Università di Bologna, Laurea in Ingegneria e Scienze Informatiche

Corso di High Performance Computing 2021/2022

Progetto d'esame

prof. Moreno Marzolla

Versione 1.0 del 29/11/2021

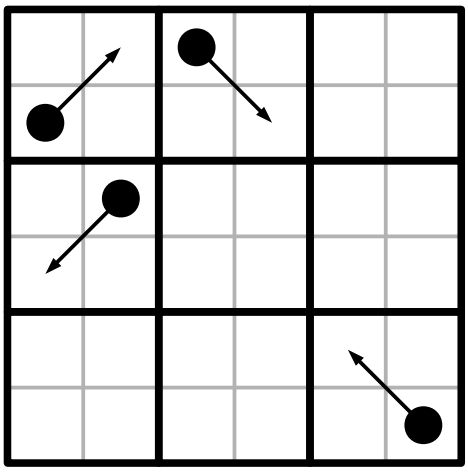
Prima versione

1. Gli automi cellulari *Lattice Gas*

Gli automi cellulari della famiglia Lattice Gas (*Lattice-Gas Celluar Automata*, LGCA) sono un semplice modello di propagazione di un fluido su una griglia regolare, e sono stati derivati dall'equazione di Navier-Stokes, uno dei pilastri della fluidodinamica. I primi modelli LGCA sono stati proposti da Hardy, Pomeau e de Pazzis [1] (HPP), e da Frisch, Hasslacher e Pomeau [2] (FHP). Questi modelli sono molto rudimentali e presentano diversi problemi dal punto di vista fisico, per cui sono stati soppiantati da un'altra classe di automi cellulari detti *metodi Lattice Boltzmann*. Scopo di questo progetto è l'implementazione parallela di un modello HPP; il codice seriale fornito è basato su una implementazione NetLogo [3], ed è stato esteso per supportare particelle fisse. Il modello HPP viene descritto anche in [4], usando il linguaggio FORTH su una architettura hardware – ormai obsoleta – sviluppata dagli autori per la simulazione di automi cellulari in modo efficiente.

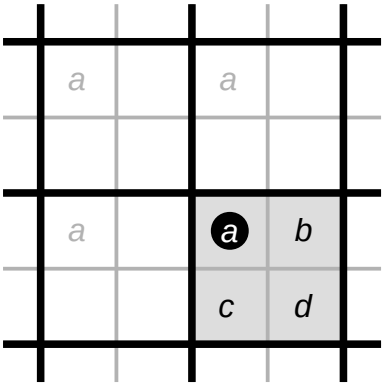
Nel modello HPP il dominio è costituito da una griglia quadrata di lato N, che deve essere pari. Il dominio ha condizioni al contorno cicliche, quindi le celle su un bordo si considerano adiacenti a quelle del bordo opposto. Ogni cella della griglia può contenere il vuoto (EMPTY), oppure una molecola di gas (GAS), oppure un ostacolo (WALL). Tutte le molecole di gas hanno la stessa massa e velocità assoluta: ogni molecola si sposta al più in una cella adiacente ad ogni passo, secondo le regole definite in seguito.

Il dominio è partizionato in blocchi di dimensione 2×2; questo significa che la lunghezza N del lato del dominio deve essere pari. Il valore GAS in una delle celle indica la presenza di una molecola di gas diretta verso la cella opposta in diagonale, come nell'esempio seguente in cui mostriamo un dominio di dimensione $N = 6$:

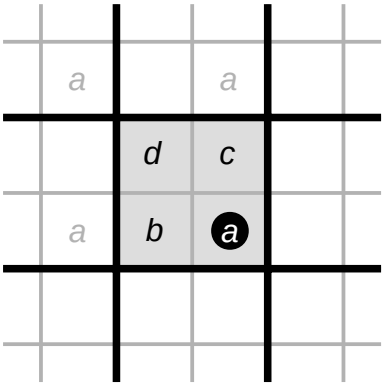


Una particolarità del modello HPP, che lo distingue dagli esempi visti a lezione come il Game of Life, è la definizione del "vicinato" di una cella. Per simulare il movimento delle molecole si usa il vicinato di Margolus (dal suo inventore Norman Margolus). Chiamiamo a una delle celle di

coordinate (i, j) entrambe pari. L'aggiornamento avviene alternando due fasi, dette "fase pari" e "fase dispari", in ciascuna delle quali il vicinato di a cambia. In particolare, detti b, c, d i vicini di a, la figura seguente mostra le posizioni di a, b, c, d durante le fasi pari e dispari:



Vicinato di *a* durante le fasi "pari"

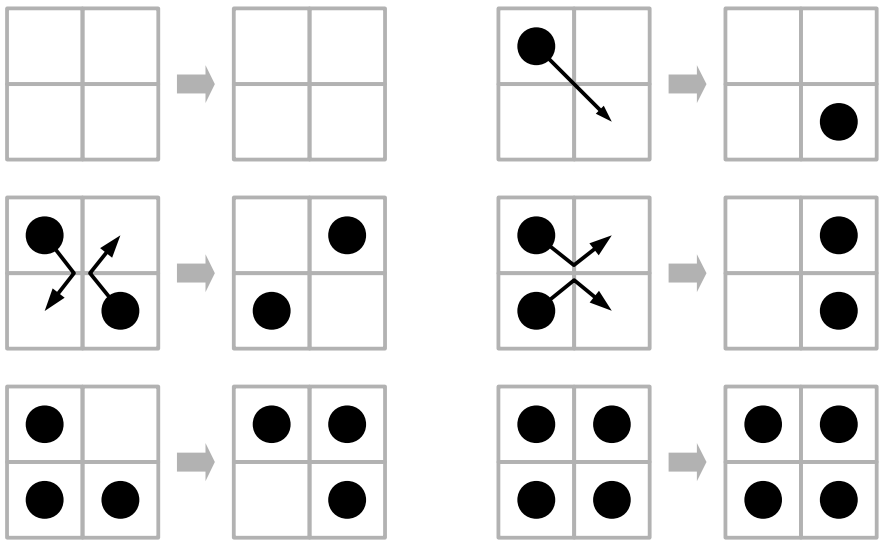


Vicinato di *a* durante le fasi "dispari"

L'aggiornamento avviene applicando le stesse regole prima al vicinato "pari" di tutte le celle a e poi al vicinato "dispari" delle stesse celle. Come in tutti gli automi cellulari visti a lezione, anche qui occorre leggere lo stato corrente da un dominio, e scrivere lo stato aggiornato in un *nuovo* dominio, che al termine dell'operazione diventerà il domino corrente.

Le regole per l'aggiornamento del dominio tengono conto degli urti delle molecole tra di loro e con le particelle fisse; inoltre, per rendere il modello consistente dal punto di vista fisico, le regole conservano il momento totale del sistema (il momento è il prodotto delle masse per i vettori velocità) e sono invarianti per rotazione e riflessione.

Nel caso di urti molecola-molecola è sufficiente definire i sei casi base seguenti [4] (i rimanenti si ottengono per rotazione e riflessione):



La presenza di particelle fisse introduce nuovi casi che possono essere ricondotti a quelli sopra, con l'eccezione che le particelle fisse coinvolte negli urti non si spostano; il codice fornito tiene già conto delle particelle fisse.

Le regole di cui sopra sono *reversibili*: se ad un certo punto si inverte l'ordine di applicazione delle fasi (prima dispari e poi pari), il tempo scorre all'indietro e alla fine si riotterrà la configurazione di partenza. Il programma seriale contiene una parte normalmente disabilitata che mostra questo comportamento. Se la si abilita, al termine della simulazione viene invertita la direzione del tempo fino a riottenere una configurazione che deve essere identica alla prima, altrimenti il programma è

sicuramente errato.

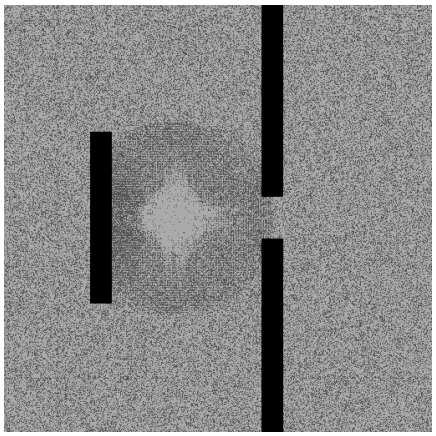


Figura 1: Esempio di output

La Figura 1 mostra l'immagine prodotta dall'esecuzione del comando `./hpp 512 32 walls.in` in cui si può osservare l'onda di pressione causata da una sfera di gas compresso in espansione.

2. Specifica del progetto

È richiesta la realizzazione di **due versioni parallele** dell'algoritmo per la simulazione dell'automa HPP: 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-hpp` (per la versione OpenMP), `mpi-hpp` (per la versione MPI) e `cuda-hpp` (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-hpp [N [S]] input_file`

dove N è la lunghezza del lato del dominio (deve essere pari), e S il numero di passi da simulare (intero maggiore o uguale a zero). Il programma produce un file in formato PGM (*Portable Graymap*) su un file il cui nome dipende da S; il file rappresenta la configurazione del sistema dopo S passi. Nel caso di MPI, il programma verrà lanciato usando `mpirun`; il file deve essere creato dal processo 0 (il master); si ricorda che tutti i processi MPI hanno accesso alla riga di comando.

Si tenga presente quanto segue:

- Si assuma N pari, $N \geq 128$; il programma deve funzionare correttamente con un valore di N arbitrariamente grande, purché ci sia abbastanza memoria (nel sistema e/o nella GPU) per contenere tutte le strutture dati necessarie.
- Verranno accettati programmi che impongono vincoli sulla dimensione del dominio (es., che funzionano correttamente solo se la dimensione N è multipla di...); la relazione deve indicare chiaramente la presenza di tali vincoli. In questi casi i progetti riceveranno una valutazione inferiore rispetto a quelli che funzionano correttamente nel caso generale.

Il programma seriale fornito con queste specifiche 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.

3. Cosa consegnare

Viene richiesto di consegnare due versioni parallele del programma:

1. La prima, chiamata `omp-hpp.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-hpp` oppure `cuda-hpp`

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 in termini delle metriche 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.
- I programmi verranno compilati e testati sul server `isi-raptor03.csr.unibo.it`.
- 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, e illustrare la scalabilità dei programmi mostrando i grafici di speedup ed efficienza. Nella relazione va indicato il proprio cognome, nome e numero di matricola.

Invito a prestare la massima attenzione alla stesura della relazione. Una relazione scadente è uno dei motivi più frequenti 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" ("~~algoritmo più prestante~~" → algoritmo con prestazioni migliori)
- "randomico" (→ casuale)

L'orrido neologismo "randomico" è particolarmente fastidioso.

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.

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. Bisogna quindi essere pienamente consapevoli di quello che si sta facendo: scopiazzare pezzi di codice da StackOverflow (prassi che sta diventando tristemente comune) senza aver capito cosa fanno è quasi sempre una pessima idea.

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/o della prova scritta) resteranno valide **fino al 30 settembre 2022**; 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-hpp.c`

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

`MarzollaMoreno/src/cuda-hpp.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 file `hpc.h` o altri file 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).

Ulteriori aspetti che potranno comportare una valutazione superiore:

- Qualità del codice, in termini di correttezza, chiarezza, ed efficienza (es., uso appropriato delle primitive e/o dei pattern di programmazione concorrente adeguati).
- 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. Non è obbligatorio misurare i tempi di esecuzione sul server. Verrà valutata la capacità di interpretare i risultati ottenuti piuttosto che i risultati stessi.

Sebbene il progetto possa essere consegnato in qualsiasi momento, effettuerò tre sessioni di valutazione al termine delle sessioni d'esame di gennaio/febbraio 2022, giugno/luglio 2022 e settembre 2022. Questo significa che alla fine di febbraio 2022, luglio 2022 e settembre 2022 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 anche ad eventuali altri miei 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] Hardy, Y. Pomeau and O. de Pazzis, *Time evolution of two-dimensional model system. I. Invariant states and time correlation functions*, J. Math. Phys. 14 (1973), pp. 1746-1759, 10.1063/1.1666248

[2] U. Frisch, B. Hasslacher and Y. Pomeau, *Lattice-gas automata for the Navier-Stokes equation*, Phys. Rev. Lett. 56 (1986), pp. 1505-1508. 10.1103/PhysRevLett.56.1505

[3] Wilensky, U. *NetLogo Lattice Gas Automaton model*. ccl.northwestern.edu/netlogo/models/LatticeGasAutomaton. Center for Connected Learning and Computer-Based Modeling, Northwestern University, 2002, Evanston, IL.

[4] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modeling*, MIT Press, 1987, ISBN 9780262200608. Disponibile online all'indirizzo people.csail.mit.edu/nhm/cam-book.pdf