Relazione di progetto
Corso di High Performance Computing

Fabrizio Margotta (fabrizio.margotta@studio.unibo.it)
Matricola 789072

Alma Mater Studiorum-Università di Bologna

2019–08–24

Abstract

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).[1]

Contents

1 Analisi

L'analisi sul modello da implementare è stata particolarmente semplificata dalle indicazioni fornite dal docente.

Vengono tuttavia di seguito presentate alcune osservazioni sul modello dei dati, ad esempio si fa notare che non è presente alcuna particolare dipendenza sui dati da parte delle iterazioni dei cicli.

Ciò permette quindi di procedere con la parallelizzazione della versione seriale fornita senza particolari intoppi.

2 Sviluppo

Entrambe le soluzioni proposte si avvalgono dell'utilizzo di una ghost area inizializzata a zero (e mai più modificata). Ciò permette di snellire notevolmente l'impatto sulle prestazioni dell'operazione di propagazione, poiché, essendoci una regione di intorno (*halo*) al dominio, non sarà più necessario controllare che le celle adiacenti a quella presa in esame siano ai bordi del dominio stesso.

É stato inizialmente ipotizzato l'accesso alla matrice considerandola come un array, in accordo con la rappresentazione in memoria delle matrici fornita dal linguaggio C. Ciò avrebbe permesso di scorrere in modo sequenziale la matrice stessa, risparmiando le chiamate alla funzione `IDX`.

Ad esempio, l'operazione di incremento di energia sarebbe stata implementata nel seguente modo:

```
for (int i = 0; i < n; i++) {
    grid[i] += delta;
}
```

Tuttavia, l'introduzione della ghost area atta a ridurre l'impatto di prestazioni in altre aree della soluzione proposta ha reso preferibile (in termini di gestione dell'accesso alla memoria) l'utilizzo della funzione `IDX`, che contribuisce inoltre ad una buona comprensione e leggibilità del codice sorgente.

2.1 Implementazione OpenMP

L'inizializzazione della ghost area a zero non è stata parallelizzata poiché i test effettuati hanno mostrato un degradamento, seppur minimo, delle prestazioni.

Lato dominio	Numero passi	T_{setup} (s)
512	1000	0.00337177
512	1000	0.00454419

2.2 Implementazione CUDA

Il dominio è stato partizionato, come da suggerimenti, in blocchi 2D organizzati a loro volta in griglie 2D.

Per le operazioni effettuate nelle funzioni `count_cells` e `average_energy` il dominio è stato invece partizionato in blocchi monodimensionali, in accordo con la rappresentazione matriciale nel linguaggio di programmazione C.

A tal proposito, per le sopracitate operazioni, è stata opportunamente implementata l'operazione di riduzione mediante operatore somma.

In un primo momento, per semplicità, si è proceduto con la realizzazione della soluzione senza l'utilizzo della memoria condivisa del Device.

3 Valutazione prestazioni

AMD Opteron (tm) Processor 6376 8 core 16 thread.

Le tempistiche indicate sono state tutte ottenute mediante le funzioni di libreria messe a disposizione dal docente. Per ulteriori informazioni si consulti il codice sorgente.

3.1 Speedup

Il calcolo dello speedup viene eseguito mediante la seguente formula:

$$S(p) = \frac{T_{serial}}{T_{parallel}(p)}$$

in cui:

p	:	# processori/core
T_{serial}	:	tempo di esecuzione della porzione seriale (T_{serial}=$T_{parallel}(1)$)
$T_{parallel}(p)$	:	tempo di esecuzione della porzione con p processori/core

Poiché l'implementazione della soluzione contiene porzioni di codice non parallelizzabili (si veda la funzione `setup`), dovremmo considerare $T_{parallel}(p)$ nel seguente modo:

$$T_{parallel}(p) = \alpha * T_{serial} + \frac{(1-\alpha) * T_{serial}}{p}$$

in cui α è il fattore relativo alla porzione di codice non parallelizzabile, calcolato come segue:

$$\alpha = \frac{T_{serial}}{T_{serial} + T_{parallel}}.$$

Tuttavia, come mostrato nella seguente tabella, la porzione di codice seriale (non parallelizzabile) impiega tempo trascurabile rispetto alla porzione di codice parallelizzata, pertanto anche α assume valore trascurabile.

Lato	Numero passi	$T_{seriale}$ (s)	$T_{parallelo}$ (s)	α
256	100000	0.00140063	14.2519	0.00009827
1024	100000	0.02356447	150.2037	0.00015686

Per le siffatte osservazioni, considereremo $T_{parallel}(p)$ in tal modo:

$$T_{parallel}(p) = \frac{T_{serial}}{p}.$$

4 Conclusioni

4.1 Note di sviluppo

Lo sviluppo dell'implementazione del modello è stato accompagnato dalla realizzazione di opportuni script orientati a rendere più rapido il testing sul server fornito dal docente.

Sono stati infatti realizzati meccanismi per il trasferimento, compilazione, esecuzione e ottenimento dei risultati da/verso il server utilizzando il programma `scp` e l'opzione `ProxyJump` di ssh. A tal proposito si consultino i file `.autoenv.zsh` e `README.md` del progetto ove è possibile trovare indicazioni e dettagli tecnici su come sfruttare il meccanismo ideato.

References

[1] Robert Burridge and Leon Knopoff. Model and theoretical seismicity. *Bulletin of the seismological society of america*, 57(3):341–371, 1967.