High Performance Computing
Departamento de Ingeniería en Informática
LAB1 : SIMD- OpenMP

1 Objetivo

El objetivo de este laboratorio es conocer, usar y apreciar las características de paralelismo a nivel de hebra usando OpenMP.

En este laboratorio usted implementará un sistema pequeño de simulación de partículas que impactan un material.

2 El bombardeo de partículas

La idea principal de este lab es simular la energía depositada en algún material por partículas de alta energía que impactan dicho material, como por ejemplo el bombardeo de partículas en un satélite. El sistema calculará y registrará la energía en Joules que cada partícula deposita en un lugar del material.

Para simplificar, simularemos un material unidimensional, y por lo tanto usaremos un arreglo 1D para acumular las energías depositadas. La Figura 1 ejemplifica el bombardeo de partículas y el registro de las energías

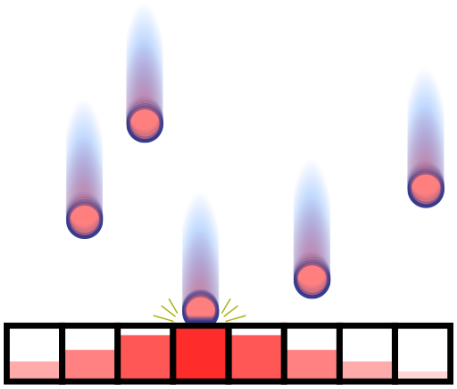

Figure 1: Deposición de energías por partículas.

Para cada partícula, se especifica la energía que trae y la posición de impacto. Aunque cada partícula impacta un lugar específico del material, la energía que deposita se extiende por todo el material, pero en función inversa a la distancia de impacto.

Sea E_p la energía de la partícula p, y sea j la posición de impacto. Luego la energía depositada en cada celda i del material es

$$E_i = E_i + \frac{10^3 \times E_p}{N\sqrt{|j-i+1|}}, \quad \forall\, i = 0, \ldots, N-1$$

donde N es el número de celdas del material. Note que es posible que la posición de impacto sea más grande que N. Esto es válido, y significa que aunque la partícula impactó fuera del material medido, su energía igualmente se registra en la zona de interés.

Cuando la energía depositada es muy pequeña, el sistema no la registrará. Para eso usaremos un umbral `MIN_ENERGY` $= 10^{-3}/N$. Luego, solo cuando $E_i \geq$ `MIN_ENERGY`, se depositará la energía.

2.1 Los archivos de entrada y de salida

Un bombardeo de partículas se especifica en un archivo de entrada, el cual contiene el número de partículas, y para cada partícula, la posición de impacto y la energía (E_p). El siguiente es un ejemplo de un archivo de bombardeo:

```
6
4 81
8 10
10 100
7 35
11 12
5 8
```

El primer número es el número de partículas, y luego viene la descripción de las partículas, una por línea. Note que en este archivo NO se especifica el número de celdas.

El simulador debe escribir a un archivo de salida la posición y valor de máxima energía, al final de la simulación, y luego las energías acumuladas en cada posición de la celda, una por línea. Por ejemplo:

```
23 780150.187500
0 762922.0625
1 765273.6875
2 767356.2500
3 769844.8125
...
```

NOTA: el programa siempre debe imprimir por `stdout` el tiempo de ejecución, sin incluir operaciones de entrada/salida.

3 El programa

El simulador se ejecutará de la siguiente manera:

```
$ ./bomb -t numero_hebras -N numero_celdas -i input.txt -o output.txt -D debug
```

donde

- `-t`: es el número de hebras
- `-N`: es el número de celdas
- `-i`: es el archivo con el bombardeo
- `-o`: es el archivo de salida

- `-D`: especifica si imprime por stdout o no. `debug=1` imprime, y `debug=0` no imprime.

La opción de debug permite imprimir por `stdout` un gráfico simple y normalizado de las energías. Para esto se usará la siguiente función externa, la cual debe declarar en su código:

```
extern void niceprint(int N, float *Energy);
```

donde los argumentos son el número de celdas, y el arreglo con las energías.

Esta función se provee en forma de librería estática llamada `libniceprint.a` la cual debe ser linkeada a su código. El siguiente es el ejemplo de salida usando esta función. El archivo `test1.txt` de entrada es:

```
8
2 620
4 3
5 10
9 2
26 30
28 30
32 530
33 20
```

Y la salida usando `N=35` y debug=1 es

```
Tiempo de simulación: 0.000014

0   2175.7537 |oooooooooooooooooooooooooooooooo
1   2371.4814 |ooooooooooooooooooooooooooooooooo
2   2632.4529 |oooooooooooooooooooooooooooooooooo
3   3088.0391 |ooooooooooooooooooooooooooooooooooo
4   3277.5754 |oooooooooooooooooooooooooooooooooooo
5   3292.6814 |oooooooooooooooooooooooooooooooooooo
6   3062.5535 |ooooooooooooooooooooooooooooooooooo
7   3084.5583 |ooooooooooooooooooooooooooooooooooo
8   3228.8975 |oooooooooooooooooooooooooooooooooooo
9   3615.6855 |ooooooooooooooooooooooooooooooooooooo
10   3639.9268 |ooooooooooooooooooooooooooooooooooooo Max
11   3459.7246 |ooooooooooooooooooooooooooooooooooooo
12   2954.7439 |oooooooooooooooooooooooooooooooooooo
13   2707.8113 |ooooooooooooooooooooooooooooooooooo
14   2601.6592 |ooooooooooooooooooooooooooooooooooo
15   2476.2971 |oooooooooooooooooooooooooooooooooo
16   2333.6682 |oooooooooooooooooooooooooooooooooo
17   2142.6091 |ooooooooooooooooooooooooooooooooo
18   2013.4703 |ooooooooooooooooooooooooooooooooo
19   1914.1119 |oooooooooooooooooooooooooooooooo
20   1833.8130 |oooooooooooooooooooooooooooooooo
```

```
21   1767.4956 |oooooooooooooooooooooooooooo
22   1712.5001 |ooooooooooooooooooooooooooo
23   1667.6620 |oooooooooooooooooooooooooo
24   1633.3096 |ooooooooooooooooooooooooo
25   1612.5475 |oooooooooooooooooooooooo
26   1618.7950 |ooooooooooooooooooooooo
27   1593.3021 |oooooooooooooooooooooo
28   1546.1212 |ooooooooooooooooooooo
29   1466.3090 |oooooooooooooooooooo
30   1411.6680 |ooooooooooooooooooo
31   1367.7074 |oooooooooooooooooo
32   1330.0363 |ooooooooooooooooo
33   1296.6825 |oooooooooooooooo
34   1265.7313 |ooooooooooooooooo
```

Note que el tiempo de simulación NO es escrito por la función `niceprint()`, sino por el programa principal, que mide el tiempo de ejecución.

4 Estrategia de paralelización

Usted debe elegir cómo paralelizar el trabajo entre las hebras del equipo. Para ello considere lo siguiente:

1. El archivo de entrada debe ser leído a memoria por la hebra principal.
2. El archivo de salida y la salida a stdout (cuando debug=1= debe ser realizada solo por la hebra principal
3. El número de hebras, para este lab, será siempre menor a 16.
4. Puede trabajar con un solo arreglo ded energías y actualizar los valores en secciones críticas o atómicas
5. Puede trabajar con un arreglo de energías por hebra y al finalizar, actualizar el arreglo resultante, usando secciones críticas o cálculos atómicos.
6. Puede proponer su propio método de paralelización.

5 Entregables

Tarree, comprima y envíe a `fernando.rannou@usach.cl` al menos los siguientes archivos:

1. `Makefile`: archivo para make que compila los programas
2. `bomb.c`: archivo con el código. Puede incluir otros archivos fuentes.

Fecha de entrega:
domingo 10 de octubre a más tardar a las 23:59 hrs.