Report on task 2 (template)

**Name: Igenov Temirlan**

**Group: bd-2004**

**E-mail: 201107@astanait.edu.kz**

**Main part:**

Step 0:

We will be using the sequential ray tracing program from Task 1. Download and install Mini-Rt library (https://github.com/georgy-schukin/mini-rt), if necessary.

Изображение выглядит как текст

Автоматически созданное описание

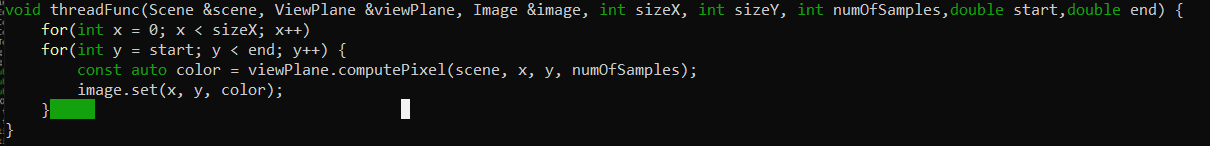
Step 1: Prepare a directory for the Task 2

Изображение выглядит как текст

Автоматически созданное описание

Step 2: Implement static scheduling with POSIX/C++ threads

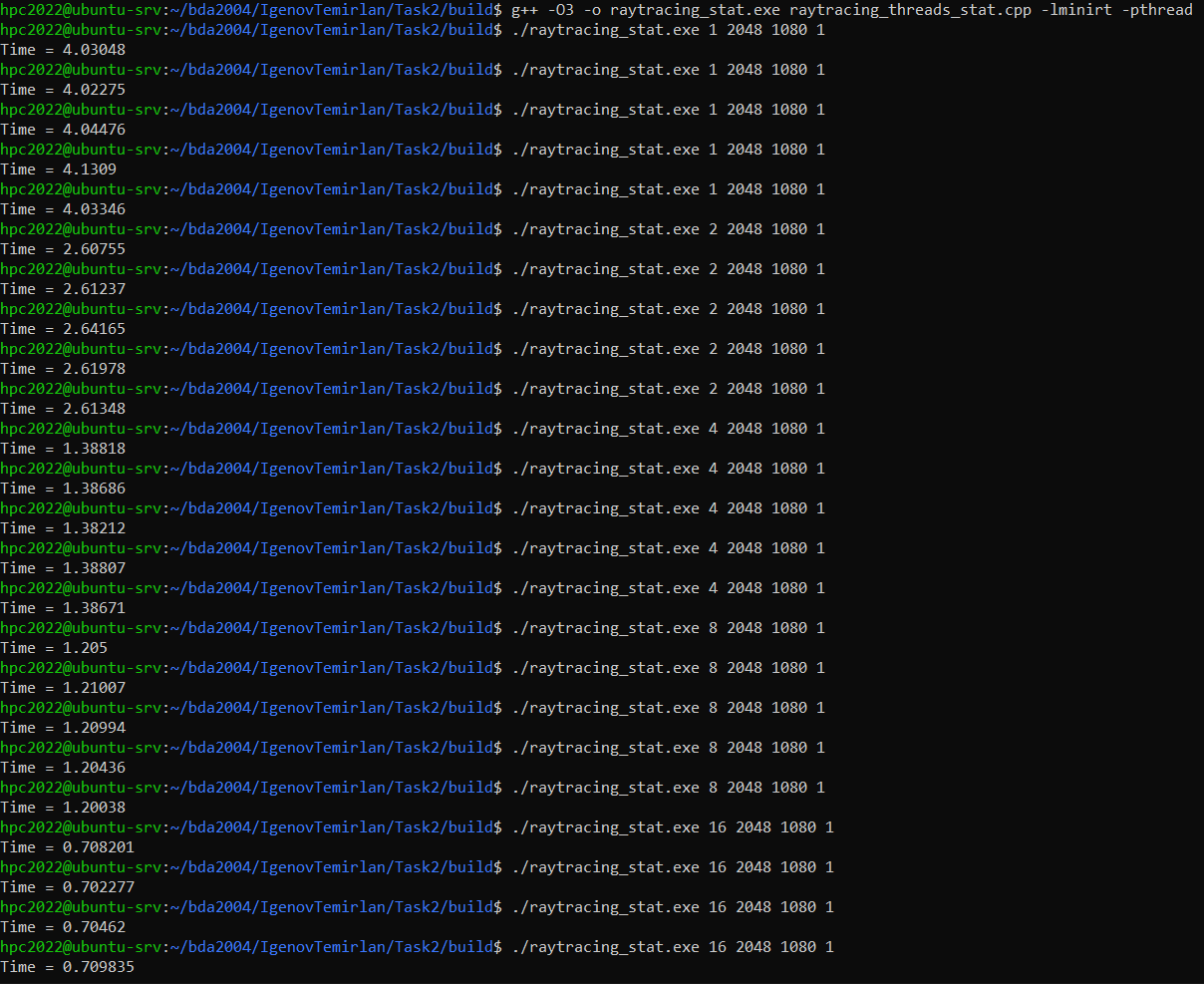
I decide divide by column. Below my code:

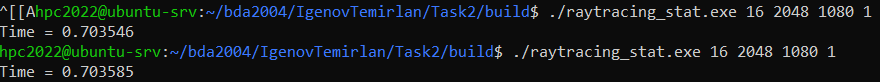


Изображение выглядит как текст

Автоматически созданное описание

Step 3: Study performance of your parallel program



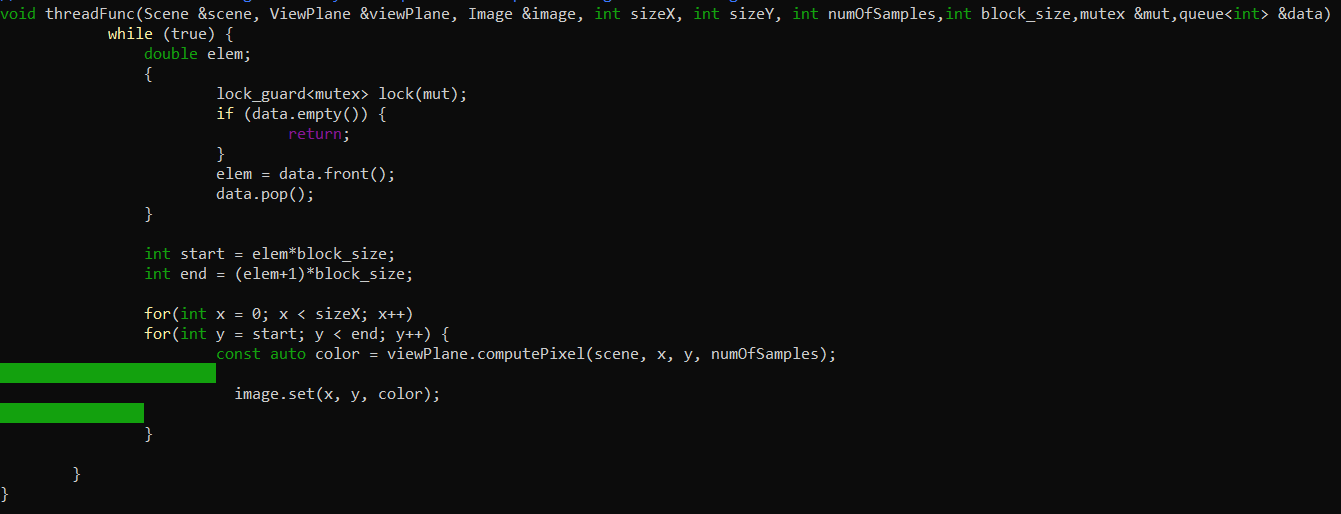


**Table of execution time, speedup and efficiency**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Threads | Run 1 (s) | Run 2 (s) | Run 3 (s) | Run 4 (s) | Run 5 (s) | **Min. time(s)** | Speedup | Efficiency |
| 1 | 4.03048 | 4.02275 | 4.04476 | 4.1309 | 4.03346 | **4.02275** | 100% | 100% |
| 2 | 2.60755 | 2.61237 | 2.64165 | 2.61978 | 2.61348 | **2.60755** | 154% | 77% |
| 4 | 1.38818 | 1.38686 | 1.38212 | 1.38807 | 1.38671 | **1.38212** | 291% | 72.75% |
| 8 | 1.205 | 1.21007 | 1.20994 | 1.20436 | 1.20038 | **1.20038** | 335% | 41.875% |
| 16 | 0.708201 | 0.702277 | 0.70462 | 0.709835 | 0.703585 | **0.702207** | 572.9% | 35.7% |

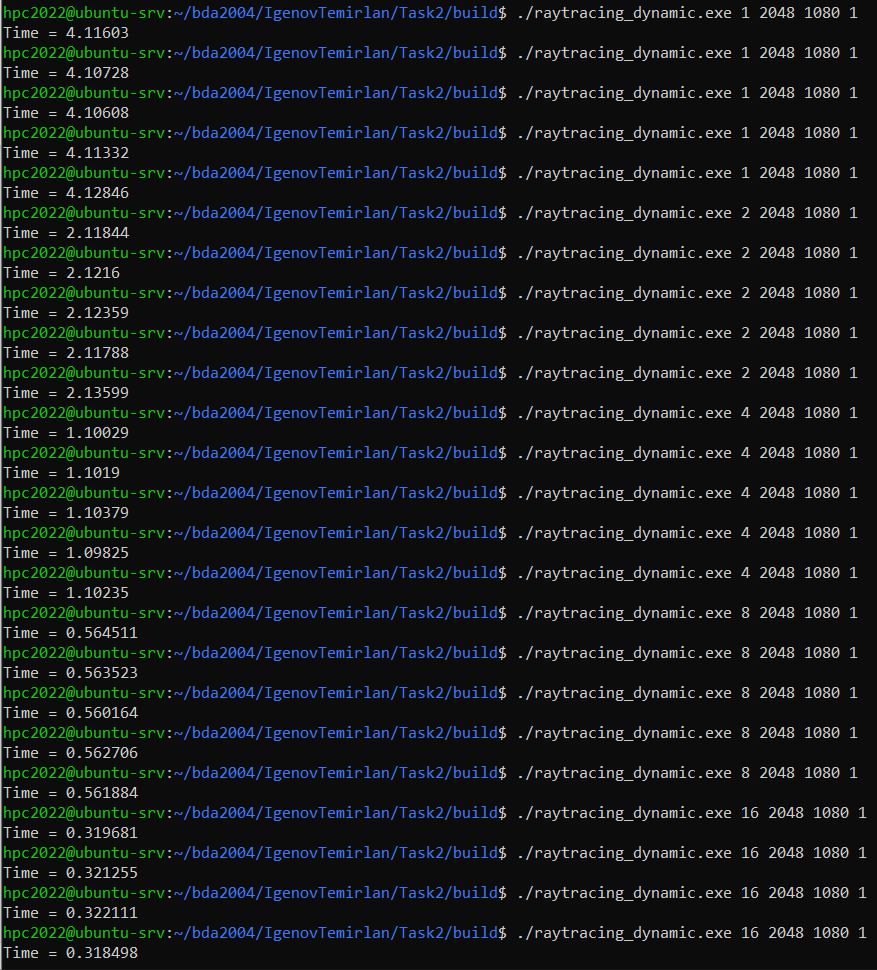
Step 4\*: Implement dynamic scheduling with POSIX/C++ threads

I use code from example. Below you can see how I add it to my work:



Изображение выглядит как текст

Автоматически созданное описание

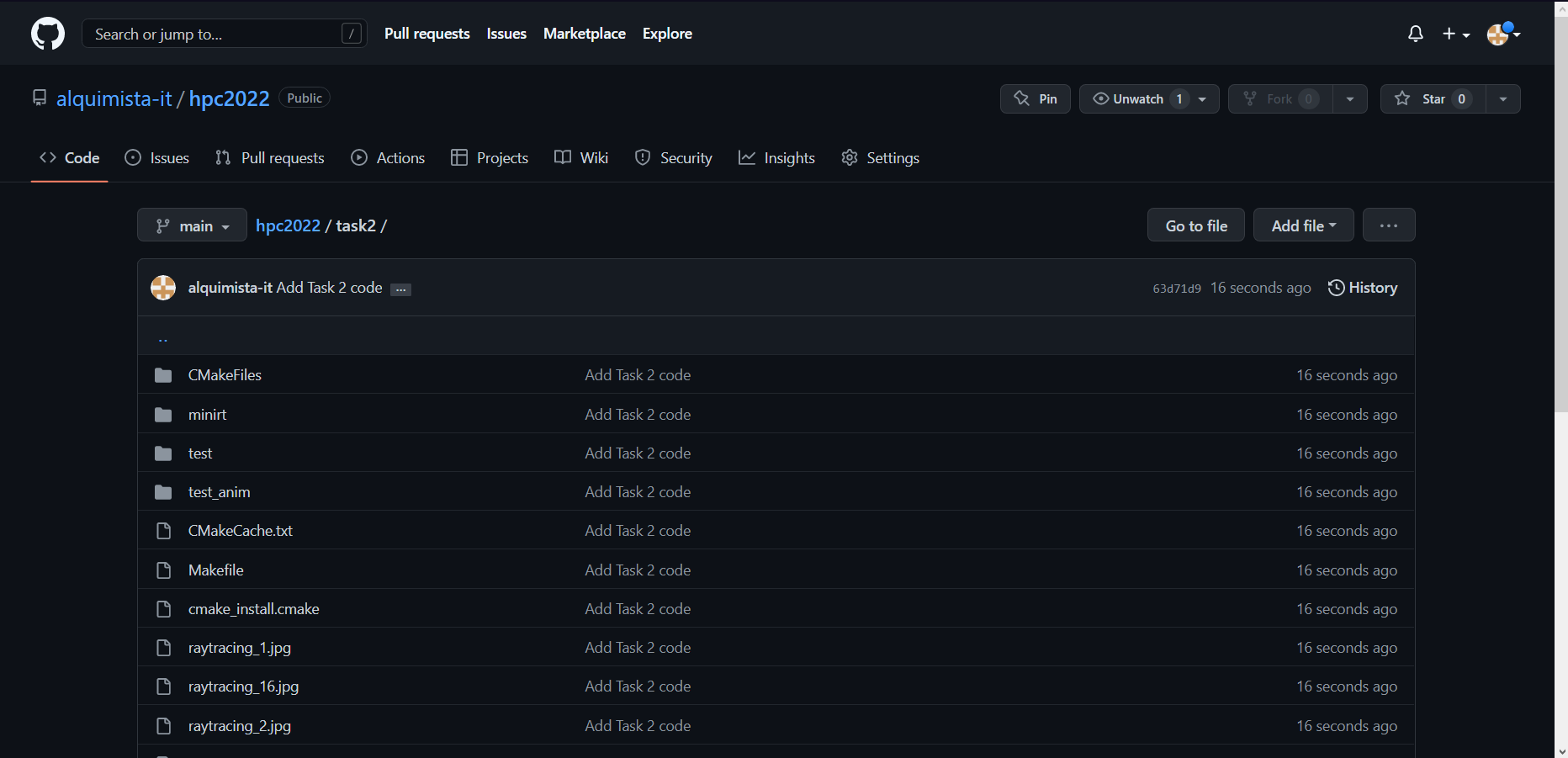
****

****

**Table of execution time, speedup and efficiency (dynamic)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Threads | Run 1 (s) | Run 2 (s) | Run 3 (s) | Run 4 (s) | Run 5 (s) | **Min. time(s)** | Speedup | Efficiency |
| 1 | 4.11603 | 4.10728 | 4.10608 | 4.11332 | 4.12846 | **4.10608** | 100% | 100% |
| 2 | 2.11844 | 2.1216 | 2.12359 | 2.11788 | 2.13599 | **2.11788** | 193.6% | 96.8% |
| 4 | 1.10029 | 1.1019 | 1.10379 | 1.09825 | 1.10235 | **1.09825** | 373.9% | 93.475% |
| 8 | 0.564511 | 0.563523 | 0.560164 | 0.562706 | 0.560164 | **0.560164** | 733% | 91.625% |
| 16 | 0.319681 | 0.321255 | 0.322111 | 0.318468 | 0.317558 | **0.317558** | 1301% | 81.3125% |

Step 5: Commit and push your changes to the Gitlab server

****

<https://github.com/alquimista-it/hpc2022/tree/main/task2>

Step 6: Conclusion in a free form

Based on the 2 tasks, I realized that the static separation of threads works in order, when the dynamic separation of threads is taken for the next block at the end of its own. That is, there the threads take the work that is in the queue and take it from there as they complete their own. And in the static one, the corresponding queue has already been defined for each thread.