گزارش تمرین OPENMP

درس:پردازش موازی

دانشجو: مهسا قادران
شماره دانشجویی: 400201048

مشخصات پردازنده:

با استفاده از دستور lscpu

```
(base) mahsa@mahsa:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          4
On-line CPU(s) list:             0-3
Thread(s) per core:              1
Core(s) per socket:              4
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           158
Model name:                      Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz
Stepping:                        9
CPU MHz:                         3100.000
CPU max MHz:                     3500.0000
CPU min MHz:                     800.0000
BogoMIPS:                        6000.00
Virtualization:                  VT-x
L1d cache:                       128 KiB
L1i cache:                       128 KiB
L2 cache:                       1 MiB
L3 cache:                       6 MiB
NUMA node0 CPU(s):               0-3
```

توضیح کد:

کد به صورت object oriented نوشته شده است. به این صورت که هر ذره یک شی از کلاس Particle است. کلاس Particle شامل متغییر های مکان ذره و مکان پیشین ذره، معادلات حرکت ذره و رنگ آن ذره می‌باشد.
توابع کلاس Particle عبارت‌اند از:

```
collision
```

بررسی نقشه مکان ذرات و محاسبه معادلات جدید با استفاده از تابع calculateNewEquations در صورت وجود برخورد.

```
goNextLocation
```

محاسبه مکان بعدی ذره و اضافه کردن مکان جدید ذره به نقشه.

```
calculateNewEquations
```

محاسبه ضرایب فرمول‌های جدید. در صورتی که برخورد رخ داده باشد این تابع فراخوانی می‌شود. و برای هر کدام از پارامترها توابع مختص آن پارامتر را فراخوانی میکند.

```
setNewA
```

محاسبه مقداد جدید a

```
setNewB
```

محاسبه مقدار جدید b

```
setNewC
```

محاسبه مقدار جدید c

```
setNewD
```

محاسبه مقدار جدید d

کاربرد توابع:

```
([Particle* read_record(Particle particles[NPARTICLE
```

از فایل ورودی مشخصات ذرات را خوانده، شی مرتب با ذره را می‌سازد و در آرایه particles ذخیره می‌کند.

```
}([void generate_output(Particle particles[NPARTICLE
```

مکان هر ذره را در فایل خروجی output.txt ذخیره می‌کند.

```
([void calc_col_map(Particle particles[NPARTICLE
```

این تابع مقادیر را از نقشه می‌خواند و در صورتی که تصادم وجود داشته باشد، پارامترهای خواسته شده را محاسبه میکند.

```
()void CleanMap
```

در این تابع با استفاده از دستور memset نقشه به حالت اولیه تمام صفر تبدیل می‌شود.

کد ترتیبی:

الگوریتم کد ترتیبی به این صورت است که ابتدا مقادیر را از فایل ورودی خوانده و لیست particles را می‌سازد.
سپس در یک حلقه تکرار شونده به اندازه تعداد iterationها:

- همه ذره ها یک مرحله با توجه به معادله شان حرکت کرده و مکان جدیدشان در نقشه ذخیره می‌شود.
- سپس همه ذره‌ها با توجه به موقعیت کنونیشان موقعیتشان در نقشه محاسبه میشود.
- در قدم بعدی با گذار از روی کل نقشه، تعداد برخوردها و انرژی های خواسته شده محاسبه میشود.
- نهایتا نقشه به حالت تمام صفر reset میشود.

پس از اتمام تمام iterationها مکان ذرات در خروجی ذخیره شده.
حافظه تخصیص داده شده به ذرات آزاد میشود.
و نهایتا پارامتر های خواسته شده در خروجی نمایش داده میشود.

اجرا کد:

فایل ورودی به صورت input.txt باید باشد.

مقادیر ابعاد نقشه، تعداد ذرات و مقدار انرژی آزاد شده در فایل constant.h می‌باشد که باید همراه با فایل p_particle.cpp, particle.h باشد.
دستور لازم برای complie و اجرا کد به صورت زیر می‌باشد.

Bash run_seq.sh {#N} {#particles}

موازی سازی:

اولین نکته ای که برای موازی سازی کد موجود است این است که برنامه به طوری نوشته شده باشد که قابلیت موازی سازی را داشته باشد و تا جای ممکن از نوشتن همزمان در یک خانه جلوگیری شود.

برای موازی سازی برنامه به وسیلهopenmp. حلقه‌ها بررسی می‌شوند و تا جای ممکن به صورت همزمان threadها هر دور از حلقه را اجرا میکنند.
حلقه iterationها با توجه به ذات ترتیبی که دارد باید هر دوره دقیقا پس از اتمام دوره قبلی اجرا شود.
حلقه بعدی مربوط به محاسبه مکان بعدی ذرات است. تنها پارامتری که نوشتار همزمان دارد خانه‌های map هستند و با توجه به این که احتمال این که چند ذره در یک خانه از نقشه باشند بسیار کم است، در کندی اجرا تاثیر چشم گیری نخواهد داشت. این خط از کد به صورت atomic انجام می‌شود.

حلقه بعدی دوباره روی ذرات است. این حلقه هیچ نوشتار همزمانی ندارد. در نتیجه با استفاده از دستور

```
pragma omp parallel for#
```

تماما به صورت موازی قابل پیاده سازی است.

نهایتا هنگام به بررسی تصادم ها یک حلقه تو در تو روی نقشه داریم. که این حلقه ها با استفاده از دستور

```
pragma omp for collapse(2)#
reduction(+:colisions_blue,colisions_red,colisions_tot, blue_energy,
red_energy) nowait
```

موازی شده اند.
به این معنی که هنگام جمع زده شدن این پارامترها هر thread متغییر درونی به ازای متغییر های ذکر شده ساخته میشود. و پس از اتمام کار با یکدیگر مقادیر محاسبه شده جمع زده می‌شوند.
علاوه بر این حلقه ها با هم collapse می‌شوند.

اجرا کد موازی:

فایل ورودی به صورت input.txt باید باشد.
مقادیر ابعاد نقشه، تعداد ذرات و مقدار انرژی آزاد شده در فایل constant.h می‌باشد که باید همراه با فایل p_particle.cpp, particle.h باشد.
دستور لازم برای complie و اجرا کد به صورت زیر می‌باشد.

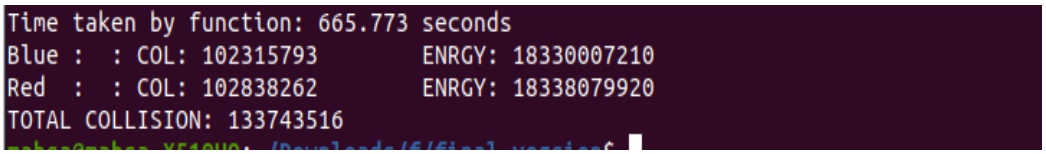
Bash run.sh {#N} {#particles}

مقایسه زمان ترتیبی و موازی:

زمان اجرای برنامه و خروجی در حالت:

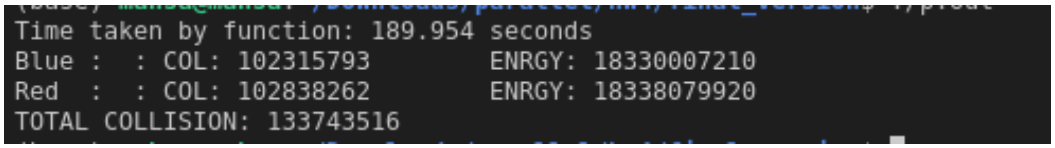
ترتیبی:

زمان اجرا = ۶۶۵ ثانیه



و در حالت موازی:

زمان اجرا = ۱۹۰ ثانیه



که به صورت قابل توجهی در حالت موازی زمان جرا کمتر خواهد بود.