Universitat Autònoma de Barcelona

Facultat de Ciències

Pràctica 1

Autors:

Gerard Lahuerta & Ona Sánchez

1601350 — 1601181

15 de Octubre del 2022

1 Introducció

L'objectiu d'aquesta pràctica és optimitzar el temps d'execució del programa energystorm, programat al fitxer energy_storm.c.
Per tal d'optimitzar el funcionament del programa, es recurrirà a la paralel·lització de totes les instruccions repetitives, sempre i quan sigui possible, que generin un cost computacional elevat, al fitxer energy_storm_omp.c.

2 Anàlisi del problema

A l'estudi realitzat anteriorment sobre els temps de diversos bucles del programa[1], es va veure que la fase que més tarda en executar-se en la simulació és el bucle iniciat a la linia **183**, ja que el percentatge de temps que s'hi inverteix és casi del 100% del total del temps d'execució, pel que els esforços en optimitzar el codi haurien d'anar enfocats a aquest bucle.

Per altra banda, també es va concloure que el bucle iniciat a la línia **198** podria ser paral·lelitzat, pel que s'ha d'estudiar quina és la millor manera d'optimitzar-lo.

S'observen també bucles diversos que podrien unir-se en un de sol, com és el cas dels fors de les línies **175** i **176** (ja que usen el mateix rang de valors per l'índex k i no depenen entre ells), es procedirà, també, a paral·lelitzar aquest nou bucle conjunt per agilitzar els càlculs quan s'ha de fer moltes iteracions del mateix (ja que depen el nobre d'iteracions en funció del nombre de cèlul·les que hi introduïm).

3 Disseny de la solució

Expossem i expliquem ara les seccions del codi modificades:

```
178 #pragma omp parallel for
179 for( k=0; k<layer_size; k++ ){
180     layer[k] = 0.0f;
181     layer_copy[k] = 0.0f;
182 }
```

[1] L'estudi esmentat és el ja entregat anteriorment, si es vol consultar és: Estudi Pràctica 1 CAP

```
200 #pragma omp parallel for
201 for( k=0; k<layer_size; k++ ) {
202     /* Update the energy value for the cell */
203     update( layer, layer_size, k, position, energy );
204 }

209 #pragma omp parallel for
210 for( k=0; k<layer_size; k++ )
211     layer_copy[k] = layer[k];

228 #pragma omp parallel for ordered
229 for( k=1; k<layer_size-1; k++ ) {
230     /* Check it only if it is a local maximum */
231     if ( layer[k] > layer[k-1] && layer[k] > layer[k+1] ) {
232         #pragma omp ordered
233         if ( layer[k] > maximum[i] ) {
234             maximum[i] = layer[k];
235             positions[i] = k;
236         }
237     }
238 }
```

S'ha simplificat la paral·lelització de la millor manera possible per tal d'optimitzar al màxim el codi. En els tres primers blocs, com el treball és igual independentment de la iteració en que es trobi el bucle i no hi ha problemes de concurrència, s'ha pogut utilitzar la implementació més senzilla pels bucles de la que disposa l'OpenMP: *#pragma omp parallel for*.

Per altre banda, en l'últim bloc, degut a que es busca el màxim i la posició d'aquest valor la llista *layer*, s'ha hagut de recorrir a la utilització de la clàusula *orderer*, ja que la clàusula reduction (si ve ens permetia obtenir el màxim) no ens permetia obtenir les dues informacions que necessitàvem.
Tot i així, la implementació és suficienment bona, ja que només recorrim a ella una vegada la condició del primer for es compleix, pel que una gran quantitat de valors no accedeixen a aquesta zona no paral·lelitzada del for.

En cas de voler consultar les zones modificades, en les seccions de codi hi ha indicades les files on es troben les comandes al fitxer energy_storm_omp.c.

4 Resultat

	TEMPS				
	SEQÜENCIAL	PARAL·LELITZAT			
		2 fluxos	4 fluxos	6 fluxos	12 fluxos
Test 2	43.369	22.099	11.168	5.870	4.194
Test 7	241.569	122.442	61.390	30.969	20.755
Test 8	5.260	3.120	2.038	1.404	1.275
	Acceleració				
Test 2	-	1.962	3.883	7.388	10.340
Test 7	-	1.972	3.934	7.800	11.639
Test 8	-	1.685	2.580	3.746	4.125

S'observa com s'ha optimitzat el programa obtenint una millora de fins a 11.639 vegades més ràpid amb 12 fluxos en el test 7.

Concluïm que hem assolit els objectius millorant zones que inicialment no havíem considerat (com el bucle de la línia 198) i millorant les que ja s'havien estudiat anteriorment.

5 Principals problemes

Enumerem ara els principals problemes que han aparegut en el nostre procés d'optimitzacdió del codi:

1. Optimització del bucle de la linia 200:

   Inicialment voliem paral·lelitzar el bucle que el contenia pero per problemes amb dades, que estaven correlacionades, ens era impossible pel que és va decidir paral·lelitzar aquest per millorar el temps d'execució.

2. Optimització del bucle de la linia 228:

   Inicialment es va probar a optimitzar el bucle utilitzant clàusules com *private*, *lasprivate* i *reduction* però no aconseguiem obtenir la posició màxima; pel que es va recurrir (eventualment) a la clàusula *orderer* per a poder obtenir una millora del temps d'execució crivant valors que no complien el primer requisit (amb l'if del primer for) i a la vegada, executant de manera seqüencial el procès d'obtenció del màxim per, així, obtenir els dos valors requerits.