Progetto di High Performance Computing 2018-2019

Moreno Marzolla moreno.marzolla@unibo.it

Revisioni

- [2018-12-19] Prima versione
- [2018-12-29] Corretti due accessi *out of bound* nella funzione `propagate_energy()`
- [2019-01-04] Rimpiazzato il file `hpc.h` nell'archivio con la versione più recente usata a lezione
- [2019-03-09] È disponibile una Versione PDF di questo documento.

Download

- ProgettoHPC1819.zip (ultimo aggiornamento 2019-01-04)

Descrizione del progetto

Lo scopo del progetto è di implementare versioni parallele di un semplice modello matematico di propagazione dei terremoti. Il modello che consideriamo è una estensione in due dimensioni dell'automa cellulare Burridge-Knopoff (BK). Nel modello BK, una porzione di crosta terrestre è rappresentata da una griglia quadrata di dimensioni $n \times n$. Ad ogni cella viene associato un valore reale che rappresenta la quantità di energia immagazzinata nella corrispondente porzione di terreno sotto forma di energia potenziale da tensione causata dallo scivolamento dalle placche tettoniche. Quando l'energia di una cella supera una certa soglia critica, parte dell'energia viene trasferita alle celle adiacenti producendo un terremoto.

Più in dettaglio, il modello è rappresentato da una matrice quadrata F di dimensione $n \times n$ di valori reali (usiamo il tipo `float`). Ogni cella contiene il valore dell'energia potenziale della corrispondente porzione di crosta terrestre. La matrice viene inizializzata con valori casuali, e il suo contenuto viene aggiornato ad istanti discreti di tempo $t = 0, 1, \ldots$ secondo le regole seguenti:

1. (**Incremento**) Si incrementa il valore di ogni cella della matrice di una costante `EDELTA`;
2. (**Propagazione**) Se l'energia $F_{i,j}$ di una cella (i, j) supera un valore critico `EMAX`, l'energia $F_{i,j}$ viene diminuita di `EMAX`, e si incrementa l'energia di ciascuna delle quattro celle adiacenti (nord, sud, est, ovest, se esistenti) di `EMAX/4`. **Il dominio NON si assume ciclico**, pertanto alcuni vicini potrebbero non esistere e quindi non riceveranno energia addizionale. L'energia trasferita da una cella a ciascuno dei vicini è sempre `EMAX/4`, indipendentemente dal numero di vicini.

Per funzionare correttamente, è necessario applicare prima la regola 1 a tutte le celle, e successivamente la regola 2.

È conveniente riformulare il passo 2 come segue:

2. (**Propagazione**) La cella (i, j) riceve energia `EMAX/4` da ciascun vicino la cui energia risulti maggiore di `EMAX`. Fatto questo, se $F_{i,j}$ è maggiore di `EMAX`, allora si sottrae `EMAX` da $F_{i,j}$, altrimenti si lascia invariata l'energia.

Questa nuova formulazione consente di esprimere la regola di aggiornamento nel modo canonico delle computazioni di tipo *stencil*, in cui il nuovo stato di una cella dipende dal proprio stato corrente e da quello dei vicini (e non viceversa).

Il modello BK è un automa cellulare *sincrono*: questo significa che si assume che tutte le celle vengano aggiornate contemporaneamente. Per tale ragione, nel passo 2 è necessario utilizzare due matrici, `cur` e `next`, contenenti rispettivamente i valori delle energie al passo corrente e al passo successivo. Il

passo 1 non necessita di ciò.

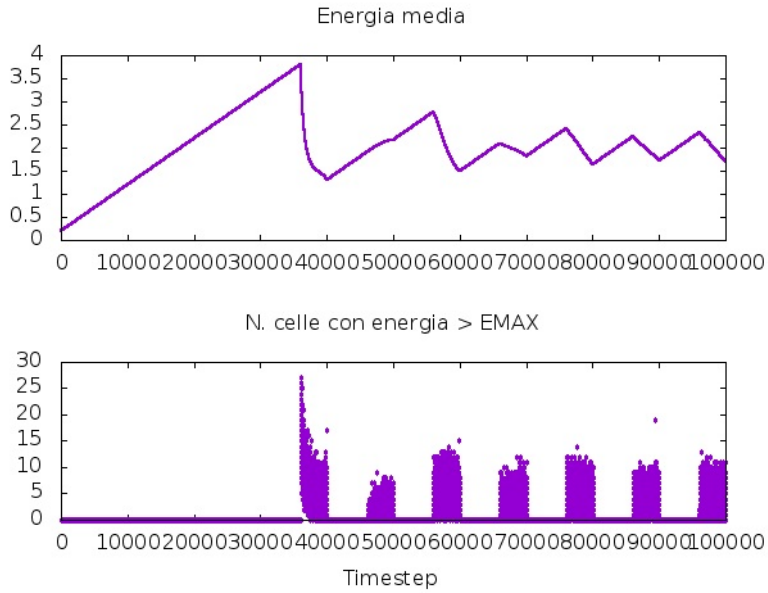

Figura 1: Energia media e numero di celle con energia maggiore di `EMAX` in un modello BK, $n = 256$, 10^5 timestep

Nonostante la sua semplicità, il modello BK esibisce un comportamento interessante. La Figura 1 mostra l'evoluzione dell'automa su un dominio di lato $n = 256$ durante 10^5 timestep. Il grafico in alto mostra l'energia media delle celle, calcolata subito dopo l'esecuzione del passo 2; il grafico in basso mostra il numero di celle con energia strettamente maggiore di `EMAX`, calcolato al termine del passo 1. Il profilo a dente di sega mostra come l'energia media aumenti fino a quando si verifica il primo terremoto, che rilascia energia abbassando il valore medio. Dopo il primo terremoto ne seguono altri di intensità inferiore.

È importante sottolineare che il modello proposto è una versione *molto* semplificata di quelli realmente usati per studiare la dinamica dei terremoti, come ad esempio i modelli proposti da Castellaro e Murgia e Akishin e altri.

Specifiche del progetto

È richiesta la realizzazione di **due** implementazioni parallele del modello BK:

- la prima deve essere realizzata usando OpenMP;
- la seconda deve essere realizzata usando uno a scelta tra MPI oppure CUDA (non entrambi).

Come punto di partenza viene fornita una implementazione seriale `earthquake.c` che può essere eseguita da riga di comando come:

```
./earthquake [nsteps [n]]
```

dove `nsteps` è il numero di passi da simulare e `n` la dimensione della griglia. Si può assumere che `nsteps` ≤ 1000000 e $n \leq 2048$ (in ogni caso la dimensione n del dominio sarà tale da non eccedere la memoria a disposizione, anche considerando tutte le altre strutture dati richieste dalla propria implementazione). Il programma utilizza valori predefiniti per le costanti `EMAX` e `EDELTA`, che suggerisco di non modificare.

L'output è costituito da `nsteps` righe di testo su standard output (una per ogni timestep). Ciascuna riga contiene due valori numerici separati da uno spazio:

- un numero intero che indica il numero di celle che, al termine del passo 1, hanno energia strettamente maggiore di `EMAX`;
- un numero reale che indica l'energia media $\bar{F}$ al termine del passo 2. L'energia media $\bar{F}$ si calcola come:

$$\bar{F} = \frac{1}{n^2} \sum_{i,j} F_{i,j}$$

Le versioni OpenMP, MPI e CUDA devono produrre risultati simili, tenuto conto che lievi differenze possono essere legate all'uso dell'aritmetica in virgola mobile (ma differenze maggiori denotano quasi sicuramente un errore). Si faccia attenzione che **un output corretto (o quasi) potrebbe essere prodotto anche da un programma errato**: per questa ragione controllerò **sempre** con la massima attenzione il codice consegnato.

La versione seriale fornita deve essere considerata come un ausilio per capire i dettagli del modello BK. *L'implementazione fornita non è (volutamente) la più efficiente*, e può essere modificata per usarla come base per le versioni parallele. L'unica cosa che non deve cambiare è l'ordine con cui le funzioni vengono invocate nel ciclo principale del programma. In particolare, ad ogni passo occorre:

a. Incrementare l'energia delle celle;
b. Calcolare il numero di celle con energia maggiore di `EMAX`;
c. Propagare l'energia in eccesso ai vicini;
d. Calcolare l'energia media delle celle.

Per comodità, viene fornito uno script di gnuplot per generare una figura simile alla figura 1. Lo script legge i dati prodotti dal programma `earthquake` che si assumono contenuti in un file chiamato `out`, e produce come risultato una immagine `earthquake.png`. L'immagine della figura 1 è stata generata con i comandi

```
./earthquake 100000 256 > out
gnuplot plot.gp
```

(al posto del comando `earthquake` deve essere possibile usare `omp-earthquake`, `mpi-earthquake` oppure `cuda-earthquake` in modo interscambiabile, ovviamente tenendo conto che la versione MPI deve essere mandata in esecuzione mediante il comando `mpirun`). L'uso di questo script è opzionale; è possibile visualizzare l'output usando qualsiasi software a propria scelta.

Cosa consegnare

Come detto in precedenza, occorre consegnare due versioni parallele del programma:

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

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

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

Ulteriori requisiti:

- Includere nome, cognome e numero di matricola in un commento all'inizio di tutti i file sorgenti consegnati e nella relazione.
- Il codice sorgente deve essere comprensibile e adeguatamente commentato.
- Includere un file README contenente le istruzioni per la compilazione e l'esecuzione dei programmi consegnati; chi lo desidera può usare un `Makefile` per la compilazione (scelta

consigliata; ci si può basare su quello fornito).
- Consegnare tutti i file necessari per la compilazione, incluso ad esempio il file `hpc.h` usato a lezione (per chi decide di usarlo).
- I programmi consegnati verranno compilati sul server `isi-raptor03.csr.unibo.it`. I programmi che non compilano o non eseguono correttamente su tale macchina non verranno presi in considerazione.
- Tutti i programmi devono compilare senza warning. Per le versioni OpenMP e MPI suggerisco di usare i flag `-std=c99 -Wall -Wpedantic`.
- La relazione non deve superare la lunghezza di **sei facciate** in formato A4, contando tutte le pagine (inclusi eventuali frontespizi). 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 spiegare il codice riga per riga (per quello ci sono già i sorgenti), ma deve illustrare le strategie di parallelizzazione adottate, e mostrare la scalabilità dei programmi mostrando i grafici di *speedup* ed *efficienza*.

Modalità di svolgimento del progetto

- *Il progetto deve essere svolto individualmente*. Non è consentito condividere in tutto o in parte il codice o la relazione con altri studenti.
- *Il progetto deve essere frutto di lavoro individuale*. 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. Come unica eccezione, il codice reso disponibile dal docente durante le lezioni o i laboratori può essere usato liberamente senza necessità di indicarne la fonte.
- Sono disponibile a fornire chiarimenti sulle specifiche del progetto (cioè sul presente testo), ma **non guarderò il vostro codice per nessun motivo se non dopo la consegna**. Lo svolgimento in autonomia del progetto è un requisito dell'esame e va preso con la massima serietà.

Consegna del progetto

Il progetto può essere consegnato in qualsiasi momento **entro le ore 23:59 il 30 settembre 2019**. La consegna deve avvenire inviando una mail dal proprio indirizzo istituzionale `@studio.unibo.it` a moreno.marzolla@unibo.it con oggetto:

> [HPC] Consegna Progetto 2018/2019

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 i sorgenti e la relazione. L'archivio sarà 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/`) con il seguente layout:

`MarzollaMoreno/src/omp-earthquake.c`

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

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

`MarzollaMoreno/src/...` (ogni altro file necessario alla compilazione e/o esecuzione del codice)

`MarzollaMoreno/src/Makefile` (facoltativo)

`MarzollaMoreno/README`

`MarzollaMoreno/Relazione.pdf`

(altri file se necessario)

Riceverete una mail di conferma dell'avvenuta ricezione del progetto; l'invio di tale mail potrebbe richiedere alcuni giorni, per cui si prega di pazientare.

È possibile consegnare il progetto prima o dopo aver sostenuto la prova scritta. Le eventuali valutazioni positive (del progetto e/o della prova scritta) **restano valide fino al 30 settembre 2019**; dopo tale data inizierà la nuova edizione del corso e **tutti i voti in sospeso verranno 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 fornite dal docente.

Valutazione del progetto

Sebbene il progetto possa essere consegnato in qualsiasi momento, effettuerò tre sessioni di valutazione al termine delle sessioni d'esami di gennaio/febbraio 2019, giugno/luglio 2019 e settembre 2019. Questo significa che alla fine di febbraio 2019, luglio 2019 e settembre 2019 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 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 agli altri impegni accademici.

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

- Il progetto compila ed esegue 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.

Ulteriori aspetti che potranno comportare una valutazione superiore:

- Qualità della relazione, in termini di correttezza e chiarezza; in particolare, verrà valutata positivamente la presenza nella relazione dei grafici di speedup ed efficienza con discussione dei risultati ottenuti. Non è obbligatorio misurare i tempi di esecuzione sul server usato per le esercitazioni.
- Qualità del codice, in termini di efficienza, uso appropriato delle primitive e/o dei pattern di programmazione concorrente adeguati (es., uso delle primitive di comunicazione collettiva MPI quando opportuno, uso della shared memory CUDA se necessaria, ecc.).
- Generalità della soluzione (es., la soluzione proposta funziona per qualunque dimensione del dominio, oppure con qualunque numero di processi MPI, ecc.).

Il voto del progetto sarà espresso in trentesimi (massimo 30); tuttavia, si terrà conto di progetti di qualità particolarmente elevata per arrotondare in modo favorevole il voto finale, o per assegnare la lode.

Suggerimenti

Versione OpenMP

Una versione OpenMP funzionante dovrebbe essere facilmente ricavabile a partire dalla versione seriale; qualche ottimizzazione è però possibile e auspicabile. Ad esempio, sebbene il dominio non debba essere considerato ciclico, può essere utile usare una ghost area composta da celle di energia costante pari a zero (perché?). Tale ghost area non necessita di essere aggiornata all'inizio di ogni iterazione. Anche le versioni MPI e CUDA possono trarre vantaggio dalla presenza di una ghost area contenente il valore zero.

Versione MPI

La versione MPI è laboriosa ma non dovrebbe presentare problemi particolari. Una parte critica è il partizionamento del dominio. Suggerisco di adottare un partizionamento in blocchi per righe, in modo da semplificare la distribuzione del dominio usando le funzioni `MPI_Scatter()` oppure `MPI_Scatterv()`.

Poiché lo stato di una cella dipende da quello delle quattro celle adiacenti (se esistono), i sottodomini gestiti dai vari processi dovrebbero essere estesi con una ghost area come discusso a lezione (si faccia riferimento ai lucidi sui pattern per la programmazione parallela). Il problema è la versione in due dimensioni dell'implementazione MPI della "regola 30" visto nella terza esercitazione MPI (si faccia riferimento alla figura 2).

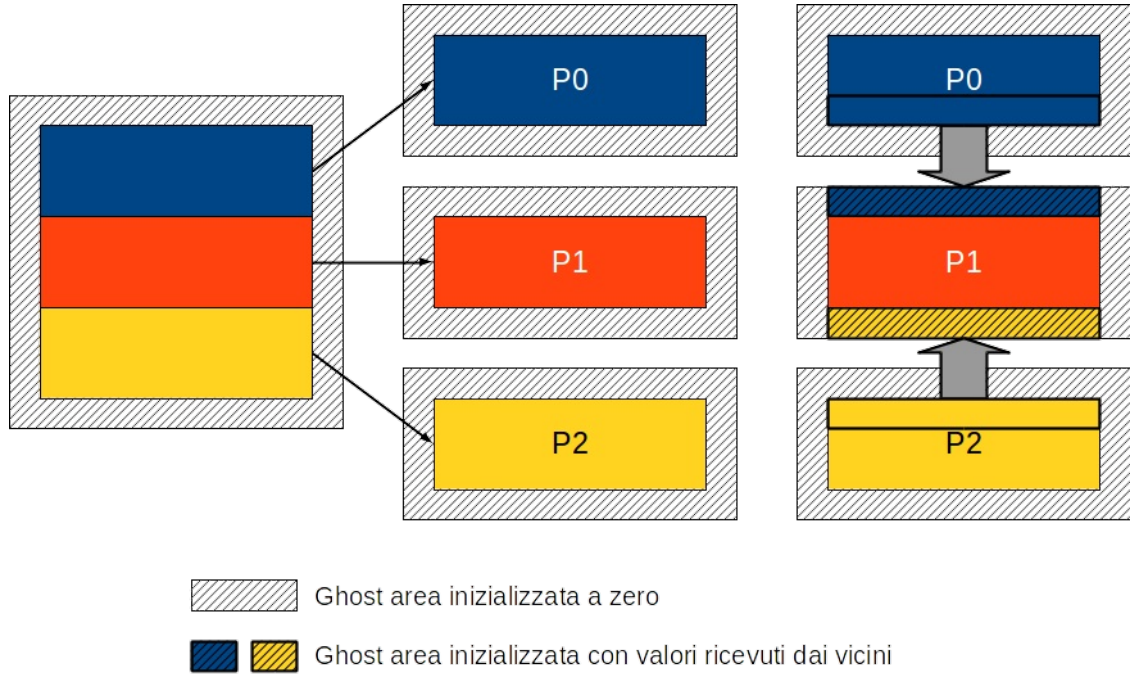

Figura 2: Pattern di comunicazione per la versione MPI

Versione CUDA

La versione CUDA è per certi versi meno laboriosa di quella MPI, ma presenta un paio di criticità a cui prestare attenzione. La prima riguarda il partizionamento del dominio, che essendo bidimensionale suggerisce l'uso di blocchi 2D organizzati in una griglia a sua volta 2D. Il problema è molto simile all'esercizio relativo all'automa ANNEAL visto nella terza esercitazione CUDA.

La seconda criticità riguarda il calcolo dell'energia media delle celle e del numero di celle con energia maggiore a `EMAX`: questo richiede una operazione di riduzione sul contenuto del dominio bidimensionale. Ma dato che in C una matrice viene rappresentata in memoria come un array, il problema può essere ridotto al caso in una dimensione (da risolvere mediante thread block 1D)…

Checklist per la consegna

Assicurarsi che i seguenti requisiti minimali siano soddisfatti prima di consegnare il progetto:

- 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 utilizzato per le esercitazioni (`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 in formato `.zip` oppure `.tar.gz`, nominato con nome e cognome dell'autore (es., `MorenoMarzolla.zip` oppure `MorenoMarzolla.tar.gz`), che include i sorgenti e la relazione in formato PDF.
- La mail di consegna ha oggetto "[HPC] Consegna progetto 2018/2019"; la mail viene spedita dal proprio indirizzo di posta istituzionale, e include cognome, nome e numero di matricola dell'autore/autrice del progetto.