Università di Bologna, Laurea in Ingegneria e Scienze Informatiche

Corso di High Performance Computing

Demo progetto d'esame – non valido per l'AA 2023-2024

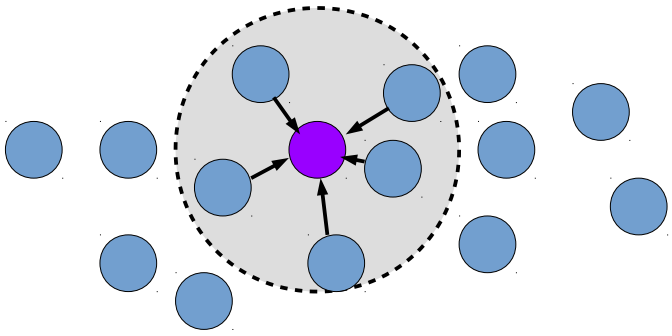
prof. Moreno Marzolla

1. Il modello SPH

SPH (*Smoothed Particle Hydrodynamics*) è un algoritmo che è stato utilizzato per descrivere il comportamento di corpi deformabili; successivamente, è stato esteso per simulare i fluidi [1]. Lo scopo di questo progetto è parallelizzare un semplice codice SPH seriale sviluppato da Lucas V. Schuermann che l'ha reso disponibile online [2].

La conoscenza delle basi teoriche del modello SPH esula dai fini di questo progetto e non è comunque richiesta per lo sviluppo del programma parallelo. Chi volesse maggiori dettagli può fare riferimento alla pubblicazione originale [1], che tuttavia è piuttosto tecnica, oppure ad una spiegazione più accessibile [3] a cura dell'autore del codice seriale.

In breve, il modello SPH usa particelle per rappresentare porzioni di spazio in cui è presente una porzione di fluido con una certa densità e pressione; si assume che tutte le particelle abbiano la stessa massa `MASS`, cioè contengano la stessa quantità di fluido. Ogni particella interagisce con quelle presenti in un intorno di raggio `H` (area tratteggiata nella figura seguente) che possono modificarne densità e pressione ed esercitare delle forze.



Conoscendo posizione (x, y) velocità corrente (vx, vy) di una particella, e la somma delle forze (fx, fy) che agiscono su di essa (fx e fy rappresentano le componenti del vettore forza lungo gli assi cartesiani), è possibile applicare la seconda legge della dinamica di Newton:

$$F = ma$$

per calcolare l'accelerazione a cui è sottoposta la particella e quindi la nuova posizione.

La parte importante dell'implementazione seriale fornita è costituita dalle funzioni seguenti:

- `compute_density_pressure()`, che aggiorna la densità e la pressione della regione di spazio associata alle porzioni di spazio rappresentate dalle particelle;
- `compute_forces()`, che utilizzando i valori aggiornati di densità e pressione, calcola le forze che agiscono tra particelle vicine, a cui si somma la forza di gravità che tende a spingere le particelle verso il basso;
- `integrate()`, che utilizzando i valori aggiornati delle forze di cui al punto precedente, unitamente alla posizione e velocità corrente di ogni particella, determina la nuova posizione e velocità.
- `avg_velocities()`, che calcola la velocità media (in modulo) delle particelle. La velocità

media dovrebbe diminuire man mano che il sistema si stabilizza, anche se non raggiungerà mai lo zero.

Le funzioni precedenti vanno eseguite nell'ordine indicato (il calcolo della velocità media viene effettuato periodicamente allo scopo di non appesantire troppo la computazione). Il programma esegue un certo numero di iterazioni, per ciascuna delle quali esegue le funzioni precedenti per calcolare la la nuova configurazione delle particelle partendo dallo stato corrente del sistema. Il numero di particelle e il numero di iterazioni sono parametri di input (vedi sezione 2).

Il programma seriale può essere compilato in due modalità:

- modalità testuale (default), in cui è possibile passare sulla riga di comando il numero di particelle N da generare e il numero di passi S da simulare; il programma stampa una riga di output ogni 10 passi, che indica la velocità media delle particelle. La velocità media dovrebbe diminuire man mano che si procede e le particelle si stabilizzano. La versione testuale si compila con il comando `make`
- modalità grafica, in cui viene mostrata graficamente l'evoluzione del sistema. Si può cliccare col mouse per generare una "goccia" d'acqua, e premere il tasto "r" per resettare il sistema nella configurazione iniziale. Per terminare occorre chiudere la finestra. La versione grafica si compila con il comando `make sph.gui`

La versione grafica è stata testata in ambiente Linux/Ubuntu con le librerie OpenGL, e ha scopo puramente dimostrativo. **Ai fini del progetto la versione grafica può essere ignorata: solo la versione testuale deve essere parallelizzata, e sarà compilata e valutata**.

Come si può verificare esaminando il codice sorgente, il programma seriale richiede tempo $\Theta(N^2)$ per ogni iterazione, in quanto vengono valutate tutte le N^2 coppie di particelle per vedere quali di essere interagiscono. L'uso di strutture dati più complesse potrebbe migliorare le prestazioni del codice seriale, ma ciò esula dallo scopo del progetto, anche perché potrebbe compromettere la possibilità di parallelizzare il programma in modo semplice.

2. Specifica del progetto

È richiesta la realizzazione di **due versioni parallele** del programma seriale fornito (nella versione che NON fa uso dell'interfaccia grafica): la prima deve sfruttare OpenMP, mentre la seconda versione deve usare, a scelta, MPI oppure CUDA (uno dei due, non entrambi). I nomi degli eseguibili devono essere `omp-sph` (per la versione OpenMP), `mpi-sph` (per la versione MPI) e `cuda-sph` (per la versione CUDA). Il programma verrà lanciato dalla riga di comando nel modo seguente (si fa l'esempio con la versione OpenMP; le altre sono analoghe):

```
./omp-sph [N [S]]
```

dove N è il numero di particelle, e S il numero di passi da simulare (intero maggiore o uguale a zero); nel caso di MPI, il programma verrà lanciato usando `mpirun`.

Si tenga presente quanto segue:

- Si assuma $N \leq$ `MAX_PARTICLES`
- Verranno accettati programmi che impongono vincoli sulla dimensione del dominio (es., che funzionano correttamente solo se N è multipla di…). La relazione deve indicare chiaramente la presenza di tali vincoli, e in questi casi il progetto riceverà una valutazione inferiore.

Il programma seriale fornito va inteso come un punto di partenza; è possibile modificarlo a piacimento per adattarlo alle proprie esigenze.

Si tenga presente che il programma, anche nelle versioni parallele, potrebbe risultare molto lento con un numero elevato di particelle. Questo non sarà considerato un problema: si effettuino i test con dimensioni massime dell'input che consentano la terminazione in tempi ragionevoli sulla mac-

china usta per i test.

3. Cosa consegnare

Viene richiesto di consegnare due versioni parallele del programma:

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

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

3. Una relazione in formato PDF della **lunghezza massima di sei facciate** che descriva le strategie di parallelismo adottate e discuta le prestazioni dei programmi realizzato usando le metriche più appropriate viste a lezione.

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).
- È necessario includere ogni altro file necessario alla corretta compilazione del software (es., il file `hpc.h` per chi ne fa uso). È possibile scomporre i sorgenti in più file da riunire in fase di linking, come documentato 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 visto durante il corso.
- La relazione non deve superare la lunghezza di **sei facciate in formato A4**, contando tutte le pagine (inclusi eventuali frontespizi, indici o altro). 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. Inoltre, deve illustrare e commentare speedup ed efficienza almeno della versione OpenMP. Nella relazione va indicato il proprio cognome, nome e numero di matricola.

Invito a prestare la massima attenzione alla qualità della relazione, perché una relazione scadente èil 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". È inoltre vietato l'uso dei seguenti termini:

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

L'orrido neologismo "randomico" è particolarmente fastidioso.

Anche se la relazione di questo progetto non è una tesi di laurea, è utile prendere visione dei consigli per la stesura della tesi sul mio sito Web, limitatamente alla parte sulla scrittura del testo. È utile fare riferimento alla guida di programmazione adottata nel laboratorio di algoritmi e strutture dati.

4. 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.
- È ammesso l'uso di porzioni di codice reperito in rete o tramite altre fonti purché (*i*) la provenienza di ogni frammento 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 costituita dal codice fornito dal docente a lezione o in laboratorio, che può essere usato liberamente senza necessità di indicarne la fonte.
- Sono disponibile a fornire chiarimenti sulle specifiche del progetto (cioè sul presente documento), ma **non guarderò il codice per nessun motivo se non dopo la consegna**. Lo svolgimento del progetto in totale autonomia è una parte importante dell'esame e va affrontato con la massima serietà.
- Apprezzo sempre gli approfondimenti di argomenti non trattati a lezione, purché siano tecnicamente corretti e bene argomentati. Occorre quindi essere pienamente consapevoli di quello che si sta facendo. È però importante sottolineare che **lo sviluppo di estensioni e approfondimenti non garantisce una valutazione migliore**, soprattutto nel caso in cui siano presenti errori, o si dimostrino carenze nelle competenze di base fornite dal corso (es., l'analisi delle prestazioni non sia corretta, il codice presenti dei bug, …).

5. Consegna e validità del progetto

Il progetto può essere consegnato in qualsiasi momento, prima o dopo aver sostenuto la prova scritta. Le valutazioni positive (del progetto e della prova scritta) restano valide **fino al 30 settembre 2023**; dopo tale data inizierà la nuova edizione del corso e tutti i voti in sospeso saranno annullati.

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/, caricando un unico archivio in formato .zip oppure .tar.gz contenente sia i sorgenti che la relazione; si tenga presente che la piattaforma Virtuale limita la dimensione dei file caricati a 20MB. L'archivio dovrà essere denominato con il cognome e nome dell'autore/autrice (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-sph.c`

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

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

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

`MarzollaMoreno/README`

`MarzollaMoreno/Relazione.pdf`

Includere ogni altro file necessario alla compilazione del codice, come ad esempio l'header `hpc.h` o altri sorgenti, nel caso si ricorra a compilazioni separate.

6. 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 italiana). 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 correttezza (in particolare, assenza di *race condition*), chiarezza, ed efficienza (es., uso appropriato delle primitive e/o dei pattern di programmazione parallela appropriati ed efficienti).
- Generalità (es., il programma funziona con qualunque dimensione del dominio).
- Qualità della relazione, in termini di correttezza, chiarezza, e presentazione. Per ottenere una buona valutazione è richiesto che la relazione contenga lo studio di speedup ed efficienza dei programmi realizzati (oppure delle metriche che si ritengono più adeguate nel caso della versione CUDA). Non è obbligatorio misurare i tempi di esecuzione sul server, ma è possibile usare hardware proprio. Verrà valutata la capacità di interpretare le prestazioni misurate, piuttosto che le prestazioni stesse del programma.

Sebbene il progetto possa essere consegnato in qualsiasi momento, effettuerò tre sessioni di valutazione al termine delle sessioni d'esame di gennaio/febbraio 2023, giugno/luglio 2023 e settembre 2023. Questo significa che alla fine di febbraio 2023, luglio 2023 e settembre 2023 valuterò tutti i progetti ricevuti fino a quel momento. Chi avesse delle scadenze, ad esempio legate a richieste di borse di studio o per potersi laureare, è pregato/a di segnalarmelo all'atto della consegna. L'unico vincolo è che il progetto venga consegnato 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.

7. 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/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.

8. Riferimenti bibliografici

[1] Matthias Müller, David Charypar, and Markus Gross. 2003. *Particle-based fluid simulation for interactive applications*. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03). Eurographics Association, Goslar, DEU, 154–159. https://dl.acm.org/doi/10.5555/846276.846298

[2] https://github.com/cerrno/mueller-sph

[3] Lucas V. Schuermann, *Particle-based Fluid Simulation with SPH*, online all'indirizzo: https://lucasschuermann.com/writing/particle-based-fluid-simulation