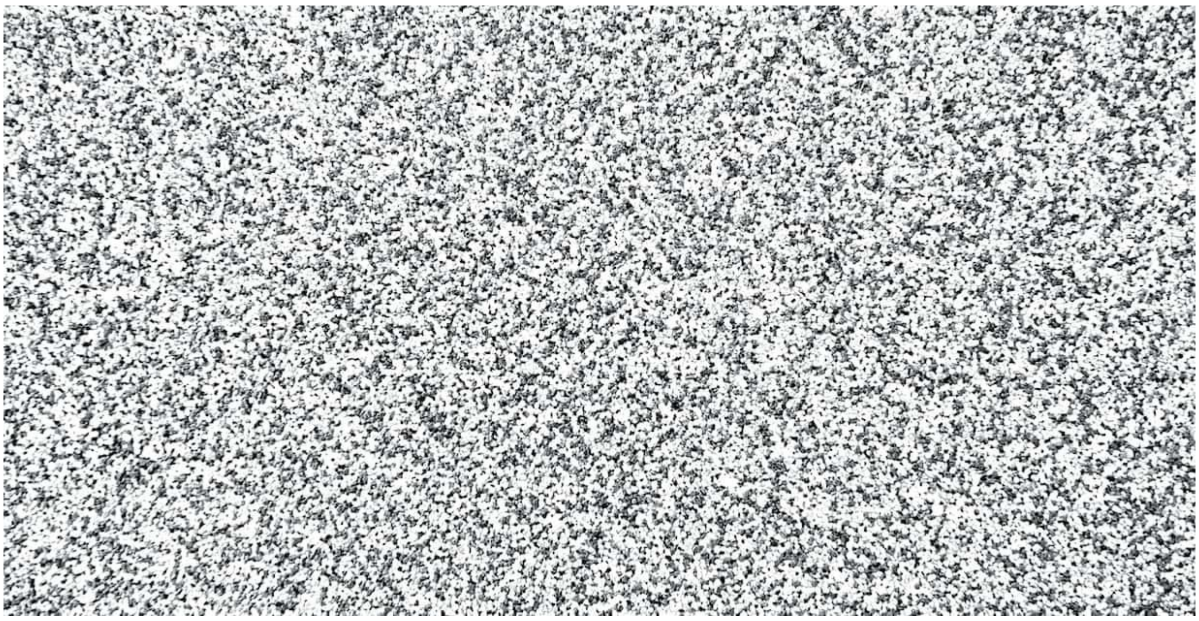
IRGPUA Project

Introduction

AMDIA company has a mission for you :

The internet is broken → All images are corrupted and are unusable.



After some deep analysis performed by the LRE labs, the issue was found :

A hacker named Riri has introduced garbage data at the entry of Internet pipelines

The corruption is as follows :

- Garbage data in the form of lines of “-27” have been introduced in the image
- Pixels got their values modified : 0 : +1 ; 1 : -5 ; 2 : +3 ; 3 : -8 ; 4 : +1 ; 5 : -5; 6 : +3; 7 : -8…
- All images have awful colors and should be histogram equalized

Engineers from AMDIA have found a solution but it is super slow (CPU folk’s issues)

Your goal is to fix the issue in real-time

AMDIA knows you will succeed in the mission and has a final task for you :

Once all images from the pipeline are cleaned up, they want to do some statistics

You should first compute each image’s total sum of all its pixel values
Then sort the resulting images based on this total sum (optional)

Of course, all of this should be done in the fastest way possible

They will provide you with their CPU code to start and test your implementation

Advice

Almost all of the algorithms / patterns used for the project will be practiced in TP

Do the TP and ask questions to have the best base to start from

Implementing a **working** (aka slow) version of the remaining algorithms / patterns should not be too complicated

Once and only once your pipeline is working, you should start optimizing

You will be working in groups of 3. Split tasks (parallel work) and communicate to share ideas about optimization ideas. You should think together but I advise you to code separately so that you all learn and have time to finish the project.

If you have any questions : nicolas.blin@epita.fr

Format

Add clear instructions so that I can run/test your code

Give clear names to files so that I can easily find your implementation (for example scan.cu)

In need to have 3 mains: the CPU provided, the GPU “by-hand” and the GPU “indus”

Project Code and Slides (or report) needs to be sent **before 04/11/2024 - 23h42**

Send me the project by email at nicolas.blin@epita.fr with all your names in the folder name and all CC’d in the email. Expect an ACK from me, else you *must* ping me

Project Expectations

- Have a working pipeline (fix images + **statistics**; sorting on GPU is optional)
- Have optimized algorithms / patterns, memory allocations...
- Have Nsight Compute & **Systems** reports (screenshots) to show your performance analysis
- Have a second working industrial (fast) version using CUB, Thrust for the algorithms (you should minimize the use of hand-made kernel as much as possible for this version)
- Use multiple of the introduced programming tools
- Have a presentation (or report, as you wish) explaining how you programmed, analyzed, and optimized each algorithm / pattern and how you used libraries for your industrial version
- I prefer having algorithms extra optimized rather than great slides ! Don't spend too much time on it
- Project Code and Slides (or report) needs to be sent **before** 04/11/2024 - 23h42

Minimal Project Expectations

- Have a working pipeline (fix images + **statistics**)
- Have a working industrial version with no handwritten kernel
- Have the full final Reduce introduced in class and programmed during hands-on
- Have a working Scan (block + grid level)
- Have a histogram at least as good as last one introduced in class
- Use some of the introduced programming tools
- Have at least one Nsight Systems overall reports analysis

Maximal Project Expectations

- All previous
- Performance / Benchmark Analysis with Nsight Compute of the different Reduce steps
- Different block-wide Scan + Decouple Lookback + Nsight Compute Analysis
- Optimize further all other patterns with introduced techniques
- Use all the introduced programming tools
- Deep Nsight Systems analysis
- More pattern/overall optimizations that were not introduced in this class ;)

Impossible Level

- Working GPU Radix Sort

God Level

- Working GPU Decouple Lookback Radix Sort