programmazione che viene adottata nel laboratorio di Algoritmi e Strutture Dati.

4. Suggerimenti

Il tempo di esecuzione dell'algoritmo SKYLINE dipende sia dai dati di input che dal risultato (questo tipo di algoritmi si dicono *output-sensitive*). In particolare, dipende dal numero N di punti, dalla dimensione D dello spazio dei punti, e dal numero K di punti che fanno parte dello skyline (è parte del progetto determinare, se la si ritiene necessaria, l'espressione del costo asintotico dell'algoritmo).

Il programma seriale fornito va inteso come un possibile punto di partenza; è possibile modificarlo per adattarlo alle proprie esigenze, oppure partire da zero con una implementazione diversa. L'unica cosa che non può cambiare è il formato dell'input e dell'output.

Nella directory `datasets/` vengono forniti alcuni file di input che è possibile utilizzare per testare l'algoritmo. Le caratteristiche dei file di input sono riassunte nella seguente tabella:

File	Numero punti (*N*)	Dimensioni (*D*)	Punti skyline (*K*)
circle	1000	2	15
test1	100000	3	26
test2	100000	4	10352
test3	100020	10	24892
test4	100009	8	17458
test5	100000	20	96973
test6	100100	50	100050
test7	100400	200	100200
worst	100000	10	100000

Alcuni input potrebbero richiedere molto tempo per essere elaborati; non è obbligatorio usarli tutti, quindi è possibile limitarsi a quelli che possono essere gestiti in tempi ragionevoli sulla propria macchina. Nel caso in cui si vogliano generare input di dimensioni diverse, si faccia riferimento ai comandi presenti nel Makefile. La maggior parte dei file sono stati generati usando il comando `rbox` che è parte del pacchetto `qhull-bin` (già installato sul server).

Invito a prestare la massima attenzione alla stesura della relazione. Una relazione scadente è uno dei motivi più frequenti di penalizzazione della valutazione dei progetti. Vanno assolutamente evitati errori grossolani come "~~ghost shell~~" invece di "ghost cell", "~~pull di thread~~" invece di "pool di thread". Anche se la relazione non è una tesi di laurea, invito a prendere visione dei consigli per la stesura della tesi sulla mia pagina Web.

5. Modalità di svolgimento del progetto

- Il progetto deve essere frutto del lavoro individuale di chi consegna. Non è consentito condividere il codice o la relazione con altri, in tutto o in parte.
- Il programma fornito può essere modificato a piacimento.
- È ammesso l'uso di porzioni di codice trovato in rete o da altre fonti purché (*i*) la provenienza di codice scritto da terzi sia chiaramente indicata in un commento, e (*ii*) la licenza di tale codice ne consenta il riutilizzo. L'unica eccezione è il codice fornito dal docente a lezione o in laboratorio, che può essere usato liberamente senza necessità di indicarne la fonte.
- Le richieste di chiarimenti sulle specifiche (cioè su questo documento) vanno inviate sul forum che è stato creato sulla piattaforma "Virtuale"; richieste inviate via mail non riceveranno risposta. Si tenga presente che non sarà possibile rispondere a domande relative al proprio codice, né a richieste di debug: il progetto è una componente essenziale dell'esame, e deve essere svolto in totale autonomia.

6. Consegna e validità del progetto

Il progetto può essere consegnato in qualsiasi momento, prima o dopo aver sostenuto la prova scritta. Le eventuali valutazioni positive (del progetto e/o della prova scritta) restano valide fino alla sessione d'esame di settembre 2025 inclusa; dopo tale data i voti in sospeso saranno persi.

Il progetto si consegna una volta sola. Non è possibile apportare modifiche o correzioni dopo la consegna: chi vuole migliorare la valutazione dovrà consegnare un nuovo progetto su nuove specifiche.

La consegna deve avvenire tramite la piattaforma https://virtuale.unibo.it/, mediante la quale è possibile caricare un unico archivio in formato `.zip` oppure `.tar.gz` contenente sia i sorgenti che la relazione. L'archivio dovrà essere denominato con il cognome e nome dell'autore (es., `MarzollaMoreno.zip` oppure `MarzollaMoreno.tar.gz`), e dovrà contenere una directory con lo stesso nome (es., `MarzollaMoreno/`) contenente a sua volta i file secondo il seguente layout:

`MarzollaMoreno/src/omp-skyline.c`

`MarzollaMoreno/src/mpi-skyline.c` (*se si svolge la versione MPI*)

`MarzollaMoreno/src/cuda-skyline.cu` (*se si svolge la versione CUDA*)

`MarzollaMoreno/src/Makefile` (*facoltativo*)

`MarzollaMoreno/README` (*facoltativo*)

`MarzollaMoreno/Relazione.pdf`

Includere ogni altro file necessario alla compilazione del codice, come ad esempio hpc.h o simili. Non è necessario includere i file di input già forniti.

Prestare attenzione che la piattaforma Virtuale limita la dimensione dei file caricati a 20MB. **Non allegare i file di input forniti**: sono molto voluminosi e in ogni caso ne ho già una copia.

7. Valutazione dei progetti

Un progetto verrà considerato **sufficiente** se soddisfa almeno i seguenti requisiti minimi:

- I programmi compilano ed eseguono correttamente su istanze di input anche soggette a vincoli (es., dimensione del dominio multipla di...) sul server `isi-raptor03.csr.unibo.it`.
- La relazione dimostra un livello sufficiente di padronanza degli argomenti trattati ed è scritta in modo adeguato (chiarezza, correttezza grammaticale, uso appropriato della punteggiatura, uso corretto della lingua). Chi non è madrelingua italiana può scrivere la relazione in inglese se lo ritiene utile.

Ulteriori aspetti che potranno comportare una valutazione superiore:

- Qualità del codice, in termini di generalità, chiarezza, ed efficienza (es., uso appropriato delle primitive e/o dei pattern di programmazione parallela appropriati ed efficienti).
- Qualità della relazione, in termini di correttezza, chiarezza, completezza e presentazione, ovviamente tenuto conto della lunghezza massima del documento.

Ai fini di una valutazione elevata non è indispensabile che i programmi consegnati siano i più efficienti possibile; verrà piuttosto valutata la capacità di utilizzare i pattern di programmazione parallela più adeguati al problema e al paradigma di programmazione adottato, la chiarezza della relazione, nonché la correttezza delle metriche adottate per la misura delle prestazioni.

Chi desidera approfondire e applicare tecniche che non sono state viste a lezione è benvenuto/a, ma occorre osservare che questo da solo non garantisce una valutazione migliore, soprattutto se sono

presenti errori o si dimostrano carenze nelle competenze di base (es., l'analisi delle prestazioni non è corretta, il codice presenta dei bug, ecc.). Di conseguenza, prima di fare cose extra è importante assicurarsi che la parte obbligatoria del progetto sia stata svolta correttamente.

Sebbene il progetto possa essere consegnato in qualsiasi momento, effettuerò tre sessioni di valutazione al termine delle sessioni d'esame di gennaio/febbraio 2025, giugno/luglio 2025 e settembre 2025. Questo significa che alla fine di febbraio 2025, luglio 2025 e settembre 2025 valuterò tutti i progetti ricevuti fino a quel momento. Chi avesse delle scadenze, ad esempio per avere borse di studio o per laurearsi, è pregato/a di segnalarmelo alla consegna. È tuttavia obbligatorio consegnare il progetto almeno **10 giorni lavorativi prima della data entro la quale si richiede la correzione** (sottolineo **lavorativi**) per consentirmi di far fronte ad eventuali altri impegni accademici.

8. Checklist per la consegna

Prima di consegnare il progetto, assicurarsi che i requisiti fondamentali elencati nella lista seguente siano soddisfatti:

□	Vengono consegnate due versioni del programma, una che usa OpenMP, e una che usa (a scelta) MPI oppure CUDA.
□	I sorgenti compilano ed eseguono correttamente sul server `isi-raptor03.csr.unibo.it`.
□	La relazione è in formato PDF e ha lunghezza minore o uguale a 6 facciate.
□	I sorgenti e la relazione indicano chiaramente cognome, nome e numero di matricola dell'autore/dell'autrice.
□	Il progetto viene consegnato in un unico archivio .zip oppure .tar.gz, nominato con nome e cognome dell'autore, che include i sorgenti e la relazione in formato PDF.

9. Riferimenti bibliografici

[1] S. Borzsony, D. Kossmann and K. Stocker, "The Skyline operator," *Proceedings 17th International Conference on Data Engineering*, Heidelberg, Germany, 2001, pp. 421-430, doi: 10.1109/ICDE.2001.914855.

[2] Wikipedia, *Skyline Operator* https://en.wikipedia.org/wiki/Skyline_operator