Imageamento de uma Estrela

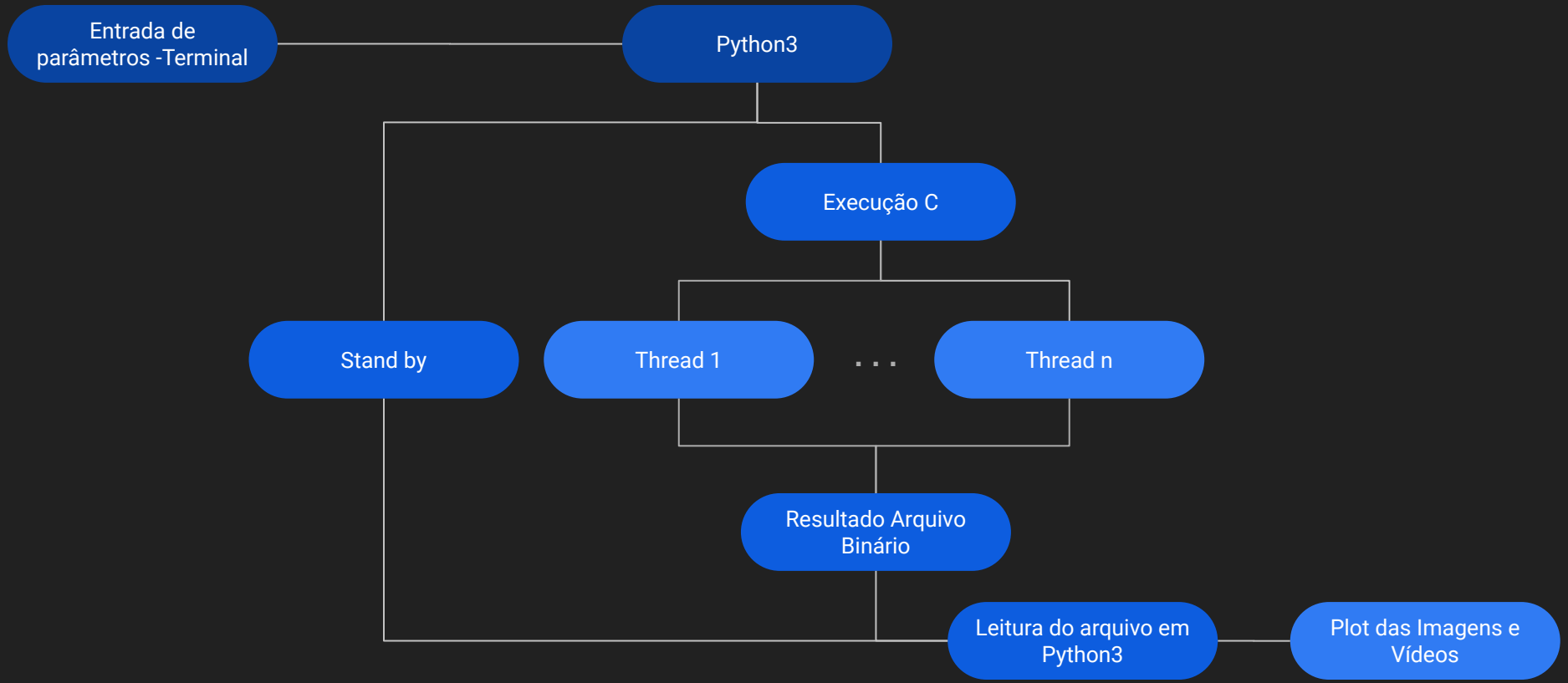
Métodos Computacionais em Astronomia 2020

Matheus J. Castro

O Problema da Estrela

- Gerar fótons com posição e direção de propagação aleatórias;
- Determinar o número de observadores em Latitude e Longitude e calcular em qual observador o fóton foi captado;
- Determinar posições, raios e intensidades aleatórias para manchas solares;
- Repetir o processo inúmeras vezes até se ter fótons suficientes para gerar uma imagem de uma Estrela;
- Ler os dados e plotar imagens e vídeos da Estrela.

Estrutura Base



Execução C

Recebe os parâmetros:

- Número de Fótons;
- Resolução de cada observador;
- Número de Manchas Solares;
- Nome do arquivo em binário;
- Nome do arquivo de log;
- Raio da Estrela;
- Constante de escurecimento de bordo;
- Número de observadores em Longitude;
- Número de observadores em Latitude;
- Número de threads que deverá usar.

Escreve a saída:

- Arquivo binário contendo a informação de todos os observadores;
- Arquivo de log com os parâmetros de execução, tempo levado, etc.

Partes Paralelizáveis

- Inicialização da Array 4D com zeros:

```
133  void initializer_4d_array(const int resolution, const int N_phi, const int N_theta,
134                            float observer_density[MAX_N_phi][MAX_N_theta][MAX_resolution][MAX_resolution]){
135      // Subrotina para inicializar com zeros uma array 4D com dimensões N_phi x N_theta x resolution x resolution.
136      #pragma omp parallel for collapse(4)
137      for(int i=0; i<N_phi; i++){
138          for(int j=0; j<N_theta; j++){
139              for(int k=0; k<resolution; k++){
140                  for(int l=0; l<resolution; l++)
141                      observer_density[i][j][k][l] = 0;
142              }
143          }
144      }
145  }
```

Partes Paralelizáveis

- Cálculo do Fóton:

```
211     #pragma omp parallel private(star_phi, star_mu, star_x, star_y, star_z, dir_x, dir_y, dir_z)
212     {
213         if(omp_get_thread_num() == 0){
214             printf("Executing for %d thread(s).\n", omp_get_num_threads());
215             fprintf(log, "Executing for %d thread(s).\n", omp_get_num_threads());
216         }
217
218         unsigned long int seed = time(0) * (omp_get_thread_num()+1);
219
220         #pragma omp for
221         for(long long count = 0; count < N_PHOTON; count++){  // executa até o número de fotons observados chegar ao esperado
222             star_point(R, &seed, &star_phi, &star_mu, &star_x, &star_y, &star_z);
223             photon_direction(R, a, &seed, &star_z, &star_phi, &dir_x, &dir_y, &dir_z);
224             observer(R, resolution, N_SUNSPOT, star_x, star_y, star_z, dir_x, dir_y, dir_z, N_phi, N_theta,
225                     observer_density, sunspot);
226
227             if(omp_get_thread_num() == 0){
228                 progress = (float)count/N_PHOTON * 100 * omp_get_num_threads();  // calcula o progresso total
229                 if(progress - floor(progress) >= -eps && progress - floor(progress) <= eps)
230                     printf("Progress: %.0f%%\r", progress);
231             }
232         }
233     }
234     printf("\n");
```

Problema dos Números Aleatórios Paralelos

Função rand() não é paralelizável!

```
24  unsigned long int rand_func(unsigned long int *seed){
25      unsigned long int rv;
26      *seed = *seed * 1103515245 + 12345;
27      rv = ((unsigned)(*seed/65536) % RAND_MAX);
28      return rv;
29  }
30
31  void star_point(const double R, unsigned long int *seed, double *star_phi, double *star_mu, double *star_x, double *star_y, double *star_z){
32      // Subrotina que define os parâmetros da posição do fóton na estrela.
33      *star_mu = ((double)rand_func(&*seed)/RAND_MAX)*2-1;     // define um valor entre -1 e 1
34      *star_phi = ((double)rand_func(&*seed)/RAND_MAX)*2*PI;   // define um valor entre 0 e 2pi
35
36      *star_z = R**star_mu;
37      *star_x = R*sqrt(1-pow(*star_mu,2))*cos(*star_phi);
38      *star_y = R*sqrt(1-pow(*star_mu,2))*sin(*star_phi);
39  }
```

Geração do Arquivo Binário

- Vai guardar somente o necessário, não vai desperdiçar bytes;
- Arquivo sequencial, não tem informação de dimensão da Array;
- Informações de dimensão são salvas nos 12 primeiro bytes (3 inteiros de 4 bytes cada);
- Salva floats de 4 bytes;
- O tamanho do arquivo também indica a quantidade de RAM necessária.

Exemplos de tamanho do Arquivo:

1. 61x61 Observadores, Resolução de 1000x1000px:
   - $61^2 x 1000^2 x 4 + 12$ = 14.884.000.012 bytes = 14.9 GB
2. 121x121 Observadores, Resolução de 1000x1000px:
   - $121^2 x 1000^2 x 4 + 12$ = 58.564.000.012 bytes = 58.6 GB

Geração do Arquivo Binário

```
147  void save_density(const int resolution, const int N_phi, const int N_theta,
148                    float observer_density[MAX_N_phi][MAX_N_theta][MAX_resolution][MAX_resolution], const char fl_name[MAX_char]){
149      // Subrotina para salvar as matrizes dos observadores em um arquivo binário, os dados são salvos de forma sequencial.
150      FILE *bin_write;
151      bin_write = fopen(fl_name, "wb");
152
153      fwrite(&N_phi, sizeof(int), 1, bin_write);
154      fwrite(&N_theta, sizeof(int), 1, bin_write);
155      fwrite(&resolution, sizeof(int), 1, bin_write);
156
157      for(int i=0; i<N_phi; i++){
158          for(int j=0; j<N_theta; j++){
159              for(int k=0; k<resolution; k++)
160                  fwrite(observer_density[i][j][k], sizeof(observer_density[i][j][k][0]), resolution, bin_write);
161          }
162      }
163
164      fclose(bin_write);
165  }
```

Execução em Python3

Os parâmetros de execução são recebidos como argumentos pelo terminal.

```
               Help Section
estrela.py v16.4
Usage: python3 estrela.py [options]
Written by Matheus J. Castro <matheusj_castro@usp.br>

Options are:
 -h,  -help                    |       Show this help;
--h, --help                    |       Show this help;
         -r [param]            |       Define star radius. Default is 1;
         -a [param]            |       Define the intensity of limb darkening. Default is 1;
         -np [param]           |       Define the number of photons in the simulation. Default is 1 billion;
         -ns [param]           |       Define the number of sunspot in the simulation. Default is 25;
         -res [param]          |       Define the array resolution that generate the video or image. Default is 1000;
         -nphi [param]         |       Define the number of observers in phi. Default is 65;
         -ntheta [param]       |       Define the number of observers in theta. Default is 65;
         -obs [param]          |       Define the latitude of observers to read. Default is 33;
         -azi [param]          |       Define the azimuth of a single observer.
                               |       If given it will plot an static image instead of a video;
         -c [param]            |       Define the name of c executable file. Default is "estrela.so";
         -fldensity [param]    |       Define the file name with frames of the video. Default is "density.csv";
         -show2d               |       If "-azi" if given, this option show the 2D plot;
         -show3d               |       Show the animated 3D plot;
         -png                  |       If "-azi" if given, this option save the 2D plot;
         -gif                  |       Save animated 3D plot in a gif file;
         -mp4                  |       Save animated 3D plot in a mp4 file;
         -alone                |       Option to run only python. All options given for C program will be ignored.
                               |       It will try to read data from the same directory that this program.
         -dir [param]          |       Specify directory. This will not create a directory for new data.
         -nt [param]           |       Specify the number of threads to execute the OpenMP parallelization.
                               |       Default is 6;
Only those options are recognized!
```

Execução em Python3

Exemplos:

- Execução completa:

```
python3 estrela.py -np 1000000 -res 600 -mp4
```

- Somente leitura dos dados com Python3:

```
python3 estrela.py -alone -dir ~/EP3/data_20201025_11-18 -png -obs 5 -azi 20 -mp4 -gif
```

Obs: proteção contra overwrite:

```
This will overwrite density.bin and log.txt files in the directory data_20201025_11-18/
Are you sure? (yes/no) y
Type "yes" or "no"
Are you sure? (yes/no) no
Execution stopped.
```

Leitura do Arquivo Binário

- Lê somente o necessário (observador especificado por -obs) para economizar tempo de execução e RAM;
- Decodifica as informações de dimensão para trabalhar no resto do arquivo.

```python
def read_density(fl_name, obs_theta):
    # Função para ler o arquivo binário das matrizes dos observadores
    # Cada observador é um frame a ser adicionado ao video

    try:
        fl = open(fl_name, "rb")
    except FileNotFoundError:
        sys.exit("\033[1;31mFile \033[m{}\033[1;31m not found, try to especify the file or the directory to read.\n"
                 "See Help Section (\033[3m-h\033[m\033[1;31m) for more information.\033[m".format(fl_name))

    # Pega os três primeiros valores como int que correspondem, respectivamente, ao
    # número de observadores no azimute, na latitude e a resolução de cada observador
    n_obs = [int.from_bytes(fl.read(4), byteorder="little"),
             int.from_bytes(fl.read(4), byteorder="little"),
             int.from_bytes(fl.read(4), byteorder="little")]

    data_3d = []
    for i in range(n_obs[0]):
        for j in range(n_obs[1]):
            if j != obs_theta:
                fl.seek(4 * n_obs[2] * n_obs[2], 1)  # move o ponteiro no arquivo para achar os bytes a serem lidos
            else:
                data_2d = []
                for m in range(n_obs[2]):
                    data_line = []
                    for n in range(n_obs[2]):
                        data_line.append(struct.unpack("f", fl.read(4))[0])  # le 4 bytes e transforma para float
                    data_2d.append(data_line)
                data_3d.append(data_2d)
    fl.close()

    return np.array(data_3d)
```

Integração Python e C - Biblioteca CTypes

A biblioteca CTypes integra o C dentro do Python. É necessário compilar o C como uma *Shared Library*.

- No compilador GCC adicione a opção *-shared*
- Em sistemas Linux use a extensão para o arquivo de saída *.so*
- Em sistemas Windows use a extensão para o arquivo de saída *.dll*

Etapas para executar o C:

1. Compilar o C como *Shared Library*;
2. Importar a CTypes no código em Python;
3. Definir no Python os parâmetros de entrada e saída do C;
4. Chamar a função/subrotina do C desejada.

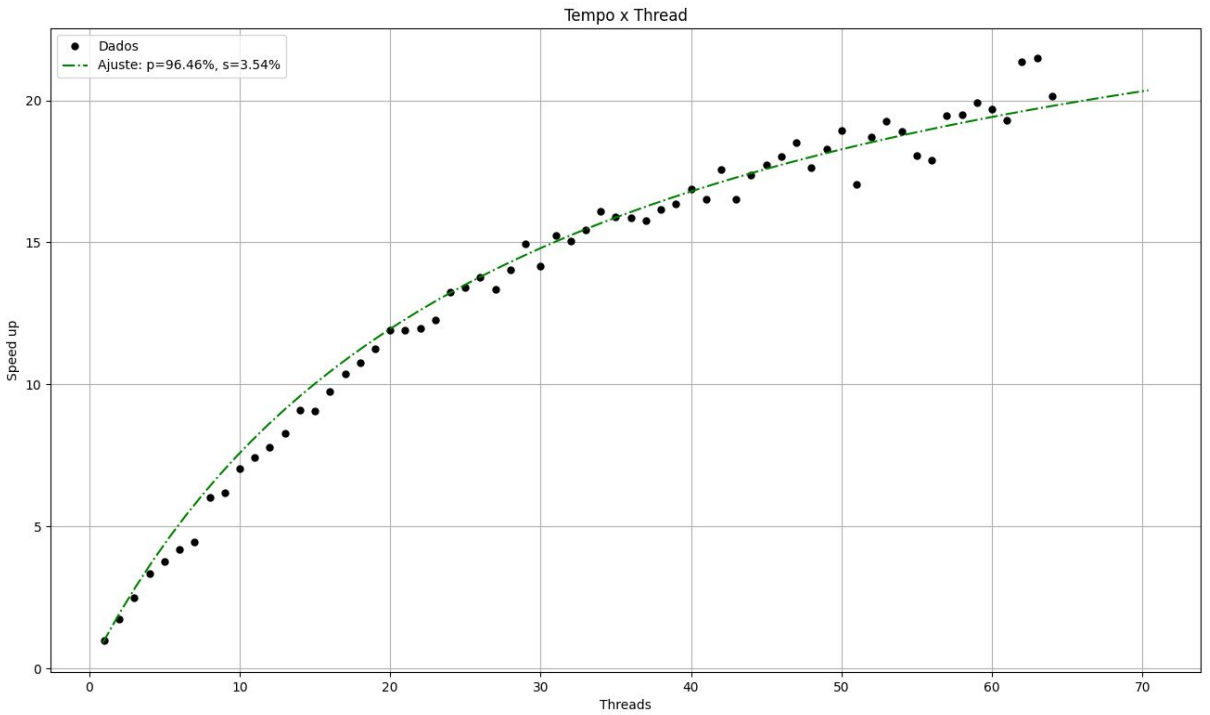
Integração Python e C - Biblioteca CTypes

```python
import ctypes
```

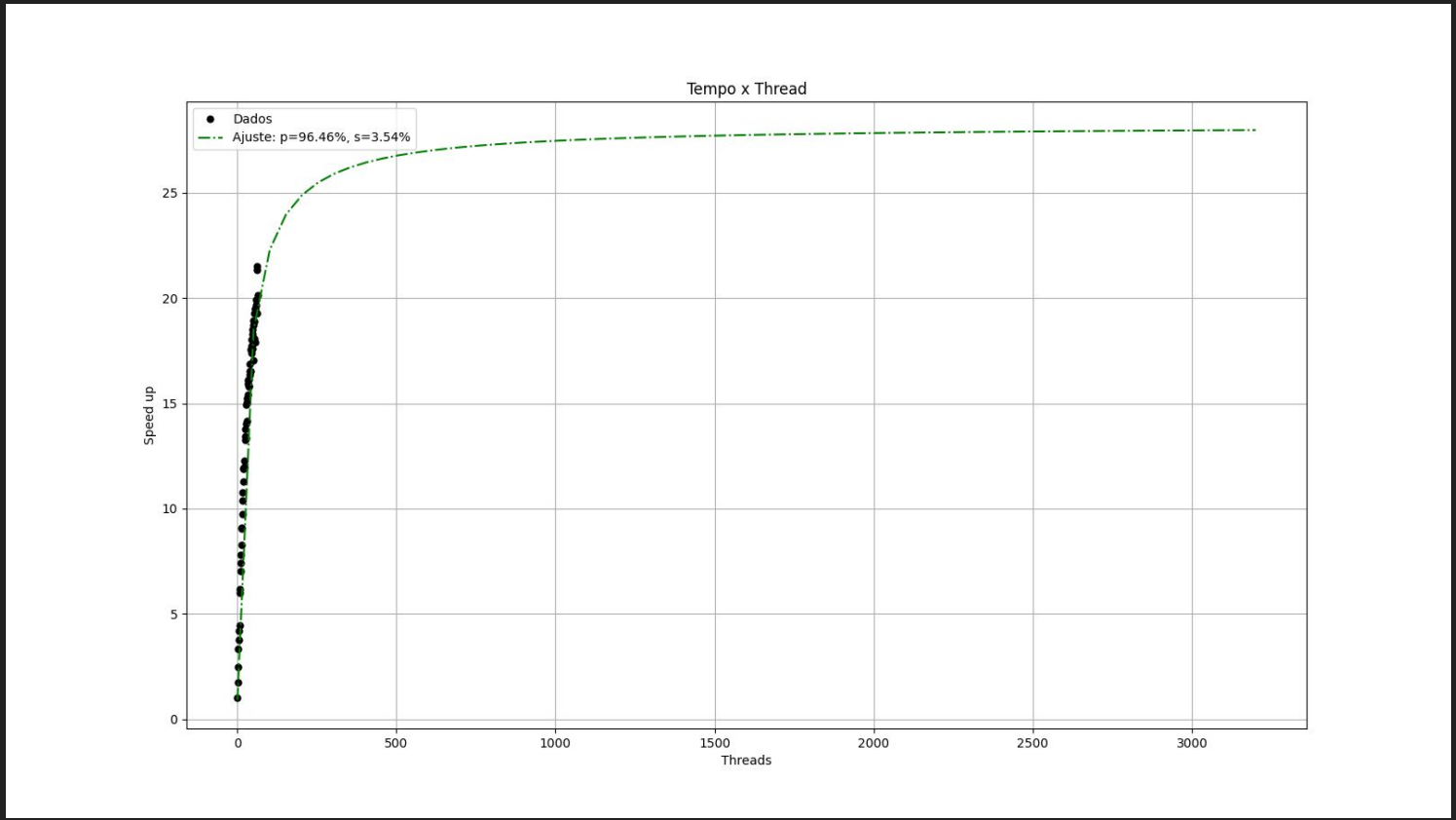
```python
def config_c_call(c_name):
    # Função que configura os aspectos da biblioteca CTypes
    c_lib = ctypes.CDLL("./{}".format(c_name))  # abre o arquivo
    # Define os tipos de entrada da subrotina em questão
    c_lib.main_execution.argtypes = [ctypes.c_char_p, ctypes.c_int, ctypes.c_int,
                                     ctypes.c_char_p, ctypes.c_char_p,
                                     ctypes.c_double, ctypes.c_double, ctypes.c_int, ctypes.c_int, ctypes.c_int]
    # Define os tipos de saída da subrotina em questão
    c_lib.main_execution.restypes = None

    return c_lib
```

```python
    c_lib.main_execution(n_photon.encode("utf-8"), resolution, n_sunspot,
                         fl_density.encode("utf-8"), fl_log.encode("utf-8"),
                         radius, a, n_phi, n_theta, nt)
```

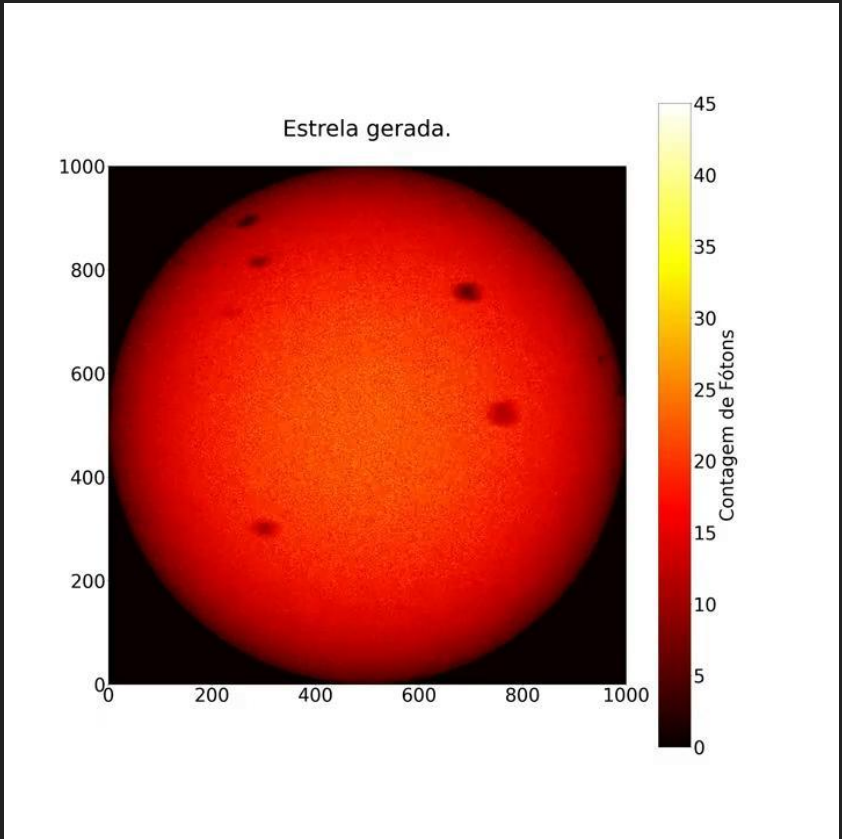
Resultados - Lei de Amdahl



Resultados - Lei de Amdahl

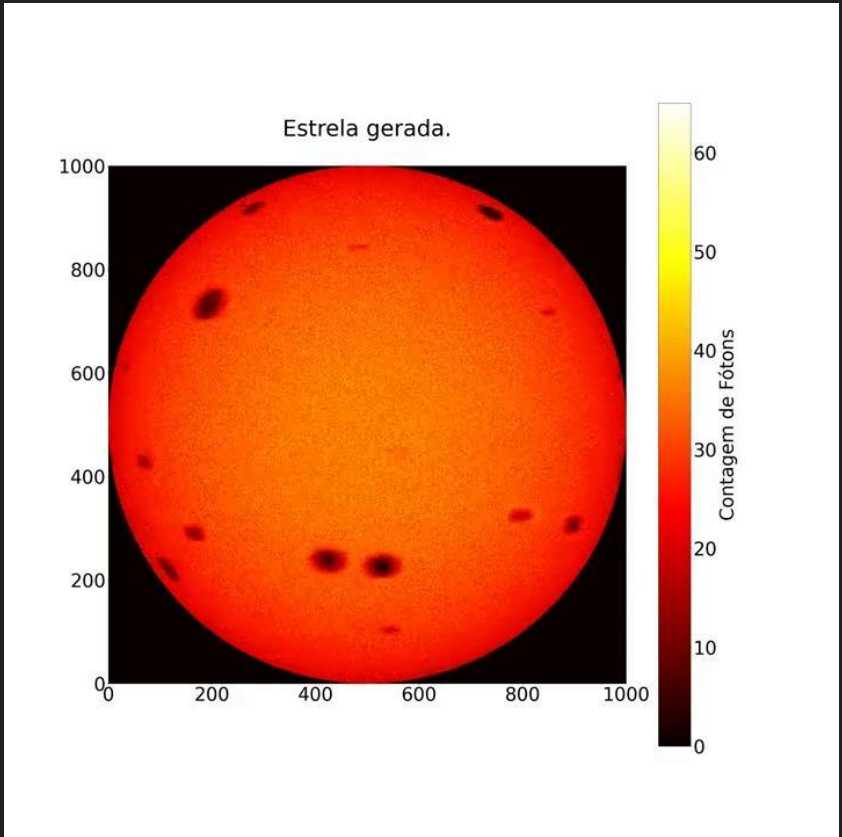


Resultados - Linear 50bi



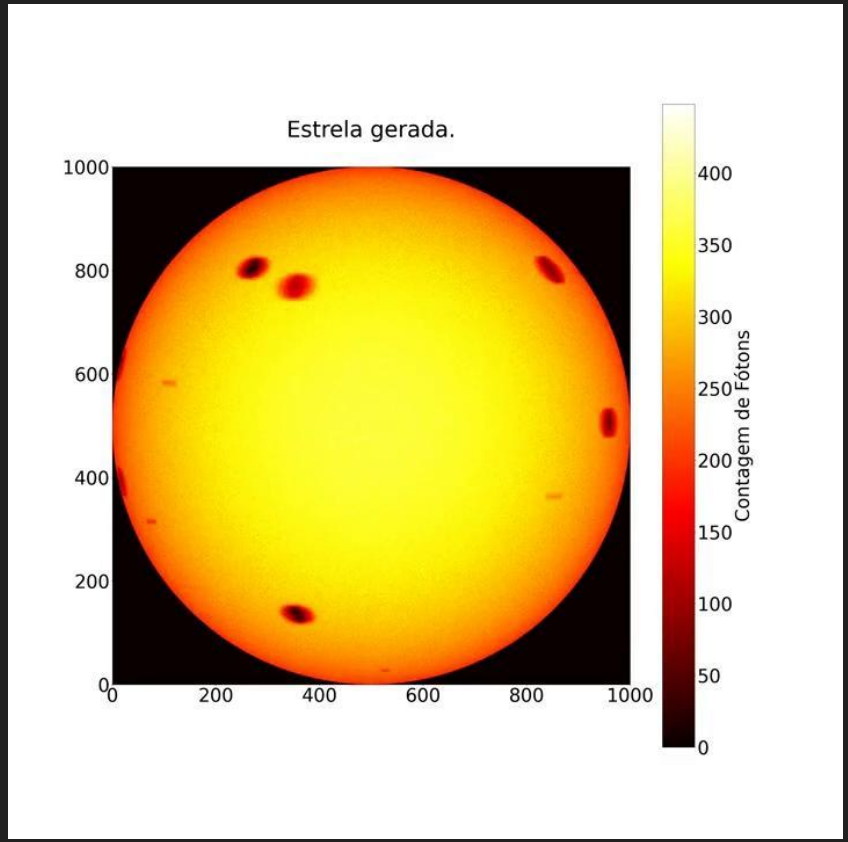

```
 1 Initializing 4D array... Done.
 2 Calculating sunspot parameters... Done.
 3 Calculating photons positions.
 4 Saving... Done.
 5 C execution time: 39209.89s
 6 Reading data.
 7 Ploting data.
 8 Total Time: 58726.99s
 9 Total Time (hh:mm:ss): 16:18:46.99s
10
11 Star relevant data:
12 Star radius (arbitrary unit): 1
13 Limb darkening constant: 20.0
14 Number of photons for each observer: 50000000000
15 Resolution of generated square image: 1000px
16 Number of sunspots: 25
17 Number of observers in azimuth: 65
18 Number of observers in latitude: 65
```

Resultados - Paralelo 100bi



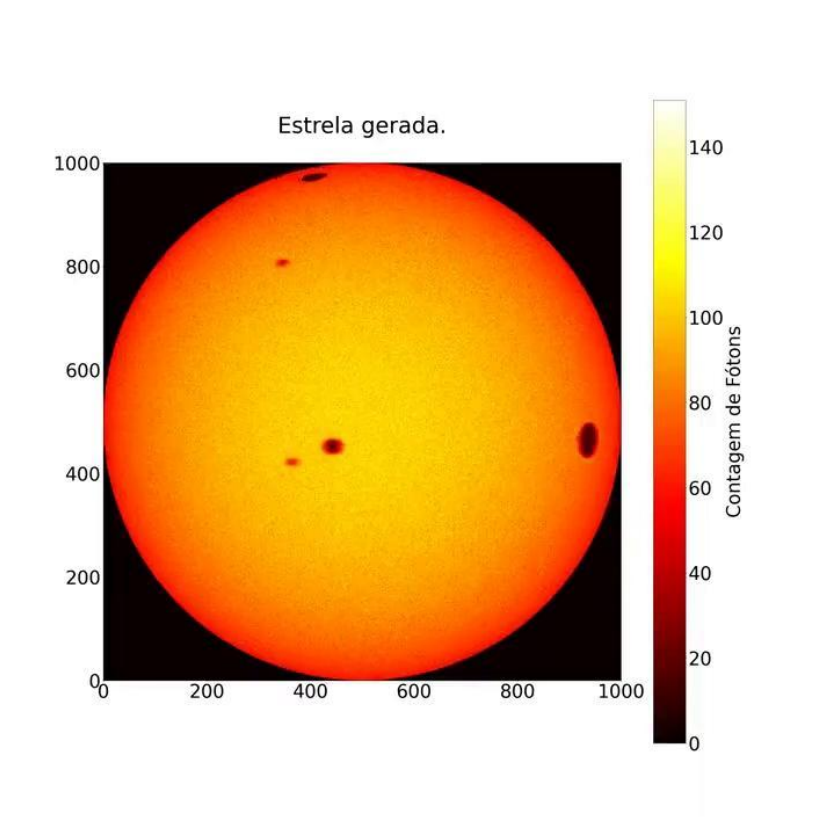

```
 1 Initializing 4D array... Done.
 2 Calculating sunspot parameters... Done.
 3 Calculating photons positions.
 4 Executing for 128 thread(s).
 5 Saving... Done.
 6 C execution time: 839.73s
 7 Total Time: 839.74s
 8 Total Time (hh:mm:ss): 0:13:59.74s
 9
10 Star relevant data:
11 Star radius (arbitrary unit): 1
12 Limb darkening constant: 1
13 Number of photons for each observer: 100000000000
14 Resolution of generated square image: 1000px
15 Number of sunspots: 25
16 Number of observers in azimuth: 65
17 Number of observers in latitude: 65
```

Resultados - Paralelo 1tri



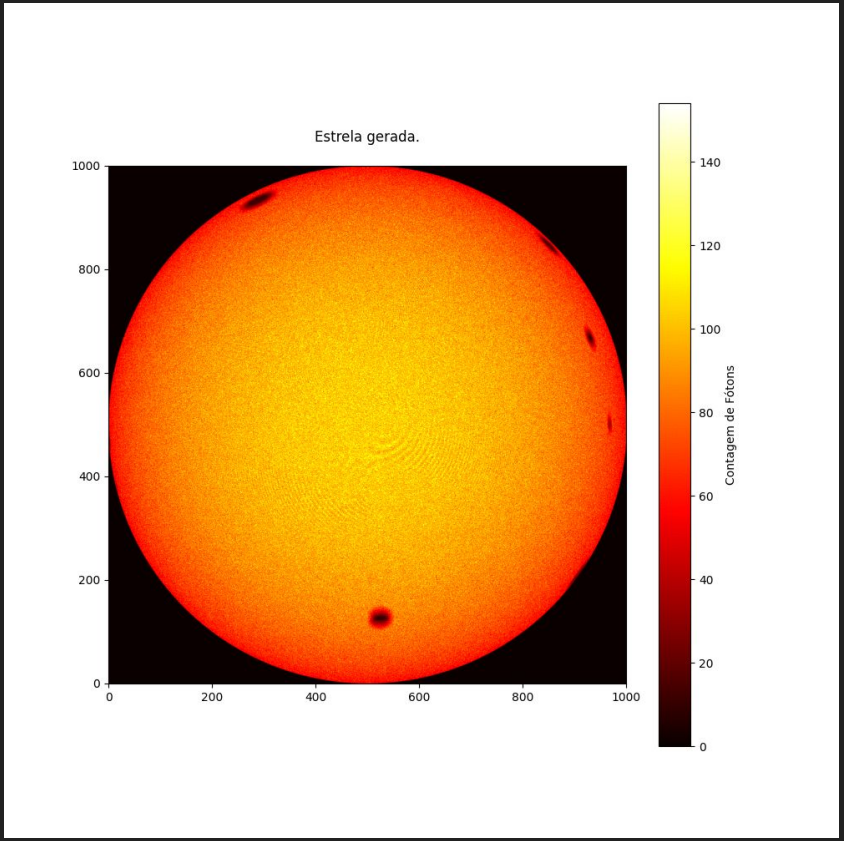

```
 1 Initializing 4D array... Done.
 2 Calculating sunspot parameters... Done.
 3 Calculating photons positions.
 4 Executing for 64 thread(s).
 5 Saving... Done.
 6 C execution time: 16922.48s
 7 Total Time: 16922.48s
 8 Total Time (hh:mm:ss): 4:42:2.48s
 9
10 Star relevant data:
11 Star radius (arbitrary unit): 1
12 Limb darkening constant: 1
13 Number of photons for each observer: 1000000000000
14 Resolution of generated square image: 1000px
15 Number of sunspots: 25
16 Number of observers in azimuth: 65
17 Number of observers in latitude: 65
```

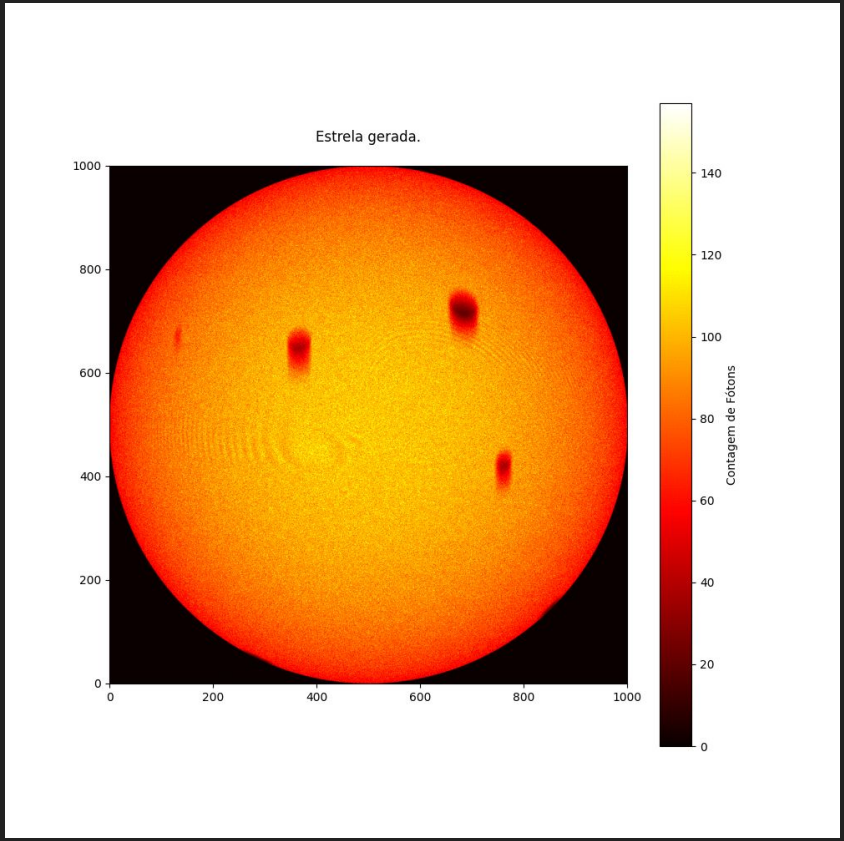
Resultados - Paralelo 1tri, 121 observadores



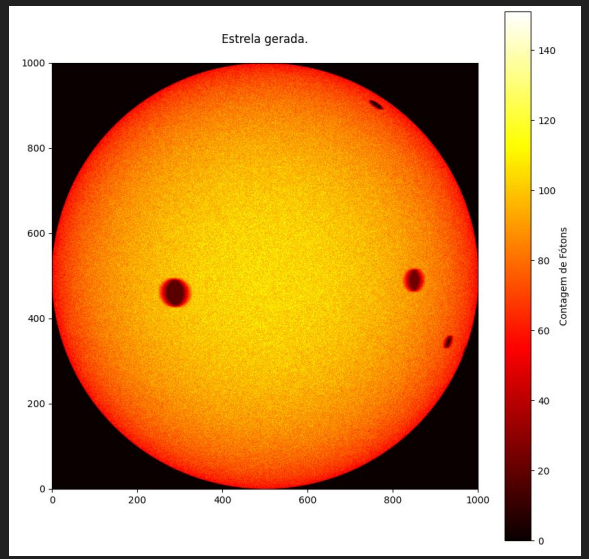

```
 1 Initializing 4D array... Done.
 2 Calculating sunspot parameters... Done.
 3 Calculating photons positions.
 4 Executing for 64 thread(s).
 5 Saving... Done.
 6 C execution time: 15492.07s
 7 Total Time: 15492.07s
 8 Total Time (hh:mm:ss): 4:18:12.07s
 9
10 Star relevant data:
11 Star radius (arbitrary unit): 1
12 Limb darkening constant: 1
13 Number of photons for each observer: 1000000000000
14 Resolution of generated square image: 1000px
15 Number of sunspots: 15
16 Number of observers in azimuth: 121
17 Number of observers in latitude: 121
```

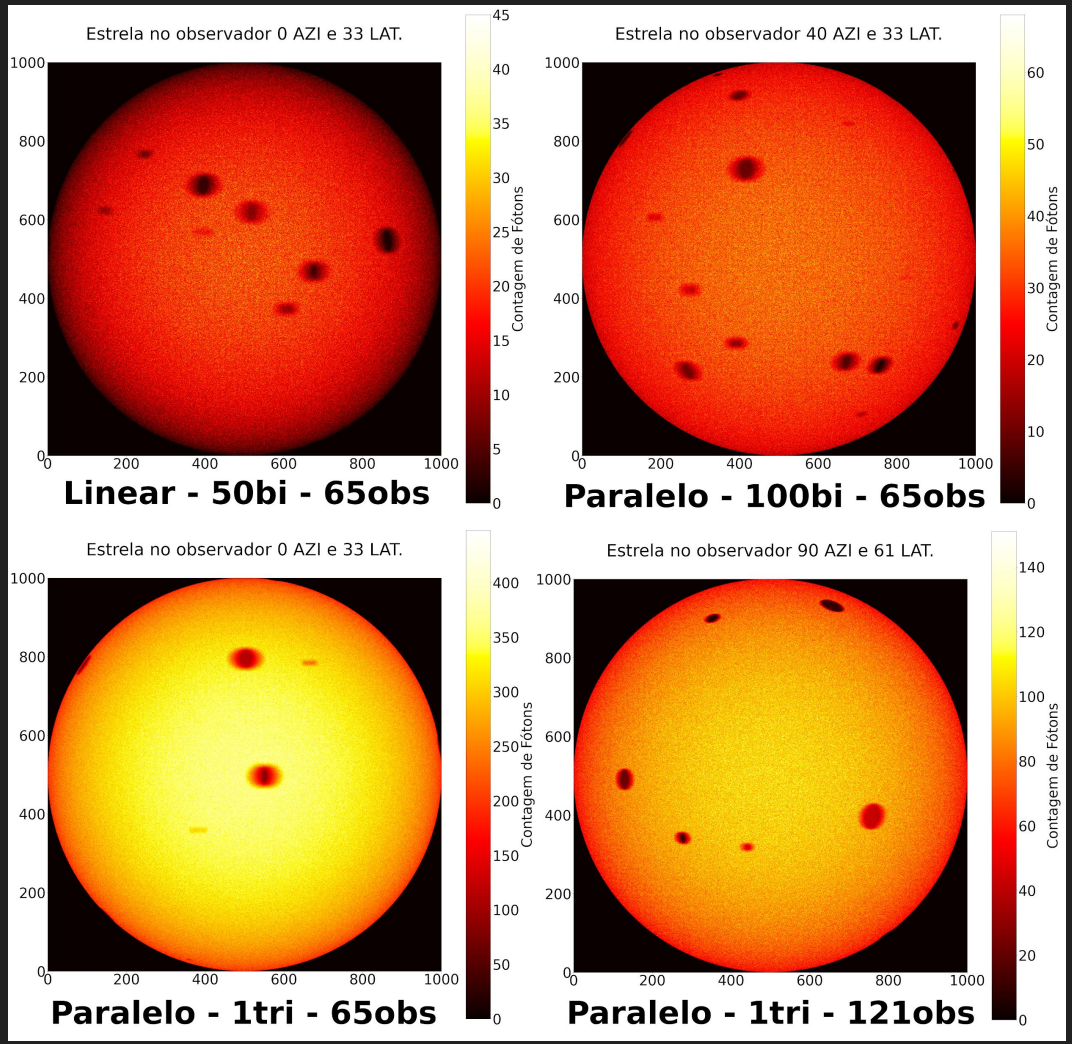
Resultados - Paralelo 1tri - Vários Observadores





Comparação


Estrela gerada.
Contagem de Fótons


Estrela no observador 0 AZI e 33 LAT.
Contagem de Fótons
Linear - 50bi - 65obs
Estrela no observador 40 AZI e 33 LAT.
Contagem de Fótons
Paralelo - 100bi - 65obs
Estrela no observador 0 AZI e 33 LAT.
Contagem de Fótons
Paralelo - 1tri - 65obs
Estrela no observador 90 AZI e 61 LAT.
Contagem de Fótons
Paralelo - 1tri - 121obs